VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

NÁVRH A IMPLEMENTACE AUTONOMNÍHO DOKOVÁNÍ MOBILNÍHO ROBOTU
DEVELOPMENT OF MOBILE ROBOT AUTONOMOUS DOCKING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Miroslav Čepl

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. Jiří Krejsa, Ph.D.

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Bc. Miroslav Čepl**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**doc. Ing. Jiří Krejsa, Ph.D.**
Akademický rok:	2018/19

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Návrh a implementace autonomního dokování mobilního robotu

Stručná charakteristika problematiky úkolu:

Jedním z požadavků na mobilní robot je jeho schopnost samostatně vyhledat nabíjecí stanici a připojit se k ní. Zatímco přibližné vyhledání nabíjecí stanice je spojeno s úlohou lokalizace mobilního robotu v okolním prostředí, samotné připojení ke stanici je obvykle řešeno nezávislým navigačním podprogramem. Cílem práce je vytvořit takový navigační subsystém. K přesnému navedení na nabíjecí stanici bude využito vizuálních markerů na stanici a jejich detekce kamerou mobilního robotu.

Cíle diplomové práce:

Seznamte se se systémem ROS a metodami detekce vizuálních markerů.
Vyberte vhodnou metodu detekce a proveďte testovací měření a analyzujte přesnost detekce a veličiny, které ji ovlivňují.
Navrhněte, implementujte a verifikujte navigační algoritmus pro připojení robotu k nabíjecí stanici.

Seznam doporučené literatury:

REN C. Luo, CHUNG T. Liao1, KUO L. Su, KUEI C. Lin, Automatic Docking and Recharging System for Autonomous Security Robot, 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005

OBŠIL, T.: Návrh dokovací stanice pro mobilní robot. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2018. 72 s.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2018/19

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt
Práce se zabývá implementací dokovacího algoritmu pro mobilní robot za pomocí vizuálních markerů. Nejprve jsou prozkoumána již realizovaná řešení, následně je navrhnut dokovací systém. Návrh je ověřen testovacím měřením přesnosti detekce markerů pomocí kamery. Po implementaci systému jsou provedeny testy v simulaci a na konkrétním robotovi. Konečná verifikace ověřuje funkčnost všech částí i celku. Výsledkem je schopnost robota ve zmapovaném prostředí samostatně dojet k nabíjecí stanici a zadokovat, po nabití lze stanici opustit.

Summary
This thesis implements solution for automatic docking for a mobile robot using visual markers. After initial survey of already implemented works, new docking solution is proposed. Feasibility of the solution is verified with tests of marker detection precision. The implementation is tested in a simulation and with a real robot. The functionality of the proposed solution is verified by long-term tests. The result of this work is robot's ability to navigate known environment to find and dock a charging station. After charging the robot is able to safely disconnect from the station.

Klíčová slova
Mobilní robot, ROS, navigace, lokalizace, dokovací stanice, nabíjení

Keywords
Mobile robot, ROS, navigation, localization, docking station, charging

ČEPL, Miroslav. *Návrh a implementace autonomního dokování mobilního robotu.* Brno, 2019. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/116771. 69 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Jiří Krejsa.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně dle pokynů vedoucího a s použitím uvedené odborné literatury.

Bc. Miroslav Čepl

Děkuji vedoucímu doc. Ing. Jiřímu Krejsovi, Ph.D. za odborné vedení, trpělivost, cenné rady a přátelský přístup při zpracování této diplomové práce. Děkuji své rodině a přátelům za podporu.

Bc. Miroslav Čepl

Obsah

1. Úvod

Jedním ze zásadních problémů mobilní robotiky je zdroj energie. Volba zdroje ovlivňuje veškeré parametry robotu. Dostupná kapacita určuje maximální délku aktivního času, dostupný výkon pak omezuje pohyb aktuátorů i výpočetní výkon řídící jednotky. Důraz je kladen na co nejnižší náklady na výrobu a provoz.

Existuje mnoho způsobů, jak napájet mobilní robot. Robot obsahuje mechanickou a elektrickou část. Elektronika je napájena zdrojem napětí, u mechanické části pak záleží na účelu robotu. Pro pohyb a interakci s okolím lze využít elektromotory nebo pneumatické či hydraulické systémy. U takto kombinovaných systémů pak je potřeba řešit přeměny energie pro zajištění chodu celého stroje. Často používaná varianta je společná baterie pro elektroniku i elektropohony. Touto variantou se zabývá tato práce.

Malí roboti se spokojí s tužkovou baterií, velký robot se neobejde bez kvalitního akumulátoru. Společné mají to, že se baterie musí nabíjet nebo měnit. V mnoha aplikacích pouhá výměna baterie postačuje. Nevýhodou je nutná interakce operátora s hardwarem a jen málo systémů podporuje běh systému během výměny. Z dlouhodobého hlediska je lepší varianta, kdy je baterie pevnou součástí robotu a pouze se nabíjí pomocí nabíječky. To umožňuje kontinuální běh robotu a od určitého počtu nabíjecích cyklů je to i ekonomicky výhodnější než primární články.

Zatímco samotný děj nabíjení je velmi dobře zvládnutý, problém nastává při potřebě připojit robot na nabíječku. Jednou z možností, jak tento problém řešit, je dokovací stanice. Když nastane potřeba nabít baterii, robot dojede ke stanici a speciálním navigačním podprogramem se se stanicí spojí. Stanice obsahuje nabíječku určenou pro daný robot. Alternativně je nabíječka součástí robotu a stanice jí poskytuje pouze nabíjecí výkon.

Cílem této práce je prozkoumat možnosti dokování pomocí vizuálních markerů a tento systém navrhnout a implementovat. Jedná se o část komplexního systému autonomního dobíjení robotu. Po rozboru úlohy je provedena rešerše již realizovaných řešení. Také jsou zkoumány základní charakteristiky frameworku ROS a systémy detekce markerů. Po upřesnění cílů je na základě vlastností dostupného robotu navržen systém automatického dokování pomocí AR markerů, poté jsou provedeny testy použitelnosti vybrané detekce. Po implementaci navrženého systému je funkčnost verifikována.

Práce se zabývá návrhem systému pro framework ROS, jehož komunita komunikuje výhradně v anglické jazyce. Z tohoto důvodu bylo rozhodnuto psát kód práce v angličtině, jde o univerzální a používaný přístup.

2. Dekompozice úlohy

Hlavním cílem této práce je implementovat autonomní dokovací algoritmus. To znamená, že na konci této práce by měl robot být schopný samostatně zajet k nabíjecí stanici a spojit se s ní, případně se od stanice odpojit a odjet do pracovní oblasti.

Obecně činnost mobilního robotu závisí na mnoha okolnostech, jako je například účel provozu, režim využívání nebo míra autonomie. Tato práce se zabývá těmito dvěma modelovými situacemi:

1. Mobilní robot se nachází ve vnitřním zmapovaném prostředí a je na dálku ovládán operátorem pomocí příkazů. Za určitých podmínek operátor vznese požadavek na autonomní zajetí do stanice a očekává se, že robot autonomně dojede ke stanici a tam se dobije. Po dokončení nabíjení se od stanice odpojí a čeká na další pokyny.

2. Robot je ovládán vyšší vrstvou řízení, která postupně vykonává naplánované úkoly. Operátor jen kontroluje běh robotu. Nabíjení je jedním z možných úkolů, buďto jako možný cíl nebo jako vnitřní mechanismus zajištění kontinuálního běhu robotu.

V obou případech je pro dosažení cíle zapotřebí několika vlastností. Primárně by se měl robot autonomně a bezpečně pohybovat v prostředí. Fyzický zásah operátora je vnímán až jako poslední možnost nápravy chyby, proto by měl systém být robustní a v případě chyby by měl jít lehce obnovit původní stav. Také je vhodné algoritmus implementovat jako nadstavbu stávajícího systému.

Pro realizaci implementace byl vybrán robot o konkrétní konfiguraci, je tedy nutné navrhnout systém s ohledem na aktuální stav. Prvním logickým krokem je rešerše již realizovaných řešení. Pro úspěšný návrh pak je vhodné znát vlastnosti a omezení použité platformy ROS. Již od počátku se předpokládalo použití vizuálních markerů, je tedy vhodné se s nimi řádně seznámit.

Následovat by mělo zkonkretizování požadavků, aby návrh co nejlépe splnil potřeby robotu.

Návrh systému by měl vzít v úvahu veškeré dostupné informace a vytvořit jasně dané a splnitelné podklady a parametry pro následnou implementaci. Ještě před úplnou implementací systému je vhodné ověřit klíčovou funkcionalitu a stanovit, jestli je vůbec možné navržený systém realizovat. Po implementaci je nutná verifikace, aby bylo možné zhodnotit úspěšnost.

3. Rešerše

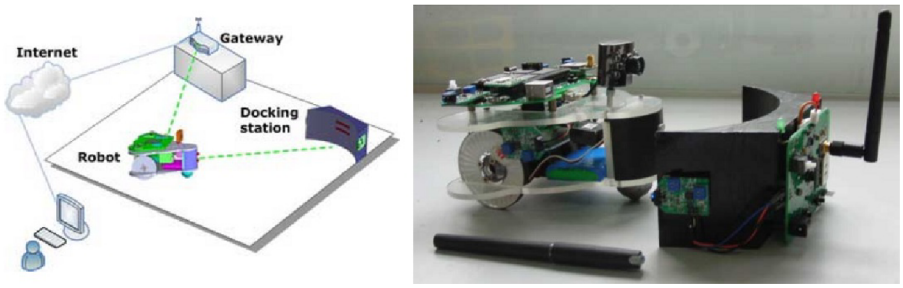
Prvním krokem je rešerše již realizovaných řešení. To umožní získat vhled do problematiky autonomního dokování, ať už se to komplexity či úspěšnosti týče. Tato práce má primárně pracovat s naváděním pomocí vizuálních markerů a kamery, nicméně i ostatní varianty jsou zkoumány pro lepší porovnání. Následně je zkoumán systém ROS a jeho možnosti. V neposlední řadě jsou zkoumány samotné markery, důraz je kladen na systémy dostupné pro ROS.

3.1. Dosud realizovaná řešení

Problematikou autonomního nabíjení se zabývá celá řada výzkumných týmů. Vždy je potřeba zohlednit účel robota a požadované vlastnosti. Některé práce zkoumají nové metody, jiné se zaměřily na maximální robustnost a úspěšnost.

3.1.1. Systém vyvinutý na Southeast University (China)

[1] představuje systém pro monitorování domácnosti. V případě vybití robot dojede k nabíjecí stanici a doplní energii. Se stanicí robot komunikuje pomocí ZigBee, s uživatelem pomocí WiFi. Pro najíždění na stanici byly použity IR diody, díky kterým se robot srovnal vůči stanici. Obr. 3.1 ukazuje navržené řešení. Autoři uvádí úspěšnost 90 % z 60 různých pokusů.



Obrázek 3.1: Řešení představené v [1]

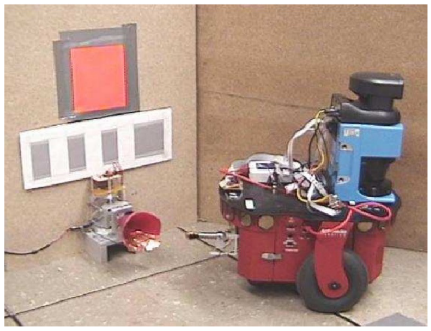
3.1.2. Systém vyvinutý na University of Brescia

[2] ukazuje robot, který k navádění ke stanici využívá dva zdroje světla. Pomocí kamery je vyhodnoceno umístění robota vůči stanici. Jedná se o velmi jednoduché a robustní řešení, autoři uvádí bezchybnou úspěšnost za vhodných podmínek. Na obr. 3.2 je robot zajíždějící do stanice.

3.1.3. Systém nabíjení pro robot Pioneer

[3] ukazuje systém nabíjecí stanice nejen pro robot Pioneer. Pro úvodní lokalizaci stanice byla použita kamera. Stanice obsahovala i laserový maják sloužící k určení orientace vůči stanici. Součástí konektoru je i IR vysílač a přijímač pro detekci spojení. Autoři uvedli, že

Obrázek 3.2: Řešení představené v [2]

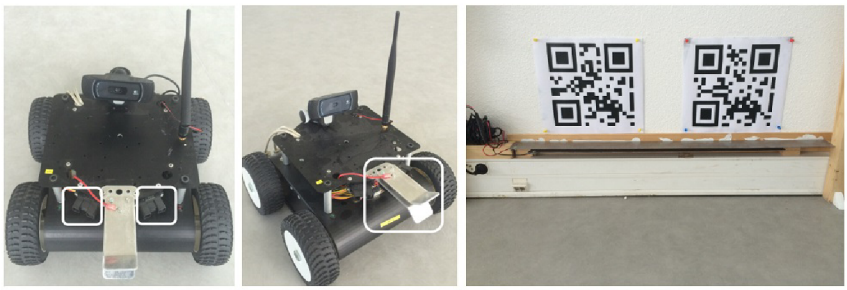


Obrázek 3.3: Řešení představené v [3]

ze 100 pokusů bylo 99 úspěšných mechanicky, v 97 došlo i k elektrickému spojení. Řešení je na Obr. 3.3.

3.1.4. Systém nabíjení pro Wifibot

[4] ukazuje systém pro platformu Wifibot. Robot má dva IR senzory vzdálenosti a web-kameru. Stanice má QR kód, který slouží k přibližnému navedení, IR senzory zpřesňují údaj o vzdálenosti ke stanici. Použitý konektor umožňuje velkou chybu při najíždění, což zlepšuje úspěšnost. Autoři uvádí, že je systém robustní pro celou řadu startovacích pozic robota. Představené řešení je na Obr. 3.4.



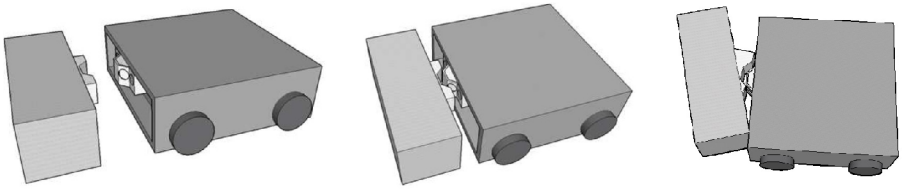
Obrázek 3.4: Řešení představené v [4]

3.1.5. Systém vyvinutý na Singapore Polytechnic

[5] ukazuje systém nabíjecí stanice a robota. Pro kompenzaci chyb byl použit mechanismus schopný náklonu. K navádění byly použity IR vysílače a přijímače. Autoři uvádí úspěšnost 23 z 25 pokusů. Navržené řešení je zobrazeno na Obr. 3.5.

3.1.6. Systém vyvinutý na Shanghai Jiao Tong University

[6] představuje robota pro sklady. Systém používá AR markeru k lokalizaci stanice. Konektor je tvořen třemi nad sebou umístěnými kontakty, které umožňují relativně velkou



Obrázek 3.5: Řešení představené v [5]

chybu při najíždění. Autoři uvádí úspěšnost 97,33 %. Na Obr. 3.6 je zobrazen robot se stanicí.

Obrázek 3.6: Řešení představené v [6]

3.1.7. Systém vyvinutý na Memorial University of Newfoundland

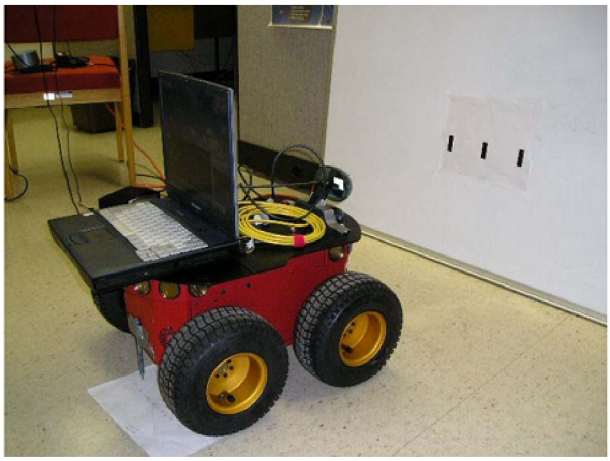
[7] zkoumá možnost najíždění ke stanici pomocí kamery. Na stěně nad stanicí jsou umístěny tři černé vertikální pruhy s definovanou vzdáleností mezi sebou. Robot pak z obrazu vyhodnocuje svou pozici. Navádění dosahovalo přesnosti najetí 1 cm ve směru pohybu ke stanici a 17 cm ve směru kolmém. Na Obr. 3.7 je robot v koncové poloze před stanicí.

3.1.8. Dokování dvou robotů

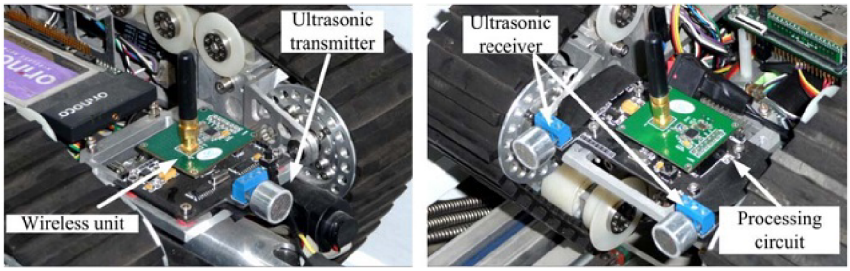
[8] ukazuje navržený systém pro vzájemné spojení dvou mobilních robotů za pomocí ultrazvukového vysílače a dvou přijímačů. Pomocí triangulace je počítána relativní poloha. Vzhledem k charakteru ultrazvukového měření byly vyvinuty i nové metody lokalizace. Autoři uvádí úspěšnost větší než 90%. Na Obr. 3.8 je vidět uspořádání senzorů.

3.1.9. Systém nabíjení pro robot ChungCheng Shin-Kong NO.1

[9] ukazuje vyvinutý systém pro bezpečnostního robota. K lokalizaci stanice robot používá kameru, ve výšce kamery je pak nad stanicí kruh, který je robotem detekován. Robot dosahoval velmi dobrých výsledků a přesnosti najíždění, dle autorů byla úspěšnost 99% ze 100 pokusů. Robot a stanice jsou na Obr. 3.9.



Obrázek 3.7: Řešení představené v [7]

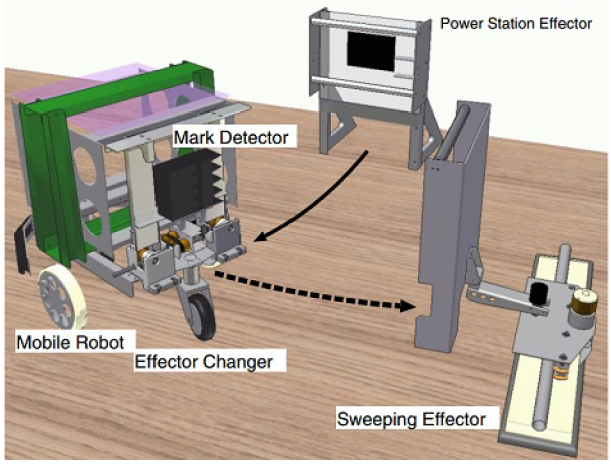


Obrázek 3.8: Řešení představené v [8]

Obrázek 3.9: Řešení představené v [9]

3.1.10. Systém nabíjení pro PC-Yamabico

[10] představuje systém výměnných modulů pro robota PC-Yamabico. První je nabíjecí modul, druhý je pracovní modul. Robot je schopný se s modulem spojit či se od něj odpojit dle potřeby. Pro navedení k modulu je použita kombinace speciální černobílé masky a série IR senzorů. Při testování robot zametal místnost pracovním modulem a poté jej vyměnil za nabíjení pro dobití baterie. Systém umožňuje tvorbu dalších pracovních modulů, čímž se dají jednoduše rozšířit schopnosti robota. Představu o fungování lze získat z Obr. 3.10.



Obrázek 3.10: Řešení představené v [10]

3.1.11. Systém nabíjení pro robot ER-1

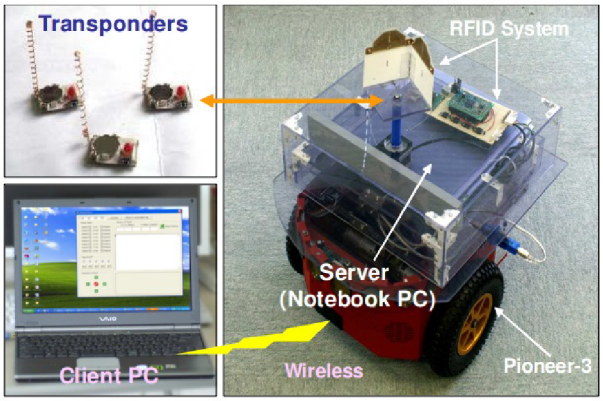
[11] ukazuje nabíjecí systém založený na rozpoznávání z obrazu. Robot je vybavený kamerou. Algoritmus analyzuje obraz a hledá naučené vzorce, například pomocí deskriptorů. Navržený konektor dovoluje velký rozptyl najížděcího úhlu. Podle autorů na konci systém dosáhl 100 % úspěšných spojení se stanicí v 50 pokusech. Pohled na stanici a robota je na Obr. 3.11.

(a) Docking station *(b) Robot-laptop docking mechanism*

Obrázek 3.11: Řešení představené v [11]

3.1.12. Navigace pro Pioneer-3 pomocí RFID

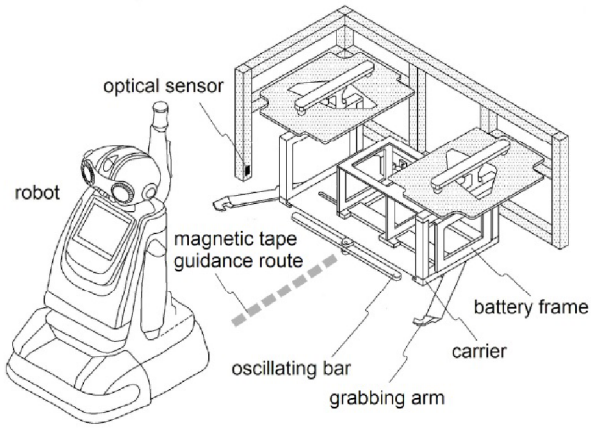
[12] představuje lokalizaci pomocí RFID transpondéru na stanici a dvou směrových antén na robotu. Navržený systém je odolný vůči mnoha zdrojům rušení, nevadí ani překážky nebo přeplněné prostředí. Jde o ověření konceptu, nejsou tedy k dispozici data o dosažitelné úspěšnosti. Pohled na vyvinutý hardware je na Obr. 3.12.



Obrázek 3.12: Řešení představené v [12]

3.1.13. Systém vyvinutý v Industrial Technology Research Institute in Taiwan

[13] ukazuje další možný přístup k autonomnímu nabíjení. Ke stanici je robot naváděn pomocí magnetického pásku na zemi. Stanice má zabudovanou optickou bránu, po detekci robota jej mechanicky zajistí. Při najíždění konektor dovoluje chybu několika stupňů. Popis systému je na Obr. 3.13.



Obrázek 3.13: Řešení představené v [13]

3.2. ROS

I přesto, že autonomní robot představuje velmi komplexní systém, téměř vždy obsahuje nutné základní subsystémy, jako je řízení, lokalizace či plánování. Aby nebylo třeba vždy vyvíjet vše od začátku, vznikla platforma ROS (angl. Robot operating system). Název evokuje dojem, že jde o operační systém, nicméně tomu tak není, k běhu je potřeba reálný OS jako například Ubuntu. ROS v sobě sdružuje základní balíčky pro řešení problémů robotiky, k dispozici jsou návody, online dokumentace a široká uživatelská základna. Platforma je otevřená, každý může přispívat svým kódem. Vše má modulární charakter. [14]

3.2.1. Distribuce

ROS je primárně určen pro OS Ubuntu, ale nic nebrání běhu i na jiných systémech. Základní dělení je na tzv. distribuce, což je soubor vzájemně kompatibilních balíčků a jádra. Distribuce jsou vydávány periodicky, každá druhá má prodlouženou podporu. V době psaní práce mezi prodloužené distribuce patřily Indigo Igloo, Kinetic Kame a Melodic Morenia. Pro ROS Indigo měla být podpora ukončena v dubnu 2019. [15]

Na Obr. 3.14 jsou loga všech distribucí.

Obrázek 3.14: Přehled ROS distribucí.

3.2.2. Balíčky

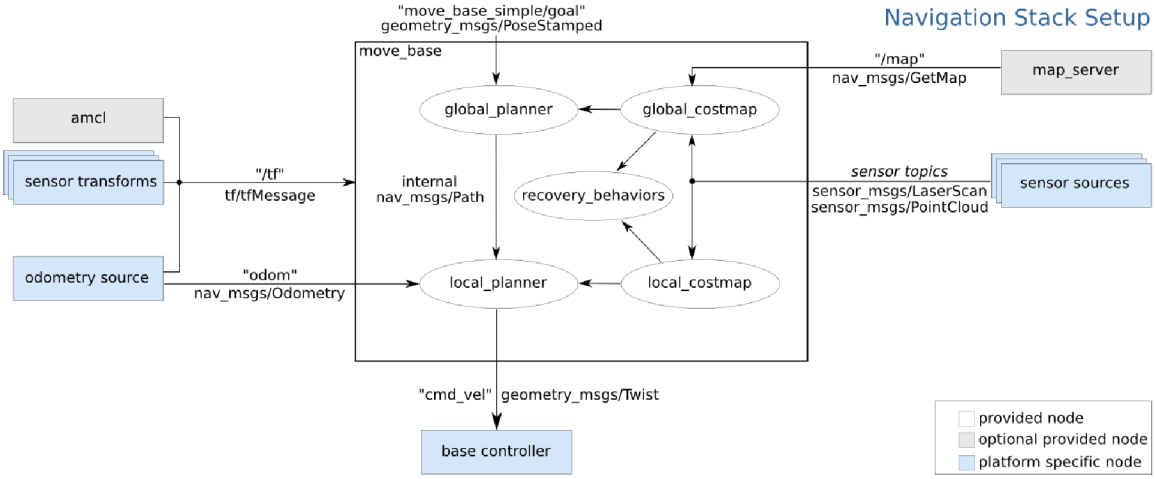
Kromě jádra systému tvoří balíčky základní funkcionalitu. Jedná se o soubor souborů a skriptů, které řeší určitý problém. Základní balíčky mají definovanou strukturu, uživatelské balíčky mohou obsahovat téměř cokoliv.

3.2.3. AMCL

AMCL je balíček pravděpodobnostní lokalizace ve 2D prostředí. Je založený na adaptivní monte carlo metodě, popsané v [16]. Balíček je odvozený od „amcl“ Player driveru Andrewa Howarda. [17] Balíček funguje tak, že data ze senzorů porovnává s mapou a počítá nejpravděpodobnější místo výskytu robota. Z principu lze použít mnoho různých senzorů, ale velmi často se používá například pouze lidar.

3.2.4. Move_base

Move_base je hlavní součástí navigace. Umožňuje plánovat a realizovat cestu ze startu do cíle ve 2D mapě. Je také možné objíždět překážky či trasu dynamicky měnit. [18] Používá lokální a globální plánovač, stejně tak jako lokální a globální costmapy. Globální plánovač navrhuje globální trasu, nepočítá ale s aktuálním stavem mapy. O překážky a dynamické změny v mapě se stará lokální plánovač, který se snaží držet globální trasy. V případě, že nelze cestu uskutečnit, například protože je cesta zablokovaná, může být globální trasa přepočítána. [18] Začlenění move_base v celku znázorňuje Obr. 3.15.

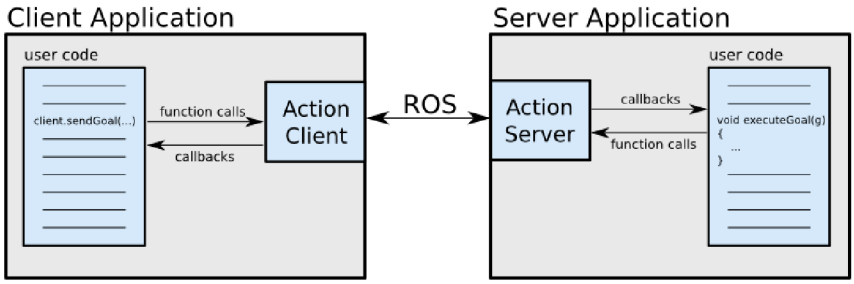


Obrázek 3.15: Přehled návazností na move_base. Převzato z [18]

3.2.5. Action server

Action server je funkcionalita balíčku actionlib[19]. Slouží k vykonávání předvídatelných úkolů. Klient posílá serveru úkol ve formě cíle, server se jej snaží splnit, na konci vrací výsledek. Jelikož může plnění cílů, jako je pohyb robotu, trvat dlouhý čas, je k dispozici možnost zjišťovat aktuální stav. Tímto způsobem stačí klientovi, aby pouze periodicky kontrolovat stav úkolu bez nutnosti použití složitého kódu pro správu úkolu. Formát cíle, výsledku a aktuálního stavu je předem definovaný. Úkol je možné také přerušit.[19]

Schéma vyjadřující začlenění action serveru do systému je na Obr. 3.16



Obrázek 3.16: Příklad použití action serveru, převzato z [19]

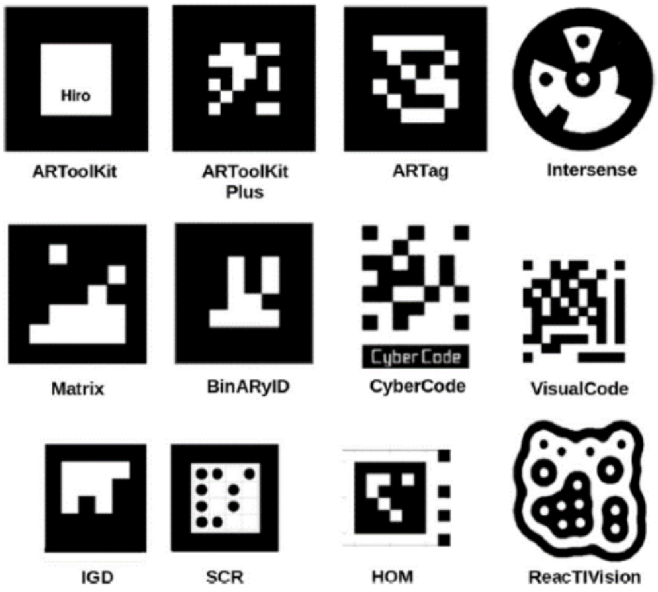
3.2.6. Launch soubory

O hlavní funkcionalitu ROSu se stará modul roscore. Po spuštění roscore jde spouštět další moduly jednotlivě pomocí terminálu. S vyšším počtem modulů se proces stává nepřehledným a časově náročným. Proto vznikly launch soubory. Po zavolání se automaticky spustí všechny potřebné moduly, také je možné velmi snadno nastavit vstupní parametry, nebo je rovnou načítat z konfiguračních souborů. Tím je zajištěna konzistence při spouštění.

3.3. Markery

Obecně jsou markery určité objekty, rozpoznatelné v obraze. Mohou být dvoj či trojrozměrné. Detekce jednoduchých markerů se používá například v radioterapii. [20] V posledních desetiletích se rozmohlo použití tzv. AR markerů (z angl. Augmented Reality – rozšířená realita). Jde o dvourozměrné obrazce, které je možné lehce detekovat, identifikovat. [21] Podobně jako QR kódy mohou obsahovat informace, nicméně v případě AR markerů jde hlavně o schopnost přesného určení polohy obrazce vůči kameře.

[21] nabízí přehled základních systémů AR markerů, čtenář získá přehled o možnostech a omezeních markerů. Na Obr. 3.17 je ukázka typů markerů.



Obrázek 3.17: Přehled různých markerů. Převzato z [6]

Tato práce se zaměřila na systémy dostupné pro framework ROS. Mezi nejznámější patří tyto balíčky:

- Ar_tools (Artoolkit) [22]
- Apriltags [23]
- Ar_track_Alvar [24]
- Aruco [25]
- Fiducials [26]

3.3.1. Ar_tools

Ar_tools je balíček, který používá knihovnu Artoolkit.[22] Artoolkit je knihovna určená pro aplikace s rozšířenou realitou. Autorem je Dr. Hirokazu Kato, ve vývoji pokračují HIT Lab při University of Washington, HIT Lab NZ při University of Canterbury a ARToolworks, Inc v Seattle. [27]

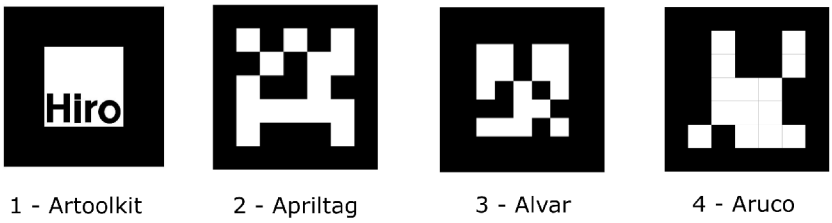
Artoolkit byl použit například v [28], také na něj navazuje systém ARTag[29]. Ukázka je na Obr. 3.18, marker č. 1.

3.3.2. Apriltags

Apriltags je systém představený Edwinem Olsonem v [30] a [31]. Vývoj dále pokračuje pod záštitou The APRIL Robotics Laboratory at the University of Michigan. [32] Primární určení je pro mobilní robotiku, důraz je kladen na přesnost lokalizace. Je možné použít markery o různé složitosti vnitřní mřížky, na balíček navazuje i systém kalibrace kamery AprilCal[33]. Ukázka je na Obr. 3.18, marker č. 2.

3.3.3. Ar_track_Alvar

Ar_track_alvar je balíček založený na knihovně ALVAR. [24] Podobně jako Artoolkit je ALVAR soubor nástrojů pro práci s rozšířenou realitou. Systém je vyvíjen v VTT Technical Research Centre of Finland Ltd. [34] Ukázka je na Obr. 3.18, marker č. 3.



Obrázek 3.18: Markery zkoumané v této práci.

3.3.4. Aruco

Aruco je knihovna pro práci s AR markery, vyvinutá skupinou Ava při the Univeristy of Cordoba. [25] Představena byla v [35] a [36]. Používá knihovnu OpenCV, je k dispozici na mnoha platformách a je kompatibilní s dalšími AR systémy. [37] Ukázka je na Obr. 3.18, marker č. 4.

3.3.5. Fiducials

Jde o knihovnu vyvinutou firmou Ubiquity Robotics. Jako základ používá knihovnu ARUCO. [26]

3.3.6. Porovnání a alternativy

Je samozřejmé, že zde zmíněné systémy nejsou jediné použitelné. Např. [38] navrhuje systém ChromaTag, využívající nejen škálu šedé, ale i barev. Mnoho týmů se pak zabývalo různým porovnáním. [29] [39] [40] Výhody při použití více markerů najednou byly zkoumány například v [41].

Na základě rešerše bylo rozhodnuto, že AR markery jsou vhodný způsob pro lokalizaci robota vůči stanici a bude jich využito při návrhu systému.

4. Upřesnění cílů

Na základě úvodní rešerše, požadovaných parametrů systému a konzultace s vedoucím práce byly stanoveny základní požadavky:

1. Algoritmus je vytvořen pro robot s diferenciálním podvozkem.
2. Dokovací systém je volán operátorem/programem a chová se jako samostatný modul.
3. Veškeré hardwarové úpravy musí být navrženy jako nadstavba nebo doplněk robota.
4. Hlavní funkcionalita je implementována na platformě ROS.
5. Přesného navádění je dosaženo pomocí jednoduché kamery a vizuálních markerů.

Tato práce řeší dva úkoly – zadokování a oddokování, jejichž analýzou lze identifikovat tři různé činnosti:

1. Autonomní jízda ve zmapovaném prostotu
2. Najíždění ke stanici
3. Odjezd od stanice

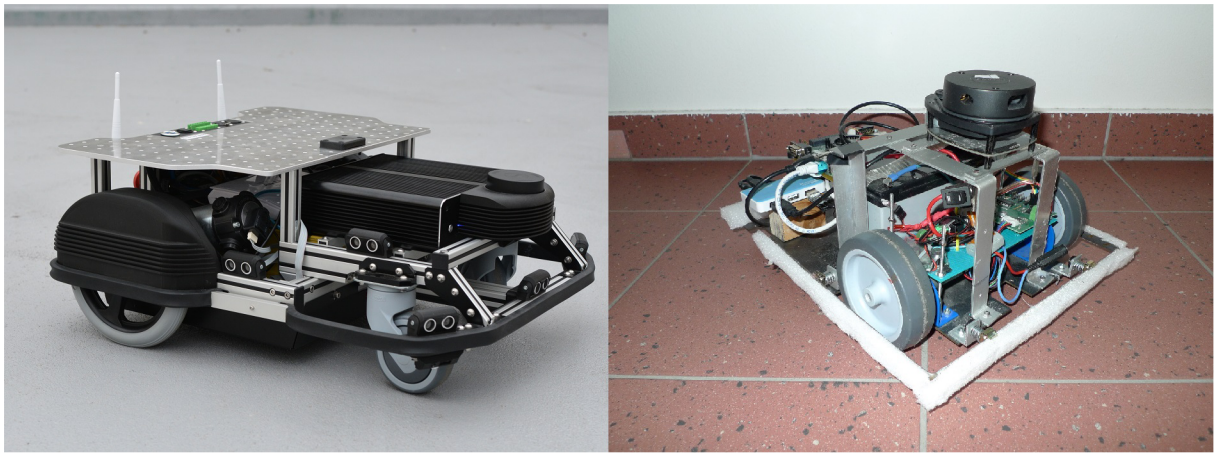
Kombinací činností lze splnit žádané úkoly – zadokování je kombinací a) a následně b), oddokování je ekvivalentem c). Alternativně by mohl robot po odjezdu od stanice i dojet na původní místo. Bylo tedy stanoveno, že by mělo být možné spustit jak jednotlivé činnosti, tak jejich kombinace.

Pro autonomní pohyb má robot k dispozici mapu operačního prostoru. Stanici je tedy možné přiřadit pevné souřadnice a pohyb do prostoru před stanicí je možný pomocí dostupného plánovače.

Při dokování se jedná o relativní pohyb mezi robotem a stanicí. Pro určení relativní polohy je potřeba vybrat vhodnou metodu. Pro přibližování je potřeba také navrhnout a implementovat vlastní plánovač trasy a regulátor pohybu.

5. Návrh dokovacího systému

Původním objektem této práce měl být robot Breach[42], bohužel nebyl v době psaní práce k dispozici volný exemplář, a proto bylo rozhodnuto implementovat a testovat systém na experimentálním robotu Leela. Topologicky jsou tyto dva roboty velmi podobné, rozdílná je poloha lidaru, celková velikost a hmotnost. Obrázek robotu Breach je na Obr. 5.1 vlevo, robot Leela je vpravo.



Obrázek 5.1: Roboty Breach (vlevo, fotografie použita se svolením Bender Robotics) a Leela (vpravo).

5.1. Robot Leela

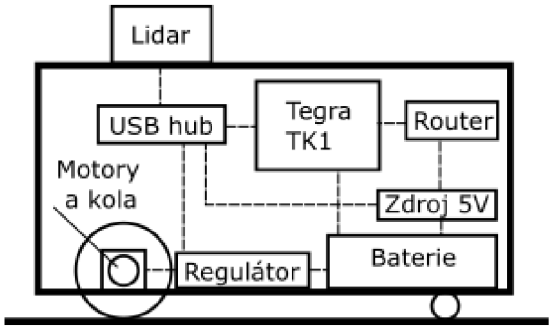
5.1.1. Popis

Jedná se o robot s diferenciálním podvozkem. Pohyb zajišťují dvě kola poháněná DC elektromotory s enkodéry a regulátorem, pro stabilitu je podvozek doplněn o otočné kolečko. Provozní rychlost je přibližně 0,5 m/s. Napájení zajišťuje 12 V olověná baterie s kapacitou 7,2 Ah. Řídícím počítačem je deska Nvidia Jetson TK1. Robot má lidar pro detekci okolí, konkrétně jde o RPlidar A1. V neposlední řadě systém obsahuje i USB hub se zdrojem stabilizovaného napětí a wifi router. Celková velikost robotu je 36x28x19 cm. Schéma propojení základních částí robotu je na Obr. 5.2, fotografie s popisky jednotlivých částí je na Obr. 5.3.

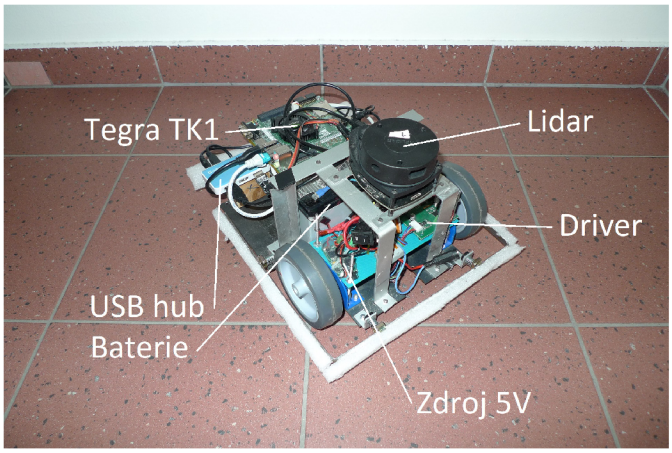
Robot je ideální pro testování nových algoritmů. Díky nízké hmotnosti je možné ho lehce přesunout, v případě poruchy nepředstavuje nebezpečí pro člověka nebo vybavení. Baterie zajišťuje několik hodin nepřetržitého provozu. Díky lidaru a odometrii je robot schopen velmi přesné lokalizace v řádu několika centimetrů. Jediná nevýhoda jsou slabé motory, které neumožňují přesné pohyby při velmi nízkých rychlostech.

5.1.2. Práce s robotem, dostupná funkcionalita

Pro správný návrh dokovacího systému je nutné pochopit, jak se s robotem pracuje. Ke spuštění slouží hlavní kolébkový přepínač. Po spuštění se zapne hlavní deska, wifi router



Obrázek 5.2: Schéma propojení jednotlivých komponent.



Obrázek 5.3: Popis částí robotu Leela.

i lidar. Wifi router vytvoří přístupový bod, pomocí kterého se může uživatel připojit. Hlavní deska má pevně danou IP adresu, a tak se jde po připojení k wifi připojit i k desce pomocí protokolu SSH. V terminálu pak jde spustit ROS a další funkcionalita. Uživatel zpravidla spustí určitý launch soubor v závislosti na úkolu.

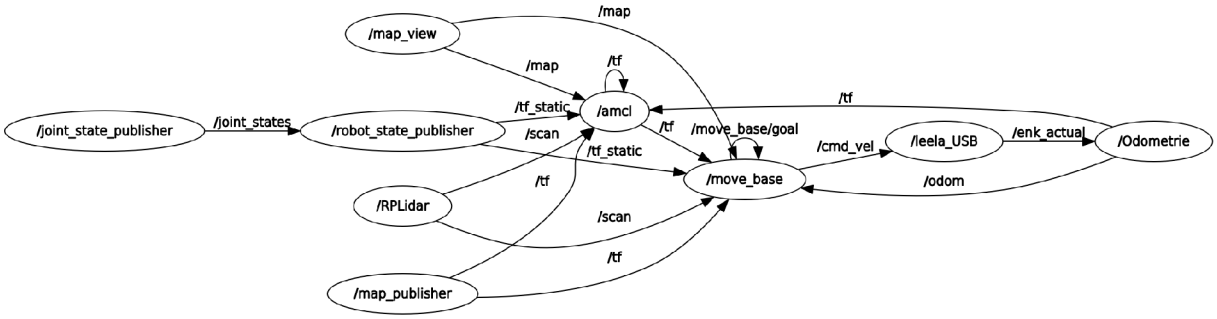
Už v základu obsahuje ROS balíčky pro navigaci a lokalizaci. Jelikož byl robot Leela už dříve používán, existují i fungující launch soubory. Jedním z nich je soubor amcl_301.launch, který umožňuje autonomní pohyb ve zmapovaném prostoru. Uživatel může zadat pomocí prostředí RVIZ cíl a robot pak naplánuje cestu a dojede tam. Této funkcionality je dále využito. Také byl již dříve vytvořen urdf model robotu. Toho je využito při nastavování transformací a v simulaci.

Kromě autonomního pohybu pomocí zmíněného skriptu může robot plnit celou řadu jiných úkolů, jako je například jízda jen pomocí dálkového ovladače, či tvorba nové mapy v neznámém prostředí pomocí operátora.

5.1.3. Popis činnosti původního launch souboru pro autonomní pohyb

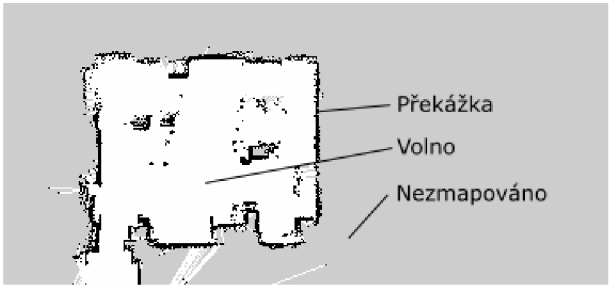
Při zavolání příkazem roslaunch je spuštěno jádro roscore v robotu. Je načten urdf model robotu a inicializovány transformace mezi jednotlivými souřadnými systémy robotu. Transformace jsou k dispozici díky modulu TF. Další skript připojí k low-level řízení, aby

z topiku /cmd_vel předával příkazy motorům. Z low-level desky se pak vrací informace o odometrii. Jednou z nevýhod použitého řešení je, že si low-level deska pamatuje poslední příkaz a snaží se jej neustále splnit. Při výpadku hlavního počítače se tak robot pohybuje nekontrolovaně. Je tedy nutné neustále obnovovat příkaz, a to i v případě, že má robot stát. Pokud by do tohoto topiku posílaly dva skripty různé příkazy, navzájem by se rušily. Je spuštěn kód pro práci s lidarem, naměřená data jsou posílána do topiku /scan. Lidar vrací hodnoty ve formě dvojice úhlu a vzdálenosti. Minimální a maximální detekovatelná vzdálenost je definovaná geometrií senzoru. Vždy hned při zapnutí robotu se lidar automaticky aktivuje a roztočí, což není vždy vítaná vlastnost. Lidar jde zastavit a opět rozběhnout pomocí speciálních příkazů „rosservice call /motor_stop“ a „rosservice call /motor_start“ z příkazové řádky.



Obrázek 5.4: Přehled spuštěných modulů při režimu autonomní jízdy.

Inicializuje se map server, který poskytuje informace o mapě. Mapa je složena ze dvou souborů. Soubor .yaml obsahuje informace o mapě (rozměry, škálování, poloha počátku aj.). Také odkazuje na soubor .pgm, což je samotná mapa, kde každý pixel reprezentuje část 2D prostoru. Bílý pixel značí prázdný prostor, černý pixel překážku. Ukázka mapy je na Obr. 5.5. Parametry použité mapy jsou definovány souborem mapa.yaml



Obrázek 5.5: Ukázka mapy pro plánovač move_base

Spustí se modul AMCL (adaptivní monte carlo lokalizace), který zajišťuje lokalizaci. Modul bere data z lidaru a odometrie a porovnává je s dostupnou mapou. V launch souboru je možné nastavit mnoho parametrů. Za zmínku stojí parametry definující počáteční odhad polohy. Pokud robot při startu stojí v definované poloze, jde jej provozovat ihned, bez nutné korekce operátorem.

Poslední se spustí modul move_base, který zajišťuje plánování a pohyb robota. I zde je možné nastavit mnoho parametrů. Mezi hlavní patří volba lokálního a globálního

plánovače a jejich parametrů. Vzhledem k tomu, že byly tyto parametry již dříve odladěny, není potřeba je při návrhu měnit. Přehled hlavních spuštěných modulů je na Obr. 5.4.

Jako celek se pak systém chová tak, že když operátor zadá cíl na mapě pomocí nástroje RVIZ, snaží se robot naplánovat trasu a následně na místo dojet. Po dojetí robot čeká na další cíl. Na tento koncept se práce snaží navázat.

5.2. Návrh dokovacího systému

5.2.1. Přehled

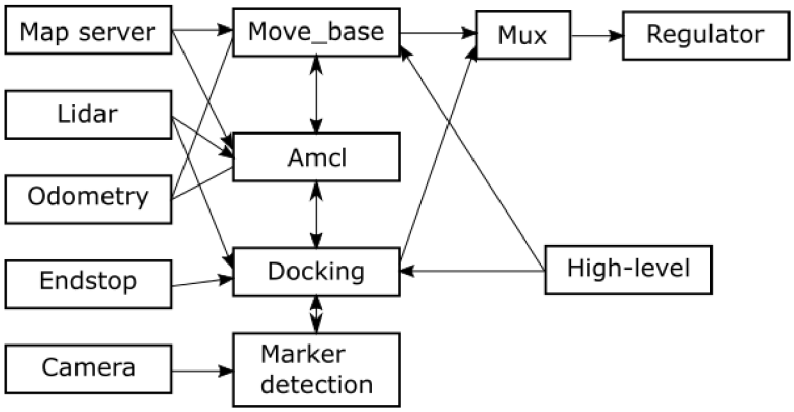
Uživatel pomocí ssh připojení a terminálu spouští launch soubor, který zajišťuje veškerou funkcionalitu, a v defaultním stavu robot čeká na úkol. Pomocí dalšího launch souboru pak uživatel specifikuje konkrétní úkol nebo sadu úkolů, po splnění se skript ukončí a robot je připraven přijmout další úkol. Alternativně by se mohl spustit skript pro kompletní autonomii a uživatel by více nezasahoval.

Uživatel by měl mít k dispozici tyto launch soubory:

- „docking_core“ – Zajišťuje základní funkcionalitu, po spuštění čeká na úkoly
- „dock“ – Z aktuální pozice by měl robot dojet ke stanici a zadokovat
- „undock“ – Robot by se měl odpojit od stanice a odjet do určité vzdálenosti

5.2.2. Softwarové uspořádání

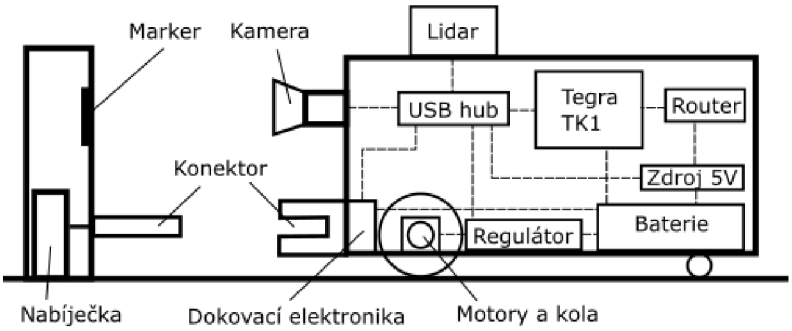
Celý systém zahrnující dokování vychází z původní funkcionality autonomního pohybu. O lokalizaci se stará modul AMCL, autonomní pohyb po mapě řeší modul move_base. Hlavní funkcionalitu dokování obstarává nový balíček. Pro detekci a zpracování dat z kamery je použitý jeden z dostupných balíčků v ROSu. Jelikož se o pohyb stará balíček move_base i balíček dokování, měl by systém obsahovat i mechanismus, jak mezi nimi jednoduše přepínat, například pomocí softwarového multiplexeru. Neměla by chybět detekce úspěšného spojení se stanicí ve formě koncového spínače (endstop). Hlavní logiku obstarává samostatný modul (master). Přehled všech navrhovaných modulů je na Obr. 5.6.



Obrázek 5.6: Moduly navrženého řešení.

5.2.3. Fyzické uspořádání

V operačním prostoru je umístěna dokovací stanice, která zajišťuje dobíjení baterie. Na stanici je umístěn AR marker, který je robot schopen detekovat a pomocí něj se i lokalizovat. K detekci markeru je použita kamera. Robot by se měl se stanicí spojit pomocí speciálního konektoru, jehož součástí je detekce úspěšného spojení. Kamera i konektor by měly být pouze nadstavbou stávající konstrukce robota. Schéma uspořádání je na Obr. 5.7.



Obrázek 5.7: Schéma propojení komponent robotu včetně dokovací funkcionality.

5.2.4. Detailní popis jednotlivých úkolů

Jízda ke stanici a zadokování (Dock)

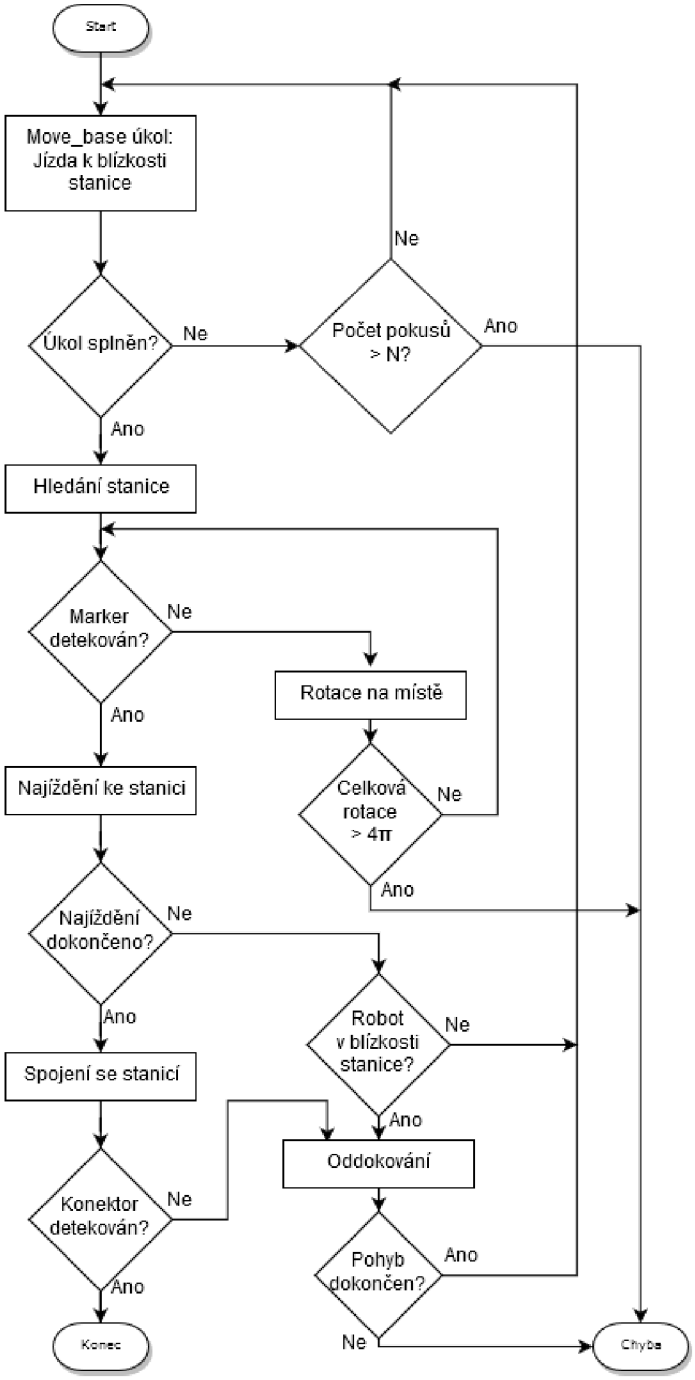
Celkový proces je složen z následujících subprocesů:

1. Start algoritmu operátorem nebo hlavním programem.
2. Vyhledání vhodné stanice
3. Jízda robota k přibližné pozici stanice
4. Lokalizace stanice a naplánování trasy k ní.
5. Přiblížení ke stanici do spojovací vzdálenosti.
6. Spojení se stanicí.

Prvním krokem je jízda do přibližné polohy stanice pomocí move_base. Z hlediska systémovosti je žádané, aby šlo snadno definovat pozici stanice pomocí launch souboru. Po dojetí na místo očekávaného dokování robot začne detekovat stanici. V případě neúspěchu by se měl robot pokusit o rotaci na místě a případné nalezení stanice. Po úspěšné detekci robot vyhodnotí svoji polohu a naplánuje cestu ke stanici. Při najíždění ke stanici je k určení polohy použita kamera sledující marker, pro pohyb je vhodný vlastní plánovač. Po dojetí ke stanici se s ní robot spojí. Pokud při najíždění nastane chyba, robot akci přeruší. Pokud se nachází v určité vzdálenosti od stanice, je žádané, aby se robot vrátil do dokovací pozice a pokus opakoval. V opačném případě je nahlášena chyba a je nutný zásah operátora. Postup je na Obr. 5.8.

Oddokování (Undock)

Předpokládá se, že se robot nachází v definované poloze – ve stanici. Jde tedy pouze o potřebu vyjet ze stanice o definovanou vzdálenost. Je ale potřeba zajistit, aby nedošlo ke kolizi, je tedy nezbytné, aby robot kontroloval okolí a byl schopen bezpečně zastavit při detekci překážky. Také je nutné počítat s možností, že je robot zablokovaný dlouhodobě. Pokud by nedošlo k odjetí v určitém časovém intervalu, je nutné zahlásit chybu, aby překážku odstranil operátor.



Obrázek 5.8: Vývojový diagram pro úkol dokování.

6. Analýza přesnosti detekce markerů

Aby navržený systém fungoval, je nutné vybrat vhodnou kameru a systém markerů. Poté je třeba otestovat limity detekce, aby mohla být stanovena případná omezení při následné implementaci.

6.1. Volba algoritmu, markerů a kamery

Problematika získávání obrazu a následného zpracování je velmi rozsáhlá. Tato práce se snažila zaměřit na praktické aspekty nutné ke splnění cílů.

6.1.1. Výběr kamery

Bylo stanoveno, že se má jednat o volně dostupnou kameru s malými rozměry. Připojení k počítači obstarává hub s USB 2.0, což limituje přenos obrazu na 720p 30 fps. Vybrána byla kamera Live! Cam Sync HD, zobrazená na Obr. 6.1, protože má velmi malé rozměry, přijatelné vlastnosti a příznivou cenu. Při testování se předpokládalo, že kamera bude mít minimální vliv na výsledky, protože se předpokládal provoz v dobře osvětleném prostředí.

Obrázek 6.1: Kamera Live! Cam Sync HD

6.1.2. Kalibrace kamery

Před použitím byla kamera kalibrována pomocí kalibračního obrazce. Tím vznikl soubor s kalibračními maticemi dle [43]. Výsledek je uložen v souboru camera_info.yaml.

6.1.3. Výběr AR algoritmu

Vzhledem k množství dostupných balíčků se přikročilo k otestování jednotlivých možností, aby byl na konci vybrán balíček, který vyhovuje této aplikaci. Z Rešerše AR markerů pro ROS jsou jako potenciální kandidáti vybrány tyto balíčky:

- Alvar

- Aruco
- Fiducials
- Ar_tools (Artoolkit)
- Apriltags/ Apriltags2

Vzhledem k složitosti systému je důležité, aby daný balíček nativně podporoval použitou ROS distribuci. Robot Leela má kvůli základové desce nainstalován systém ROS Indigo, bez možnosti použití vyšších verzí. Robot Breach má mít verzi ROS Kinetic, stejně jako počítač pro operátora. V budoucnu se plánuje přejít na vyšší verze, jako je ROS Melodic. Přehled nativní podpory jednotlivých balíčků je v tabulce 6.1.

Tabulka podpory ROS distribucí:

Tabulka 6.1: Tabulka podporovaných distribucí

	Aruco	Ar_tools	Alvar	Fiducials	Apriltags
podporované ROS distribuce	Indigo Kinetic	Indigo	Indigo Kinetic Lunar Melodic	Indigo Kinetic	Indigo Kinetic

Jelikož balíček Fiducials závisí na balíčku Aruco a není příliš známý, je vyřazen z možností. Balíček Ar_tools má nativní podporu pouze pro ROS Indigo, ale je velmi známý, a proto je ponechán do dalšího kroku výběru.

6.1.4. Instalace a ověření funkce

Nejprve je testována jednoduchost a bezproblémovost instalace jednotlivých balíčků. Instalace je prováděna na dvou virtuálních instalacích systému Xubuntu s distribucí ROS Indigo a ROS Kinetic. Je testováno, jestli je balíček vůbec možné jednoduše nainstalovat a jak složité je zprovoznit detekci markerů. Výsledky snažení shrnuje tabulka 6.2.

Tabulka 6.2: Tabulka podporovaných distribucí

	ROS Kinetic	ROS Indigo
Alvar	Nainstalováno, detekce funkční	Nainstalováno, detekce funkční
Aruco	Nainstalováno, detekce funkční	Nainstalováno, detekce funkční
Ar_tools	Nainstalováno, detekce nefunkční	Nelze nainstalovat
Apriltags	Nainstalováno, detekce funkční	Nainstalováno, detekce funkční

Kromě balíčku Ar_tools se podařilo všechny balíčky nainstalovat a zprovoznit. Liší se ale čas a úsilí k tomu potřebné. Na základě zkušeností během instalace je výběr zúžen na balíčky Alvar a Apriltag.

6.1.5. Test pracovního rozsahu markeru

Další velkou limitací je prostorový pracovní rozsah detekce v závislosti na velikosti markeru. Při dokování stanice je požadováno, aby robot najel do těsné blízkosti, ale zároveň

byl schopný detekovat stanici z dostatečné vzdálenosti při navigaci k ní. Proto je proveden test, kdy je měřena přibližná minimální a maximální vzdálenost detekce.

Minimální vzdálenost je určena jako místo, kdy algoritmus periodicky detekuje marker. Toto místo je ovlivněno hlavně výškou záběru kamery. Balíček Alvar detekuje markery z větší vzdálenosti oproti Apriltag, pravděpodobně protože se dynamicky přizpůsobuje obrazu a potřebuje mít v záběru bílý pruh kolem markeru. Maximální vzdálenost je určena jako místo, kde algoritmus detekuje marker v přibližně 50 % případů. Ze zkušeností s měřením vyplývá, že se úplná ztráta detekce projevuje jen několik centimetrů od určeného místa, což indikuje silně nelineární chování. Dá se předpokládat, že největší vliv na maximální vzdálenost bude mít rozlišení obrazu a jeho ostrost. S tím korespondují výsledky, kdy oba algoritmy dosahují velmi podobných hodnot.

Výsledky orientačního měření jsou v tabulce 6.3, kde h je velikost markeru a d je vzdálenost markeru od kamery.

Tabulka 6.3: Výsledky testů minimální a maximální vzdálenosti detekce

	Apriltag			Alvar	
h[mm]	d_{min}[cm]	d_{max}[cm]	h[mm]	d_{min}[cm]	d_{max}[cm]
30	4,3	163	30	4,5	165
35	5	175	35	5,5	175
40	5,7	217	40	6,3	208
50	7,5	265	50	7,5	250
60	8,9	333	60	9	306
70	10,3	386	70	10,5	371
80	11,6	433	80	11,9	430
100	14	528	100	14,8	536
120	17,5	>560	120	17,5	>560

Dle výsledků jsou balíčky srovnatelné. Pro potřeby dokování bylo stanoveno, že vyhovuje velikost markeru h = 40 mm nebo h = 50 mm. Zde je zvoleno, že bude použit marker o velikosti h = 50mm.

6.1.6. Výkon

V neposlední řadě je důležitá výpočetní náročnost. Obraz z kamery je publikovaný s frekvencí 14 Hz, je to tedy maximum i pro detekci markerů. Běžící skript Apriltag publikuje detekce s frekvencí 4 Hz, skript Alvar publikuje s frekvencí 8 Hz. Skript Apriltag také potřebuje více výkonu pro svůj běh, což je v dané situaci nežádoucí.

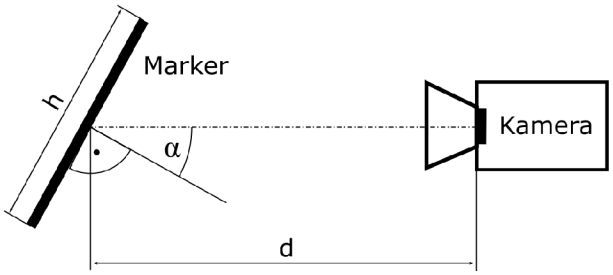
Vzhledem ke všem skutečnostem byl zvolen balíček Alvar. Mezi hlavní důvody patří snadná instalace, intuitivní práce s konfigurací, možnost použít předem vygenerovaných markerů a menší nároky na výkon. Balíček Alvar má také podporu v nejvíce distribucích, oproti tomu Apriltag musí použít jinou verzi pro ROS Indigo a Kinetic a proto není zaručena konzistentnost.

Výběr samotného algoritmu ale nezaručuje, že kvalita detekce bude dostatečná pro tuto práci. Je nutné další testování konkrétních vlastností.

6.2. Testovací měření jednoho markeru

6.2.1. Test maximálního úhlu detekce markeru

Je potřeba zajistit, že je robot schopný detekovat marker během najíždění na stanici. Jako limitní případ je uvažováno, že je marker otočený o 60° vůči kameře. Orientačním měřením byl zjištěn maximální úhel mezi normálou markeru a spojnicí marker-kamera $\alpha = 80°$, což požadavkům vyhovuje. Použit byl marker o straně h = 50 mm ve vzdálenosti d = 40 cm od kamery. Uspořádání při měření ukazuje Obr. 6.2.

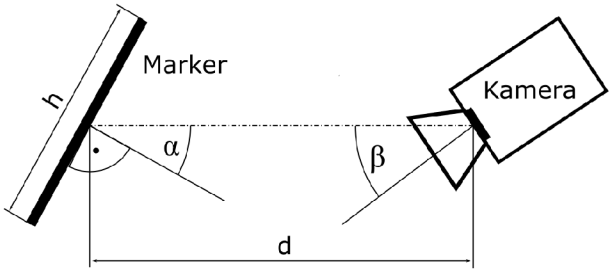


Obrázek 6.2: Uspořádání při měření maximálního úhlu.

6.2.2. Testy přesnosti

Pro lokalizaci robota vůči stanici je potřeba, aby robot správně určil polohu markeru v prostoru a zpětně spočítal svou polohu vůči stanici. Proto byla provedena série měření, kdy jsou kamera a marker umístěny do referenčních vzdáleností a natočení, která jsou pak porovnávána s detekovanými hodnotami. Pro měření bylo použito uspořádání na Obr. 6.3, tímto způsobem se dá simulovat jakákoliv pozice v prostoru a je přesně určena vzdálenost mezi objekty.

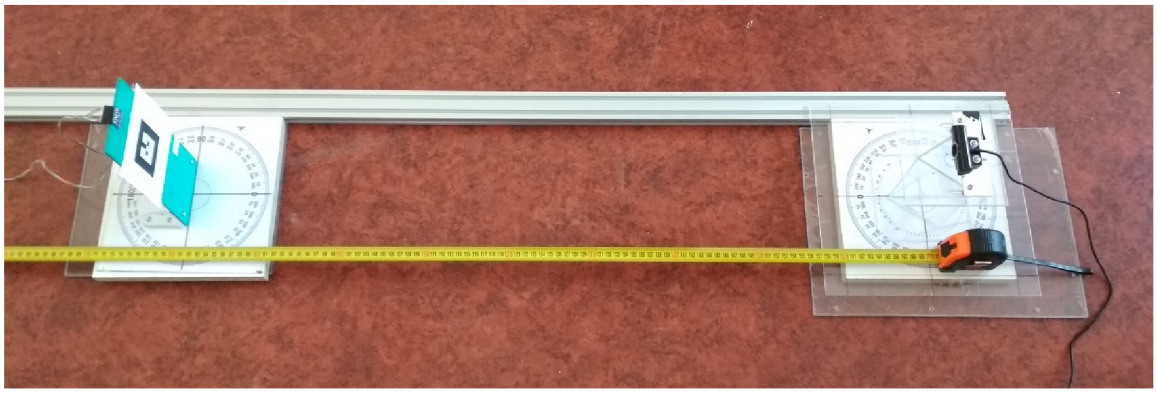
Výstupem detekce markeru je poloha markeru v souřadném systému kamery a jeho natočení vůči kameře ve formě kvaternionu.



Obrázek 6.3: Uspořádání při obecném měření.

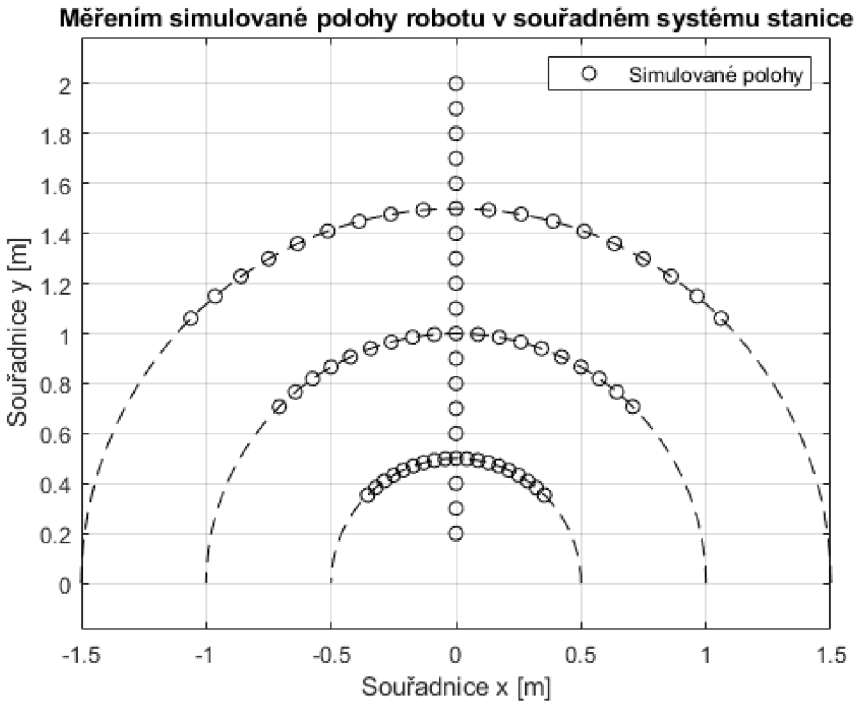
Měření proběhlo při přirozeném osvětlení (orientačně 450 lux) a při osvětlení zářivkami (orientačně 200 lux). Měřicí přípravek je na Obr. 6.4. Vzhledem k použitým měřidlům je odhadovaná přesnost referenčních hodnot měření v jednotkách milimetrů pro vzdálenost a jednoho stupně pro úhly, což je dostačující.

Byla provedena celá série měření. Pokud by pozice markeru byla pevná, tak lze z parametrů α, β a d určit referenční polohu kamery v rovině před markerem. Obr. 6.5



Obrázek 6.4: Realizace měření.

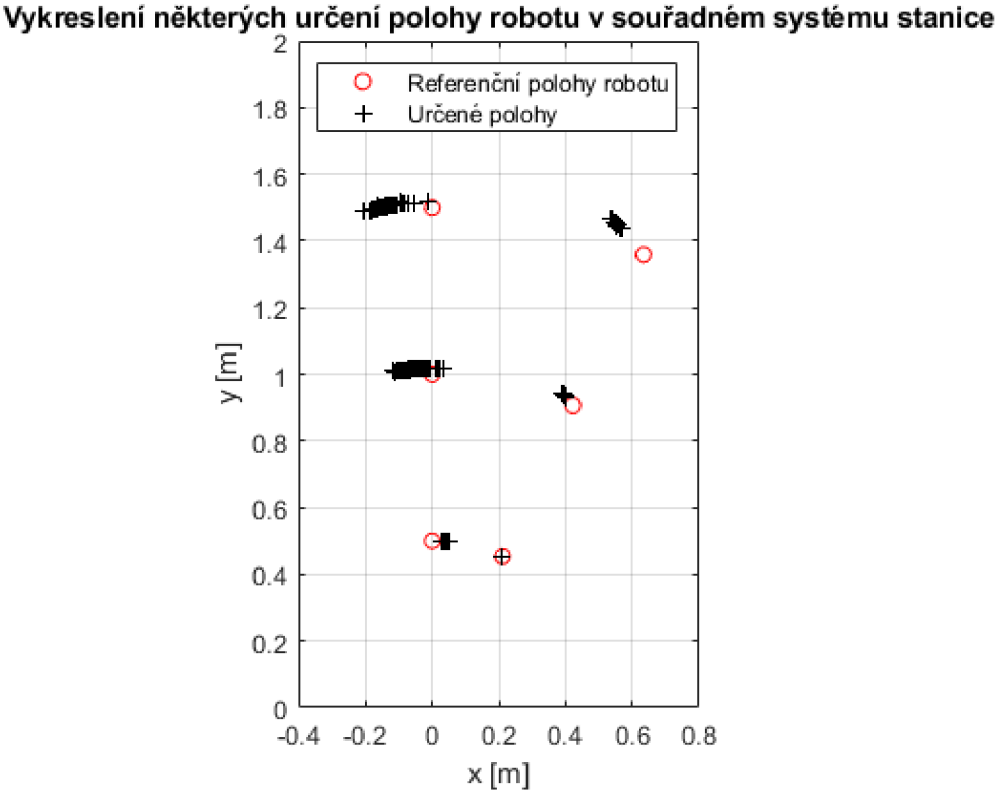
ukazuje referenční body, které byly simulovaně měřeny, vztažené k pevné pozici markeru. Pro ukázku jsou na Obr. 6.6 vyneseny některé referenční body i s přepočtenými body měření. Při každém měření v referenčním bodě bylo provedeno 100 detekcí markeru, aby bylo možné data zpracovávat statisticky.



Obrázek 6.5: Simulované polohy robotu.

Vyhodnocení: Jak vzdálenost kamery a markeru ovlivní chybu určení vzdálenosti

Mezi nejdůležitější vlastnosti patří schopnost správně určit vzdálenost mezi kamerou a markerem. V každém referenčním bodě s parametry d = 20 až 200 cm, $\alpha = 0°$, $\beta = 0°$ je pro každou detekci určena euklidovská vzdálenost, pro analýzu je použita střední hodnota.



Obrázek 6.6: Příklad naměřených dat

Na Obr. 6.7 je vynesena chyba detekované vzdálenosti vůči referenční vzdálenosti d v závislosti na referenční vzdálenosti.

Očekávání bylo, že s rostoucí vzdáleností bude chyba větší, což se potvrdilo. Nicméně nejde o lineární závislost. Pro potřeby dokování stačí, že je poloha přesně určena v blízkosti stanice. Ve vzdálenosti menší než 1 m je chyba menší než 1 cm, což je dostačující.

Vyhodnocení: Jak vzdálenost kamery a markeru ovlivní chybu určení natočení markeru

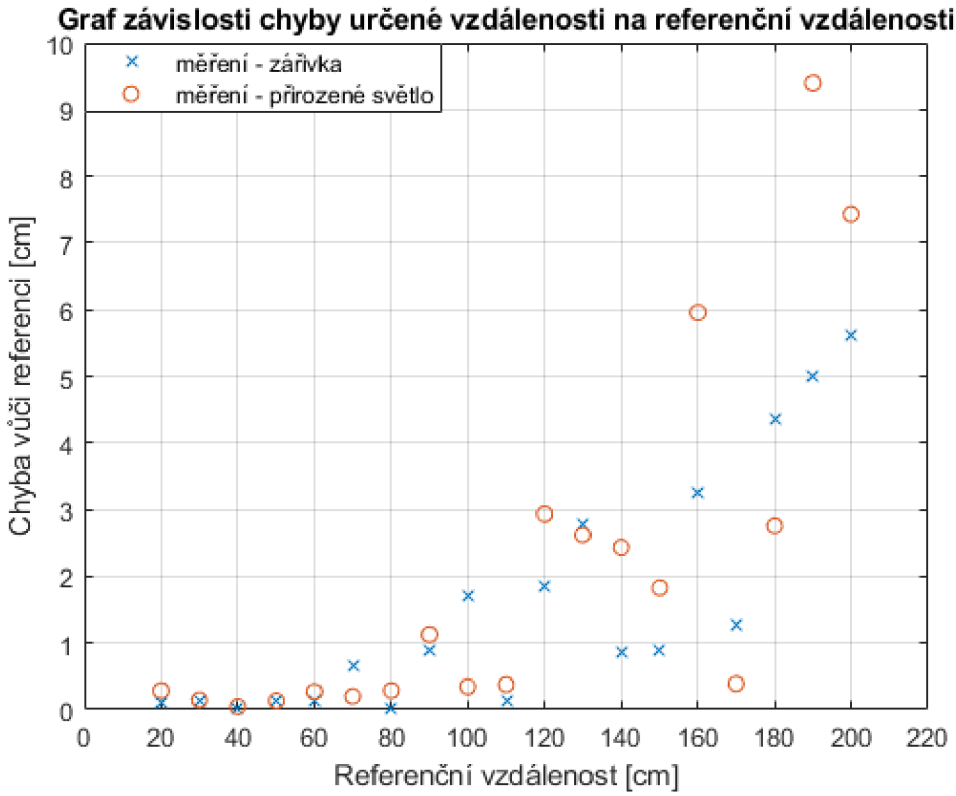
Druhou důležitou vlastností je schopnost určit natočení markeru. Natočení je zásadní při určení polohy robota, a proto by chyba měla být co nejmenší. Na Obr. 6.8 je vynesen graf pro parametry d = 20 až 200 cm, $\alpha = 0°$, $\beta = 0°$.

Očekávání bylo, že s rostoucí vzdáleností bude chyba větší, což se potvrdilo. I na malé vzdálenosti je chyba kolem 5°, ve větších vzdálenostech může být chyba větší než 10°, jedná se o relativně velké chyby.

6.2.3. Detekce za pohybu

Na závěr byla testována schopnost detekovat pohybující se markery. Tato vlastnost je kritická při pohybu robotu, kdy příliš rychlá rotace by mohla vést ke špatné detekci a následnému přerušení celé dokovací rutiny.

Dá se očekávat, že vliv rotace robotu bude mnohem větší než vliv od dopředného pohybu. Proto bylo navrženo měření na Obr. 6.9. Krokový motor pohybuje kamerou

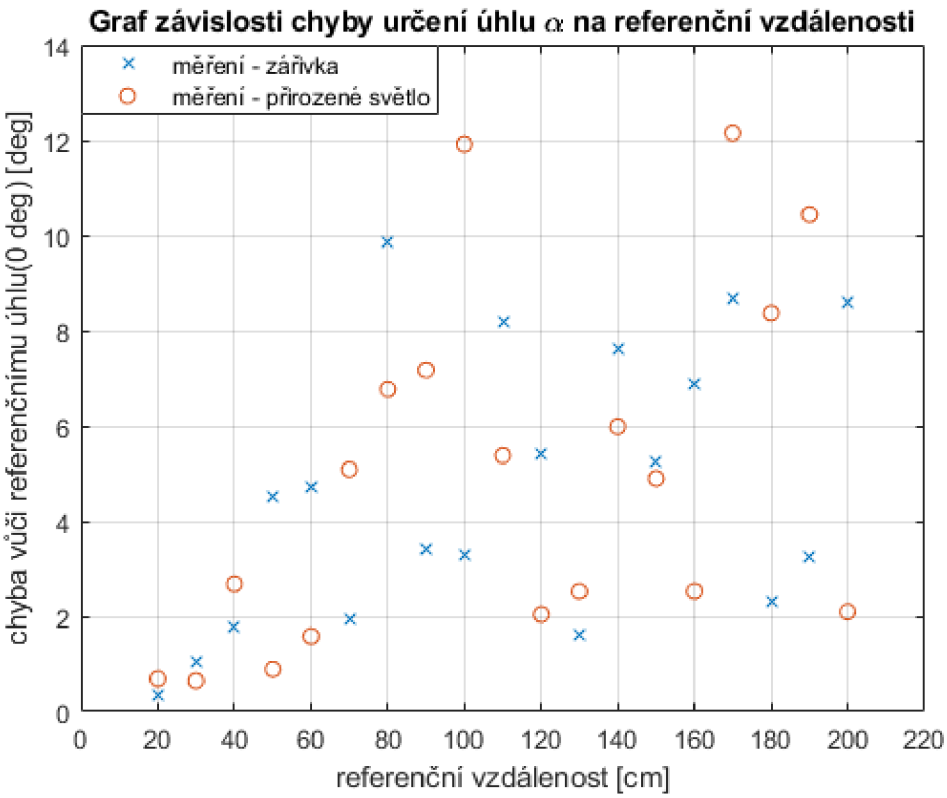


Obrázek 6.7: Chyba určení vzdálenosti.

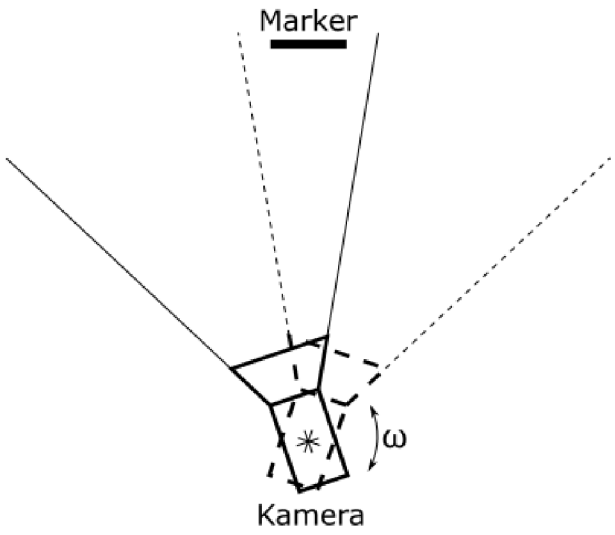
definovanou úhlovou rychlostí tam a zpět a je zaznamenáváno, kolik procent zpráv obsahuje korektní detekci markeru. Pro jednoduchost je marker vždy v záběru. Výsledkem je graf 6.10 určující procenta úspěšné detekce. Z grafu byla určena mezní úhlová rychlost $\omega = 0{,}15$ rad/s. Při vyšší rychlosti klesá úspěšnost detekce. Detekce neklesla nikdy na nulovou hodnotu z důvodu, že na konci rotace musí kamera zpomalit na nulovou rychlost, aby se následně mohla pohybovat opačným směrem, a tak tato fáze umožňuje detekci. Nicméně z grafu jasně vyplývá, že od mezní rychlosti úspěšnost detekce prudce klesá a jde tedy o validní experiment.

6.2.4. Závěr z měření jednoho markeru

Schopnost určit vzdálenost mezi markerem a kamerou je velmi dobrá, ale určení natočení markeru je nepřesné. Pro určení přesné polohy ve větší vzdálenosti nemusí být jeden marker dostatečný. Řešením by mohlo být použití dvou markerů ve vodorovné rovině v určité vzdálenosti od sebe. Předpokládá se, že robot stojí na zemi, zvýšená přesnost je potřebná pouze ve vodorovné rovině. Při použití dvou markerů nezávisí na jejich natočení v prostoru, ale pouze na vzdálenosti k nim. Proto bylo provedeno testovací měření se dvěma markery.



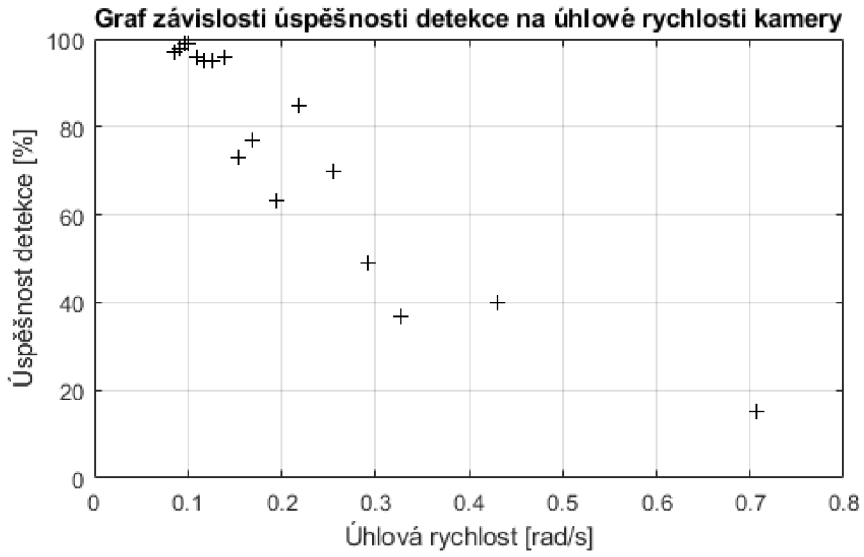
Obrázek 6.8: Chyba určení úhlu α.



Obrázek 6.9: Schéma měření za pohybu.

6.3. Měření se dvěma markery

Hlavní myšlenka použití více markerů je založena na potřebě přesněji určit polohu, aniž by se zvětšila velikost použitého markeru. K původnímu markeru tedy přibyly dva mar-



Obrázek 6.10: Výsledky měření za pohybu

kery, každý z jedné strany. Pokud bude kamera ve stejné úrovni jako markery, je možné informaci o vertikální poloze markeru zanedbat.

Dá se předpokládat, že pro daný pracovní rozsah existuje optimální vzdálenost mezi markery. Při velké vzdálenosti by je kamera brzo ztratila ze záběru. Pokud však by byly moc blízko sebe, zvyšuje se chyba lokalizace a uspořádání ztrácí smysl.

Pro ověření této myšlenky bylo provedeno měření ve dvou místech před stanicí, kamera vždy směřovala směrem ke střednímu markeru. Parametrem je vzdálenost markerů od sebe, výstupem je určená poloha robotu vůči stanici. Použité rozložení a grafické výsledky jsou na Obr. 6.11.

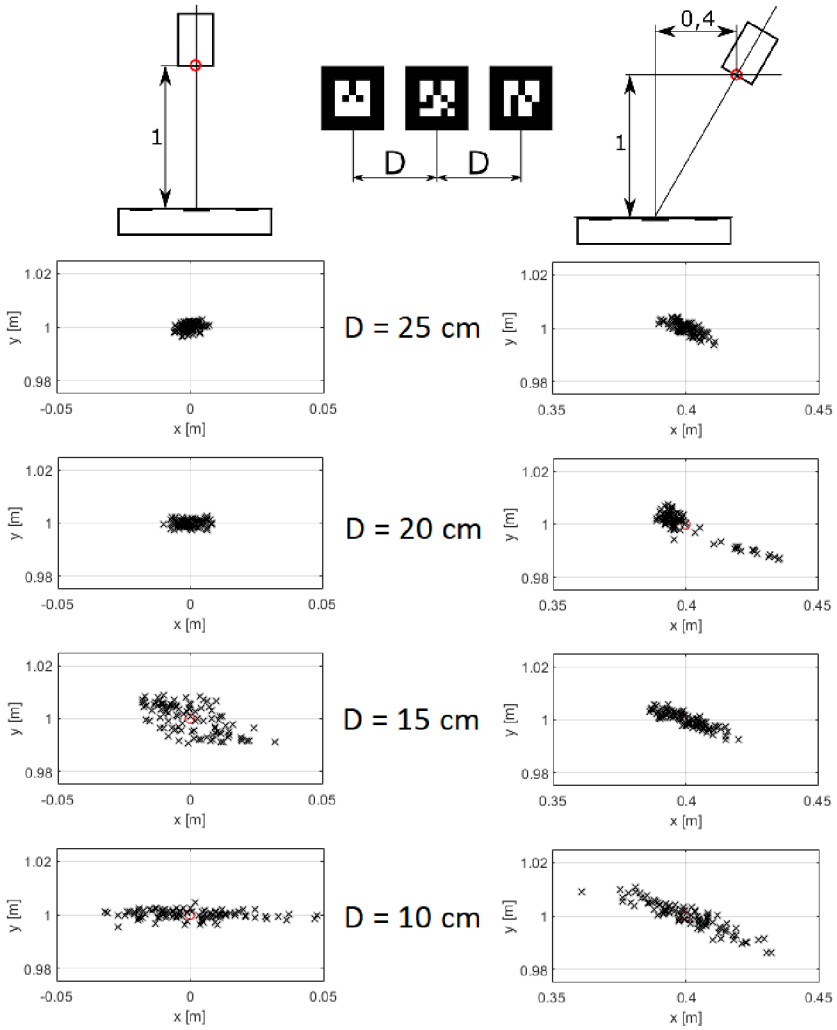
Z měření a testování vyplývá, že i nejmenší testovaná vzdálenost dokáže plnit svůj účel. Stanice tedy může zabírat minimum místa z hlediska šířky. Navržené uspořádání efektivně nahrazuje velký marker, při zachování přesnosti ve vodorovném směru a minimální výšky jednoho markeru.

Balíček Alvar je schopen detekovat i uspořádání více markerů jako jednoho celku. To by defaultně zajistilo dobrou přesnost, nicméně od této varianty bylo upuštěno, protože je nutné, aby robot určil polohu i v případě, že jeden z markerů nevidí.

6.3.1. Porovnání výsledků

Z výsledků vyplývá, že lokalizace za pomocí dvou markerů je vždy lepší než pouze pomocí jednoho. Z hlediska absolutní chyby polohy je použitelný i jeden marker, avšak velký rozptyl určené polohy by nepřispíval ke stabilitě naváděcího algoritmu. Jeden marker je také více náchylný na chybu při umisťování markeru na stanici.

Porovnání výsledků určení polohy z měření jednoho a dvou markerů pro referenční polohu [0,1] (v souřadném systému stanice) je na Obr. 6.12.

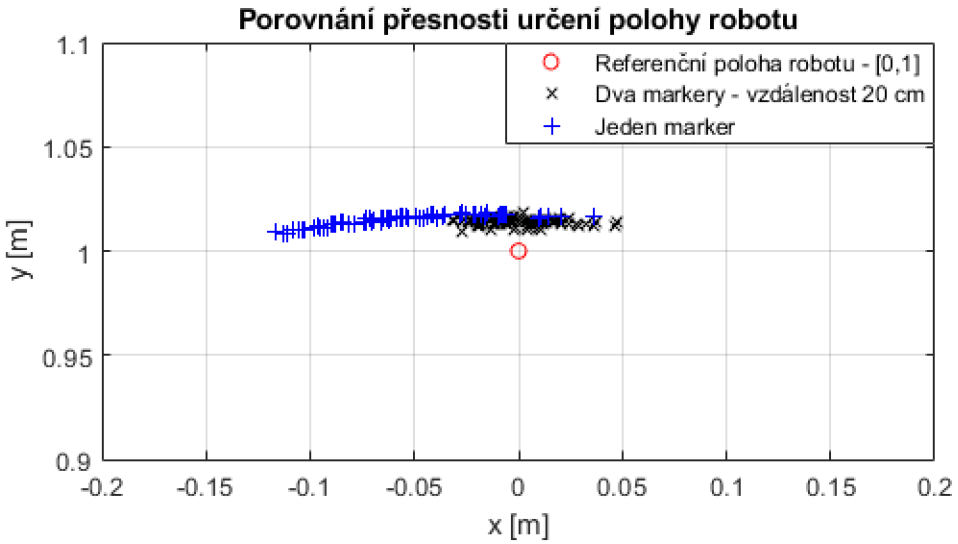


Obrázek 6.11: Výsledky měření dvou markerů.

6.3.2. Závěr z měření

Na základě výsledků bylo rozhodnuto, že budou použity 3 markery vedle sebe. Středový je určený pro navigaci v malé vzdálenosti od stanice a jako referenční marker při velké vzdálenosti. Boční markery slouží ke zpřesnění lokalizace, pokud je kamera vidí.

Celkově měření dokazuje, že použití markerů umožňuje dostatečně přesné určení relativní polohy mezi robotem a stanicí a může být přikročeno k samotné implementaci.

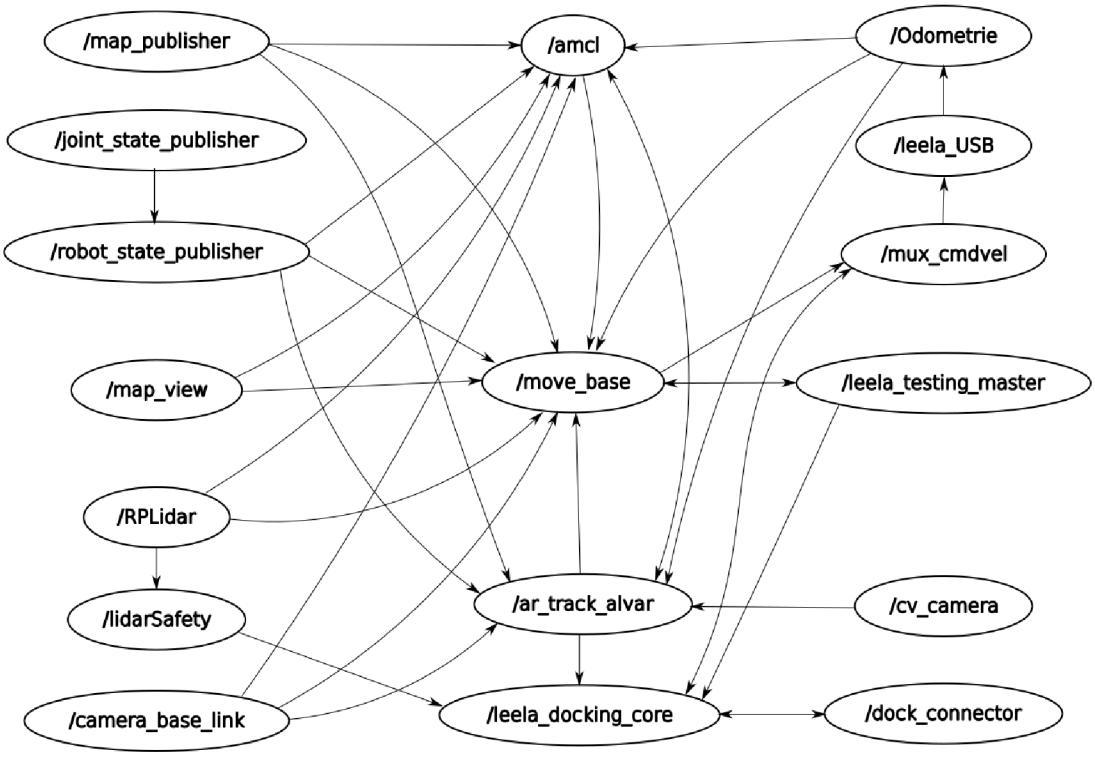


Obrázek 6.12: Porovnání přesnosti při použití jednoho / dvou markerů.

7. Implementace

7.1. Přehled systému

Jako celek se implementovaný systém neliší od navrženého konceptu. Z praktických důvodů je funkcionalita rozdělena na více skriptů, běžících paralelně. To umožňuje měnit jednotlivé části bez zásadního vlivu na celek, jde o modulární charakter, na kterém je ROS založen. Na Obr. 7.1 je přehled všech spuštěných modulů při navádění na stanici.



Obrázek 7.1: Aktivní moduly při dokování.

7.2. Zpracování obrazu, detekce markerů

7.2.1. Balíček cv_camera

Pro zobrazení dat z kamery je použit balíček cv_camera [44]. Po spuštění vytvoří topiky /cv_camera/image_raw s daty z kamery a /cv_camera/camera_info s parametry kamery, včetně kalibračních matic. Data o kameře jsou načtena ze souboru camera_info.yaml.

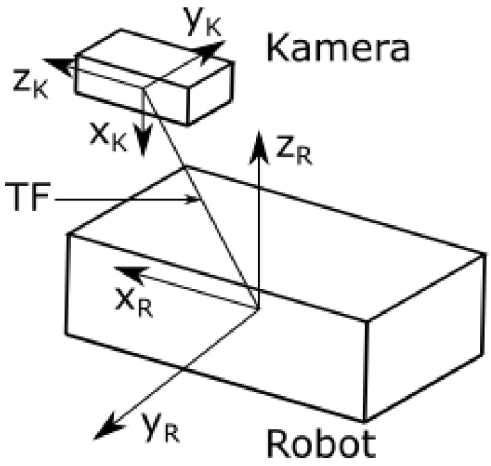
Kamera je připojena přes USB port, nicméně není třeba port definovat, balíček cv_camera je schopný kameru sám detekovat.

7.2.2. Balíček Alvar

Jedná se o samostatný balíček, který na základě obrazu detekuje markery. Hlavní funkcionalita je ve skriptu ”individualMarkersNoKinect”. Balíček obsahuje předpřipravený launch

soubor s ukázkou nastavení parametrů. Na jeho základě byl vytvořen vlastní kód pro launch soubor. Změněna je velikost použitých markerů. Také bylo nutné zadat správný název topiků s daty z kamery, protože si algoritmus sám provádí rektifikaci obrazu.

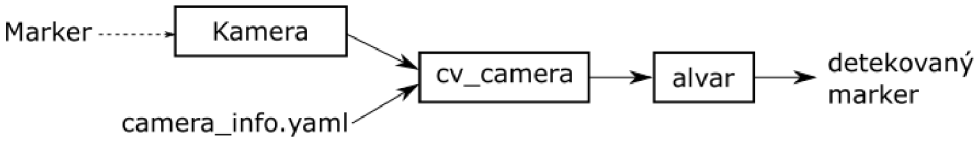
Souřadnice detekovaných markerů jsou v souřadném systému kamery, je tedy nutné i nastavit transformaci mezi kamerou a robotem. Balíček navíc používá nestandartní uspořádání os, jak je vidět na Obr. 7.2. Na Obr. 7.3 je pak schéma, jak na sebe balíčky alvar a cv_camera navazují.



Obrázek 7.2: Vztah mezi souřadnými systémy kamery a robotu.

Nastavené parametry:

- ”marker_size” default=”5” - Velikost markeru v cm
- ”max_new_marker_error” default=”0.08” – Převzatý parametr
- ”max_track_error” default=”0.2” – Převzatý parametr
- ”cam_image_topic” default=”/cv_camera/image_raw” – Zdroj obrazu
- ”cam_info_topic” default=”/cv_camera/camera_info” – Info o kameře
- ”output_frame” default=”/camera” – Název výstupního souřadného systému



Obrázek 7.3: Řetězec zpracování obrazu.

7.3. Dokovací balíček

Jelikož se vše tvořilo pro robot Leela, je balíček pojmenován leela_docking. Prakticky by měl kód fungovat i na robotu Breach. Pouhé jméno „docking“ by bylo příliš generické a mohlo by dojít ke kolizi s jiným kódem.

Balíček obsahuje funkcionalitu pro samotné dokování, stejně jako připravené launch soubory pro autonomní provoz robotu.

7.3.1. Zdrojové soubory

leela_docking_core.cpp

Skript zajišťuje hlavní funkcionalitu najíždění ke stanici a odjíždění. Po zapnutí se zpřístupní dokovací action server, který čeká na nový cíl.

Action server přijímá nový cíl pomocí zprávy ve formátu akce, který je definovaná souborem „leela_docking.action“ ve složce action. Akce se skládá ze tří částí – goal, result, feedback.

Goal – Cíl nebo akce, požadovaná klientem. Odesílá klient do serveru.

Result – Data, která klient dostane na konci akce. Odesílá server klientovi.

Feedback – Data, která jsou periodicky posílána klientovi během vykonávání akce. Slouží k informování o aktuálním stavu úlohy.

Každá část může obsahovat proměnné a struktury dle potřeby aplikace. Akce použitá pro dokování:

- Goal: docking_command – Číslo ve formátu int32, reprezentuje požadovanou akci. 0 – Zadokovat, 2 – Oddokovat
- Result: docking_result – Číslo ve formátu int32, reprezentuje výsledek akce. 0 – úkol splněn, nenulové číslo značí chybu
- Feedback: docking_state – Číslo ve formátu int32, indikuje jaká část úlohy je právě plněna

Nezávisle na běhu action serveru se zpracovávají data z detekce markeru. Při nové zprávě s markery se zkontroluje, jestli robot nevidí markery, které přísluší dokovací stanici. Pokud je vidí, uloží si je do globálních proměnných. Odebírány jsou i zprávy o stavu dokovacího konektoru a detekci překážek pomocí lidaru.

Po příchodu a vyhodnocení nového cíle začne skript pracovat v režimu stavového automatu. Každý stav odpovídá jednomu úkonu, jsou definovány následující stavy:

- SEARCHING_STATION
- DOCKING_STATION
- MATING
- UNDOCKING

SEARCHING_STATION je prvním stavem při úkolu dokování. Robot hledá stanici kontrolou přítomnosti markerů. Pokud je nalezen středový marker, je nalezena i stanice a automat přechází do stavu DOCKING_STATION.

Pokud není stanice detekována, pokusí se ji robot najít ve svém okolí. Robot začne na místě rotovat malou rychlostí a zastaví se při detekci středového markeru. Rotace je omezena na dvě celé otáčky kolem své osy a má také časový limit. Pokud není stanice nalezena, skript končí a vrací chybu. Pokud je lidarem detekována překážka, robot zastaví a čeká na odstranění překážky nebo do vypršení časového limitu.

Ve stavu DOCKING_STATION začne robot najíždět ke stanici s použitím zvoleného algoritmu. Více se naváděcími algoritmy zabývá část 7.3.3. Při bezchybné jízdě robot dojede do místa určeného ke spojení se stanicí a přejde do stavu MATING.

Najíždění na stanici je přerušeno, pokud robot zcela ztratí výhled na markery. Primárními příčinami může být špatné navedení najížděcím algoritmem nebo vnik překážky před stanici. Skript končí a hlásí chybu.

V případě detekce překážky lidarem se robot zastaví a čeká na odstranění. Je implementován časový limit, pokud vyprší, akce je přerušena.

Ve stavu MATING se robot snaží spojit se stanicí. Z předchozího najíždění by měl robot stát přímo ve směru konektoru. Robot zahájí jízdu dopředu, kterou končí detekcí spojení pomocí dokovací elektroniky. Z důvodu bezpečnosti je opět implementován časový limit.

Pokud je spojení úspěšné, akce končí, je nahlášeno úspěšné splnění klientovi a skript čeká na další úkol. V opačném případě robot přechází do stavu UNDOCKING a pokouší se o odjetí od stanice. Jízda od stanice je důležitá z toho důvodu, že při chybě v této fázi zůstane robot stát těsně u stanice a plánovač move_base není schopný navrhnout trajektorii bezpečného odjetí.

Pokud je nový cíl odjetí od stanice nebo nastane chyba při spojování, přejde automat do stavu UNDOCKING. Robot se začne pohybovat směrem od stanice, pohyb je ukončen při odjetí do dostatečné vzdálenosti, čímž končí i akce. V případě detekce překážky robot zastaví a čeká na její odstranění. Pohyb je omezen časovým limitem. Pokud limit vyprší, skript zahlásí chybu a akce se přeruší.

leela_dock.cpp

Skript se stará o dojetí robota ke stanici a zadokování. Nejprve je provedena kontrola vstupních parametrů, které definují pozici bodu před stanicí. Pokud není parametr k dispozici, použije se defaultní hodnota.

Pozice bodu před stanicí je přepočítána na formát vhodný pro move_base a začne čekání na action servery move_base a dokování. Poté se robot pokusí o jízdu ke stanici a zadokování, celkem je povoleno pět pokusů, než je úkol označen za nesplněný.

Prvním krokem je jízda ke stanici. Bod před stanicí je odeslán do move_base jako cíl a skript čeká na dokončení. Po úspěšném dojetí je odeslán příkaz k zadokování do dokovacího action serveru a opět se čeká na splnění úkolu.

Pokud v jakékoliv části algoritmus selže, provede vše od začátku jako další pokus. Pokud je dokování úspěšné, skript se ukončí.

Parametry:

- x – souřadnice dokovacího bodu před stanicí v ose x, typ double, jednotky jsou metry
- y – souřadnice dokovacího bodu před stanicí v ose y, typ double, jednotky jsou metry
- a – úhel značící směr z bodu před stanici ke stanici, typ double, jednotky jsou radiány

leela_undock.cpp

Jde o velmi jednoduchý skript, který do dokovacího modulu odešle příkaz k odjetí od stanice. Po odeslání skript čeká na dokončení. Předpokládá se, že v případě chyby je nutný zásah operátora, proto není pokus o odjetí opakován a skript se ukončí v případě úspěchu i chyby.

leela_docking_testing.cpp

Slouží k automatickému testování dokovací funkcionality. Skript nejprve čeká na spuštění move_base serveru a dokovacího modulu. Poté je funkce řízena stavovým automatem, který neustále opakuje těchto 5 příkazů:

1. Jízda do místa daleko od stanice.
2. Jízda ke stanici.
3. Spojení se stanicí.
4. Čekání (nabíjení).
5. Odjetí od stanice.

Pokud se v jakémkoliv místě kromě nabíjení stane chyba, začne sekvence od začátku, tedy jízdou mimo prostor stanice. Čekání/nabíjení je realizováno čistě jako časovač s pevně danou hodnotou. Jedná se o experimentální jízdu a je nutná přítomnost a pozornost obsluhy v každém okamžiku.

leela_docking_testing_triple.cpp

Jedná se o variaci předchozího skriptu leela_docking_testing.cpp. Jsou definovány celkem tři pozice daleko od stanice, do kterých může robot dojet. Cílový bod je vybírán náhodně, jeden z bodů je defaultní. Skript opět pracuje jako stavový automat, v případě chyby začne sekvence od začátku. Body jsou vybrány tak, aby při jízdě do bodu před stanicí musel robot dojet ze všech tří stran.

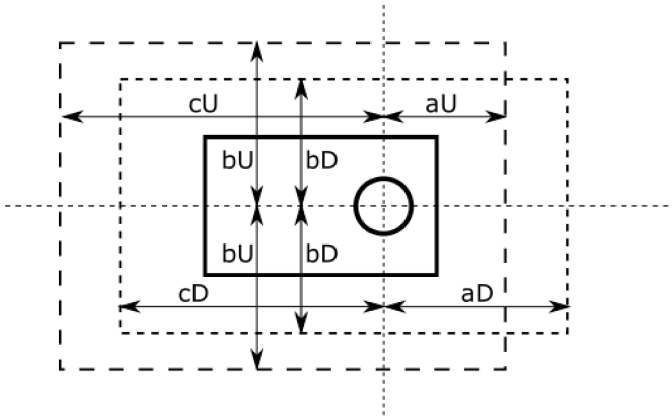
lidarSubscriber.cpp

Pohyb robota musí být bezpečný. Skript zajišťuje detekci překážek na základě dat z lidaru, aby při najíždění a odjíždění od stanice nedošlo ke kolizi.

Kolem robota jsou vymezeny dva virtuální prostory, pokud se jakýkoliv předmět do prostoru dostane a lidar ho zachytí, vyhodnotí to skript jako překážku. Prostory jsou definované jako obdélníky, jejichž strany jsou vzdálené od lidaru o určitou vzdálenost. Vzdálenosti jsou definované parametry ”a”, ”b” a ”c”, uspořádání je na Obr. 7.4.

Jelikož skript nerozlišuje směr k překážce, mohl by nastat problém při dokování. Malá hodnota parametru ”a” by nezabránila srážce při dokování, velká hodnota by nedovolila robotu opustit stanici při odjíždění. Proto jsou implementovány dva prostory, pro dokování jsou použity parametry ”aD”, ”bD”, ”cD” a pro oddokování parametry ”aU”, ”bU”, ”cU”. Všechny parametry lze nastavit pomocí launch souboru.

Samotný skript funguje tak, že při spuštění na základě parametrů spočítá limitní vzdálenosti pro 360 paprsků. Při běhu pak data z lidaru nejprve transformuje na 360 paprsků



Obrázek 7.4: Hlídané zóny při dokování.

a následně je porovnává se spočtenými limity. Výstup skriptu je do topiku /obstacle_detected, přehled možných hodnot je v tabulce 7.1.

Tabulka 7.1: Tabulka možných výstupů skriptu lidarSubscriber:

Hodnota	Význam
0	Defaultní hodnota, bez překážky
1	Překážka v dokovací oblasti
2	Překážka v oddokovací oblasti
3	Překážka v obou oblastech zároveň

dock_connector.py

Tento skript slouží ke komunikaci s mikroprocesorem dokovacího konektoru. Skript periodicky dotazuje mikroprocesor, který mu vrací aktuální stav detekce konektoru, napětí baterie a proudu do baterie. Pokud není třeba kontrolovat konektor, postačuje nízká frekvence dotazů. Naopak při najíždění na stanici je rychlost detekce konektoru zásadní. Proto je vstupem do skriptu status dokovací procedury, který mění frekvenci dotazů.

Při komunikaci je použit další python soubor, který obsahuje definice a funkce pro korektní komunikaci. Délka zpráv je pevná.

Formát dotazu počítače:

Byte	0	1	2	3
Význam	StartByte	Len	Cmd	CS

- StartByte – Pevně daný první byte zprávy s hodnotou 0x05
- Len – délka zbytku zprávy, zde Len = 2
- Cmd – Příkaz pro dokovací elektroniku. Při návrhu se počítalo s možností ovládat relé mezi konektorem a baterií, neimplementováno. Defaultní hodnota 0 značí žádost počítače o stav dokovací elektroniky a baterie.
- CS – Kontrolní součet, spočtený jako negace sumy ostatních bytů zprávy.

Formát odpovědi mikroprocesoru:

Byte	0	1	2	3	4	5	6	7
Význam	StartByte	Len	DockStatus	Uhigh	Ulow	Ihigh	Ilow	CS

- StartByte, Len a CS mají stejný význam jako u dotazu počítače.
- DockStatus – Byte, kde jednotlivé bity značí aktuální stav dokovací elektroniky. První dva bity jsou rezervovány pro stav relé. Třetí bit značí detekci zasunutí konektoru. Čtvrtý bit je nepoužitý. Zbývající bity jsou rezervovány pro chybové stavy. V této práci je implementovaná funkcionalita pouze pro bit detekce konektoru.
- Uhigh – Horní byte z 16bitového čísla reprezentujícího napětí baterie
- Ulow – Dolní byte z 16bitového čísla reprezentujícího napětí baterie
- Ihigh – Horní byte z 16bitového čísla reprezentujícího proud z baterie
- Ilow – Dolní byte z 16bitového čísla reprezentujícího proud z baterie

Čísla reprezentující napětí a proud odpovídají hodnotám čteným z analogového vstupu, jsou celočíselného typu a nabývají hodnot 0 až 1023. Skutečná hodnota proudu závisí na použitém referenčním napětí a použitém děliči napětí. Hodnota reprezentující proud závisí na použitém proudovém čidle a jeho rozsahu.

fake_marker.cpp

Simulátor Stage nepodporuje simulaci kamery a následnou detekci markerů. Proto je vytvořen skript, který detekci napodobuje. Pozice a natočení všech markerů jsou předem dané, z modulu amcl lze získat pozici robota. Nejprve je kontrolováno, jestli je robot v pracovní oblasti markerů a má šanci je vidět. Poté se kontroluje, jestli je robot správně natočený a má markery v záběru. Pokud ano, jsou spočítány souřadnice, jak by je určila detekce Alvar. Na konci jsou viditelné markery publikovány pro RVIZ a také je stvořena zpráva, imitující výstup z algoritmu Alvar.

7.3.2. Launch soubory

leela_docking_301.launch

Spojuje veškeré potřebné skripty a uzly kromě vyššího řízení. Navazuje na původní launch soubor pro autonomní pohyb, navíc zajišťuje dokovací funkcionalitu. Musí být spuštěn na robotu. Konkrétně jsou parametry nastaveny pro místnost 301, předpokládá se spuštění, když je robot ve stanici. Při zavolání se spouští tyto uzly:

- Leela_docking_core – Dokovací funkcionalita
- lidarSafety – Vyhodnocení překážek
- dock_connector – Komunikace s mikroprocesorem konektoru
- mux_cmdvel – Mux pro zprávy pohybu

- joint_state_publisher – Transformace kloubů robotu
- robot_state_publisher – Transformace mezi částmi robotu
- leela_USB – Komunikace s driverem motorů
- Odometrie – Přepočet dat z enkodéru na odometrii
- RPLidar – Lidar
- map_view – Map server, poskytuje data o mapě
- map_publisher – Poskytuje transformace vůči mapě
- amcl – Adaptivní monte carlo lokalizace
- move_base – Plánovač
- cv_camera – Poskytuje data z kamery
- camera_base_link – Transformace mezi kamerou a robotem
- ar_track_alvar – Detekce AR markeru z obrazu

Cmd_vel mux

Příkazy pro pohyb jsou standardně odesílány do topiku /cmd_vel. V případě dvou zdrojů pohybu by došlo ke kolizi, proto je pro přepínání mezi algoritmy použit modul mux, kterým lze dynamicky přepínat zdroj zpráv. Defaultní stav je takový, že z pohledu celku není třeba nic měnit a uživatel může bez problémů posílat vlastní příkazy, například pomocí gamepadu. Změna nastane pouze při rotaci před stanicí nebo najíždění na ni a po dokončení je vstup automaticky vrácen do původní hodnoty. Volání modulu mux je možné pomocí rosservice jak z příkazové řádky, tak z kódu.

leela_Dock.launch

Slouží k iniciaci jízdy ke stanici a dokování. Spouští se soubor leela_dock.cpp. Defaultní hodnoty souřadnic stanice jsou nastavené z testování, při změně stačí v launch souboru změnit parametry souřadnic bodu před stanicí. Příklad nastavení souřadnic vypadá následovně:

- <param name=”x” type=”double” value=”0.66”/> <!– Souřadnice x –>
- <param name=”y” type=”double” value=”-0.69”/> <!– Souřadnice y –>
- <param name=”a” type=”double” value=”3.14”/> <!– Úhel od osy x –>

leela_Undock.launch

Slouží k iniciaci odjetí od stanice. Spouští se soubor leela_undock.cpp. Skript nemá žádné parametry, použití launch souboru je čistě pro udržení konzistentnosti.

leela_Docking_testing.launch

Slouží ke spuštění testovacích skriptů. Na výběr jsou dva soubory:

- leela_docking_testing.cpp – Testování s odjetím jen do jednoho bodu
- leela_docking_testin_triple.cpp – Testování s náhodným odjetím do jednoho ze tří bodů

Změna souboru je provedena pouhým přepsáním typu volaného uzlu.

leela_docking_simulation.launch

Slouží ke kompletní simulaci robotu v počítači. Kombinuje funkcionalitu launch souborů leela_docking a leela_docking_testing, navíc jsou nahrazeny všechny uzly, které komunikují s hardwarem robotu. Spouští se tyto uzly:

- Stage – Simulátor robotu a prostředí
- leela_docking_testing – Vyšší vrstva řízení celého systému
- rviz – Vizualizace dat
- fake_marker – Nahrazuje kameru a detekci markerů
- Leela_docking_core – Dokovací funkcionalita
- mux_cmdvel – Mux pro zprávy pohybu
- joint_state_publisher – Transformace kloubů robotu
- robot_state_publisher – Transformace mezi částmi robotu
- map_view – Map server, poskytuje data o mapě
- map_publisher – Poskytuje transformace vůči mapě
- amcl – Adaptivní monte carlo lokalizace
- move_base – Plánovač
- camera_base_link – Transformace mezi kamerou a robotem

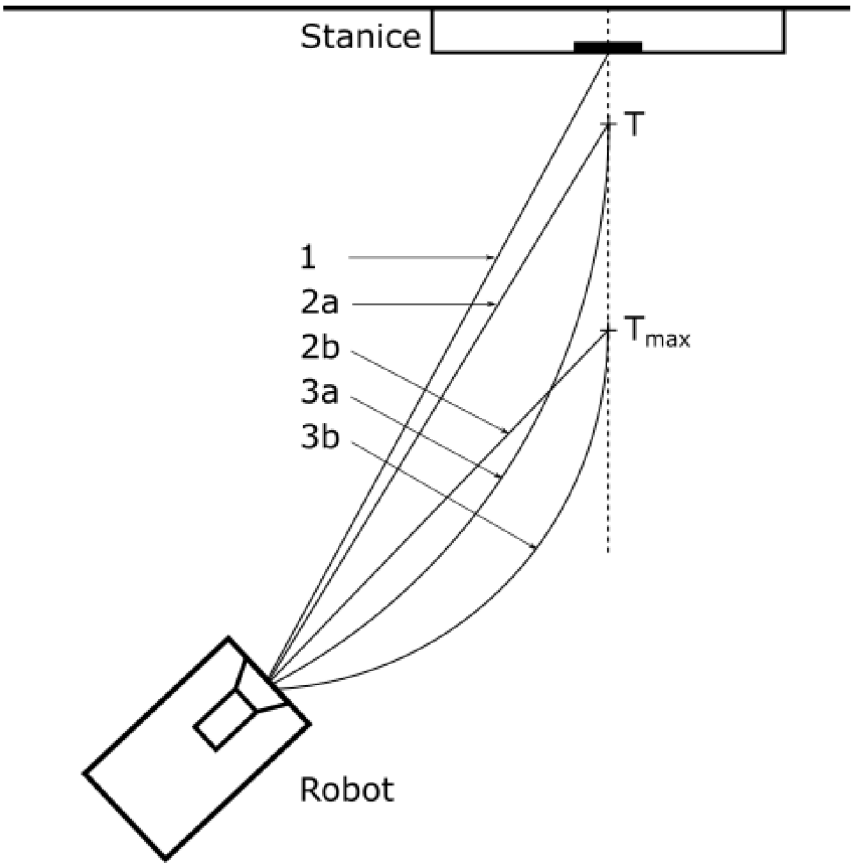
Také je nastaven parametr „use_sim_time” na „true“, aby ROS používal čas simulace.

7.3.3. Naváděcí algoritmy

Samotný pohyb robotu je řešen dvěma způsoby. První z nich je pomocí původního plánovače move_base a lokalizace amcl. Tato metoda je použita pro jízdu ze vzdáleného bodu do přibližné polohy stanice. Předpokládalo se, že robot dojede na určené místo před stanicí s přesností 5 až 10 cm a konečnou úhlovou chybou menší než 5°. Tím je určena kruhová oblast, odkud musí robot dojet do stanice. Toto řešení obsahuje veškeré bezpečnostní prvky a metody záchrany, během pohybu je robot schopný celou trasu přeplánovat či objíždět překážky.

Po dojetí robotu ke stanici pak přebírá řízení dokovací algoritmus. Řešeny jsou dvě úlohy, a to nalezení stanice a následné najetí. Pokud je systém správně nakonfigurovaný, dojede robot čelem ke stanici. Pokud by se tak nestalo a kamera stanici nevidí, začne se robot otáčet na místě a stanici hledat. Tento pohyb je realizován jednoduchým příkazem k rotaci.

Po lokalizaci stanice by měl robot začít najíždět ke konektoru. Existuje mnoho způsobů jak naplánovat cestu. V této práci bylo prozkoumáno několik variant. Grafický přehled je na Obr. 7.5.



Obrázek 7.5: Navigační algoritmy.

Varianta 1

Jedná se o jednoduché najíždění po přímce na prostřední marker. Ostatní markery nejsou potřeba. Algoritmus vyžaduje, aby robot začínal před stanicí, jinak hrozí nebezpečí najetí pod nežádoucím úhlem.

Varianta 2a

Před stanicí je vytvořen imaginární bod T (target) s předem definovanou vzdáleností od stanice. Robot se pak snaží po přímce k tomuto bodu dojet. Po najetí dostatečně blízko se robot přepne do režimu pohybu s variantou 1, jelikož boční markery už téměř nejsou v záběru a robot se nachází přímo před stanicí.

Varianta 2b

Oproti variantě 2a není vzdálenost bodu T od stanice pevně stanovena. Na základě aktuální polohy je spočítán bod T tak, aby byl v co největší vzdálenosti od stanice, ale zároveň na něj byl robot schopný najíždět a kamera pořád viděla středový marker. Tímto způsobem pak robot má více prostoru pro následné natočení ke středovému markeru a srovnání při najíždění.

Varianta 3a

Tato možnost vychází z varianty 2a, ale při najíždění na bod T jde o pohyb po kružnici. Záměr je takový, že pokud je kružnice v bodě T tečná k normále stanice, tak na konci pohybu nemusí robot provádět rotaci na místě a může rovnou najet ke stanici.

Varianta 3b

Zde je opět rozvedena myšlenka, že je bod T v co největší vzdálenosti od stanice. Robot pak má víc prostoru pro korekci přímého najíždění.

Všechny varianty byly implementovány a vyzkoušeny, v konečném kódu zůstala pouze preferovaná varianta 2a.

7.3.4. Regulace pohybu

Veškerý pohyb je řízen pomocí příkazů ve formátu geometry_msgs/Twist do topiku /cmd_vel. Tento topik je odebírán skriptem, který posílá zprávy do low-level řízení motorů pomocí sériové linky. Skript navíc obsahuje jednoduchý PI regulátor, také low-level má vlastní regulaci.

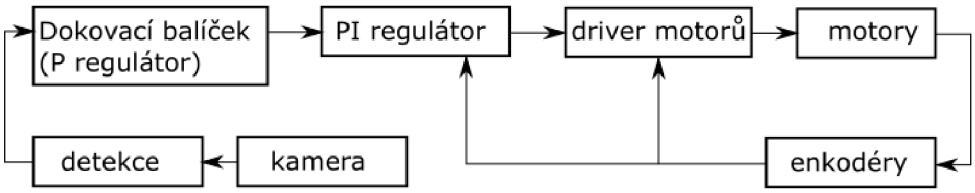
Při autonomním pohybu po mapě je pohyb řízen modulem move_base.

Při použití vlastního plánování, ať jen pro rotaci, či najíždění, je pohyb řízen tak, že dopředná rychlost je konstantní a úhlová rychlost je regulována na základě úhlové odchylky robotu od požadované polohy.

Pro potřeby dokování stačí jednoduchý P regulátor, jelikož regulace úhlové rychlosti již I složku obsahuje. Pro bezpečnost jsou všechny tyto regulace doplněny o podmínku sledující výstup skriptu, který kontroluje lidar data. Při najíždění na stanici se nepředpokládaly dynamické nebo jiné překážky, prostor by měl být volný. V případě detekce

překážky, například průchod člověka, se robot zastaví a čeká na uvolnění prostoru. Pokud by v určitém čase nedošlo k uvolnění, zahlásil by robot chybu a přerušil celý proces.

Smyčka vyšší vrstvy řízení vyhodnocuje pohyb s frekvencí 10 Hz. Vzhledem k tomu, že nové údaje o pozicích markerů přichází s frekvencí asi 8 Hz, je řídící frekvence dostatečně rychlá, což potvrzují i reálné testy. Přehled regulace během dokování je na Obr. 7.6.



Obrázek 7.6: Regulační řetězec.

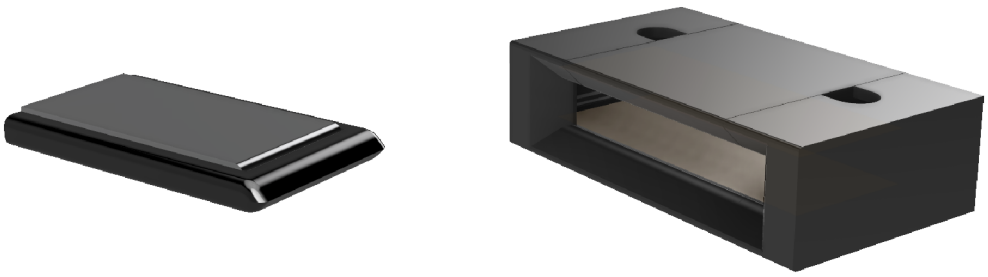
7.3.5. Kompilace

Vzhledem k modularitě systému ROS není důležité, kde je kód zkompilovaný a spuštěný. Nicméně bylo třeba kompilovat jak v robotu Leela, tak i na počítači. Na obou zařízeních se vyskytly různé problémy, například chyběly různé knihovny potřebné k běhu.

Vyvinutý balíček leela_docking je zde prezentovaný jako celek, nicméně v praxi existují dvě verze. První z nich je v robotu a obsahuje již odladěný a zcela funkční kód. Druhá verze je v počítači a obsahuje i pomocné skripty pro vývoj, testování či simulace.

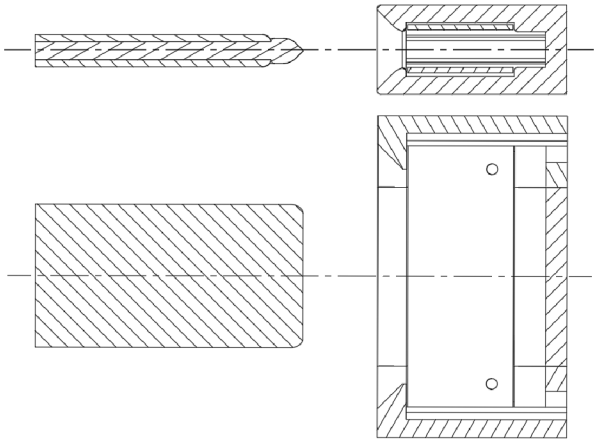
7.4. Detekce stanice a nabíjecí konektor

Samotná navigace pomocí markeru mohla být pro robot Leela dostatečná, nicméně pro zvýšení bezpečnosti byl na základě návrhu vytvořen dokovací konektor a obslužná elektronika. Tímto způsobem může být otestována kompletní zamýšlená funkcionalita autonomního nabíjení.



Obrázek 7.7: Modely konektorů.

Základ tvoří konektor s dvěma plochami. Robot nemá odpružení kol a dá se tedy předpokládat, že se nebude měnit jeho výška. Proto může být použito uspořádání na Obr. 7.7, na Obr. 7.8 jsou pak řezy konektory. Široký konektor umožňuje určitou chybu při najíždění, přechodový odpor je relativně malý a kontakt tak může bez problémů přenášet požadovaný proud. Pro jednoduchost konektor nemá komunikační linky.



Obrázek 7.8: Řez konektorem.

Jedná se o experimentální uspořádání, je nutná odborná obsluha. Baterie je chráněna pojistkou.

Systém obsahuje mikroprocesor, který komunikuje se základovou deskou pomocí sériové komunikace. Čip kontroluje vstupy a hodnoty pak předává počítači. Kontrolován je stav optické brány, přidáno bylo také měření napětí a proudu baterie pro diagnostické účely. Napětí je nejdříve sníženo odporovým děličem a následně měřeno pomocí vestavěného A/D převodníku. Proud je měřen pomocí čidla, připojeného do série s baterií, výstup čidla je opět analogová hodnota, měřená mikročipem.

Stav brány je posílán do topiku /dock_endstop, data o baterii do topiků /battery_voltage a /battery_current.

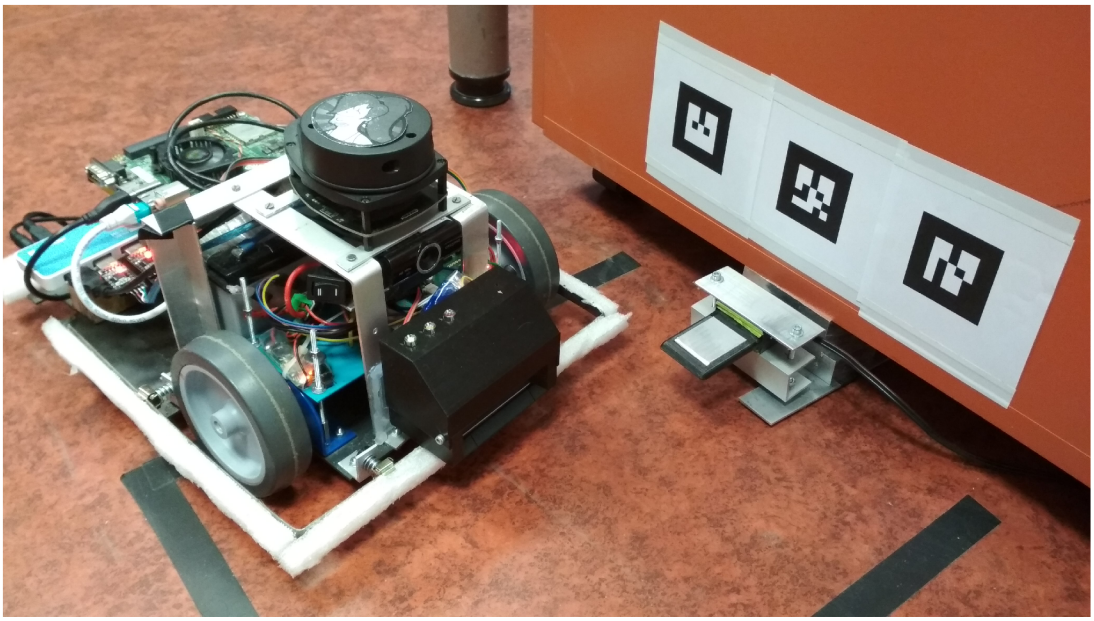
Pro signalizaci stavu jsou k dispozici tři led diody. Zelená dioda, napájená pouze přes odpor, indikuje přítomnost napájecího napětí pro mikroprocesor. Žlutá dioda indikuje spojení se stanicí. Červená dioda je vyhrazena pro indikaci poruchy. Žlutá i červená dioda je řízena tranzistorem, aby šlo stav spínat mikroprocesorem. Realizovaný HW je na Obr. 7.9.

7.5. Změny v operačním systému robotu

Lidar, driver a následně i dokovací elektronika používají stejný převodník na USB. Jde o modul s čipem CP2102. Z výroby mají všechny tyto převodníky stejné sériové číslo a nelze je rozlišit. Po připojení do USB portu se vytváří virtuální COM porty, které jsou automaticky číslovány. V praxi to znamená, že pokud má určitý skript pevně uložené jméno periferie, se kterou má komunikovat, musí se periferie připojovat v předem stanoveném pořadí, protože na základě převodníku nelze rozlišit, o jakou periferii se jedná.

Aby se předešlo případné kolizi, lze definovat tzv. udev pravidla. Vždy po připojení nového zařízení jsou na něj pravidla aplikována. Na základě jedinečných parametrů lze každému zařízení přiřadit symbolické jméno, které je vždy stejné a unikátní.

Čipy CP2102 lze identifikovat podle jedinečné kombinace parametrů idVendor a idProduct. Pomocí utility CP21xxCustomization_Utility lze sériové číslo změnit na unikátní, což umožní i rozlišení konkrétního čipu.



Obrázek 7.9: Robot Leela vedle nabíjecí stanice.

Pravidla jsou umístěna ve složce /etc/udev/rules.d, soubory jsou vyhodnocovány postupně na základě jména. Pro robot Leela byl vytvořen soubor „20-leela_USB.rules“, který USB periferiím přiřazuje symbolická jména, přehled je v tabulce 7.2. Ve skriptech pak lze nová jména použít stejným způsobem jako původní automatické jméno přiřazené systémem.

Tabulka 7.2: Tabulka symbolických adres USB převodníků

Zařízení	CP2102 Sériové číslo	Symbolické jméno
RPLidar	„BR_RPLIDAR_01“	„usbLidar“
Motor driver	„BR_MDRIVER_01“	„usbLeela“
Dokovací elektronika	„BR_DOCK_01“	„usbDock“
Čip CP2102 – idVendor = „10c4“, idProduct = „ea60“		

8. Verifikace

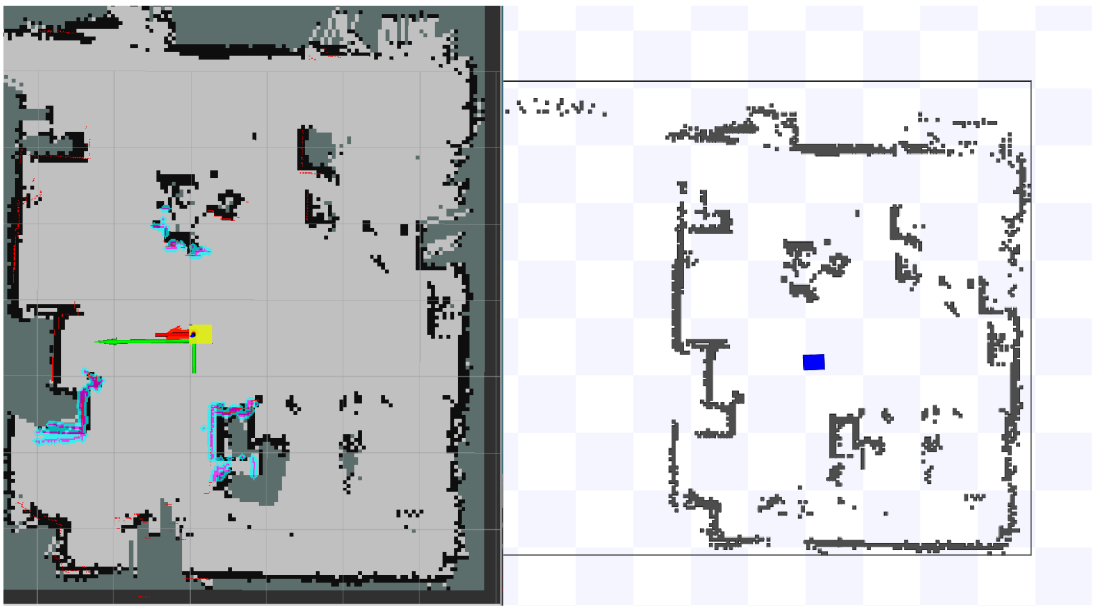
Ještě před prvním zapnutím robota je vhodné systém otestovat v simulaci. V případě robotu Leela to není vyloženě nutné, kdyby robot havaroval, tak by se nic nestalo, protože má slabé motory a nízkou hmotnost. Nicméně jde o práci s HW, jednoduchá simulace šetří čas, určitou dobu není nutná manipulace s robotem či jeho nabíjení a velmi snadno lze odladit systémové chyby.

Od určité fáze byl systém testován přímo na robotu, stejně tak byla provedena i konečná verifikace.

8.1. Simulace

Pro simulaci pohybu robotu ve známém prostředí je v ROSu k dispozici modul Stage[45]. Jedná se o jednoduchý 2D simulátor. Po inicializaci představuje reálný hardware. Má vlastní mapu, robot simuluje na základě urdf souboru. Vstupem jsou příkazy pro pohyb v topiku /cmd_vel, výstupem je odometrie. Také lze simulovat celou řadu senzorů, v tomto případě to je lidar.

Prvním krokem bylo simulovat původní funkcionalitu autonomní navigace. Na Obr. 8.1 je vidět porovnání mezi simulátorem a zobrazením v RVIZu.

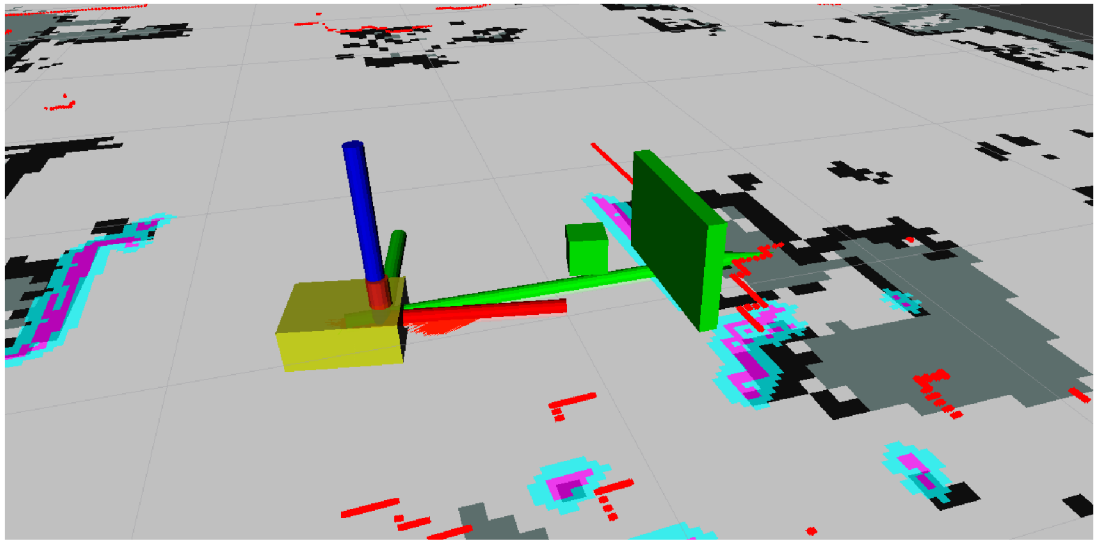


Obrázek 8.1: Ukázka RVIZ (vlevo) a Stage (vpravo).

Druhým krokem bylo simulovat výstup detekce markerů. Stage takové uspořádání nepodporuje, bylo tedy nutné napsat vlastní simulaci. Vytvořený skript na základě odebírané polohy počítá viditelnost markeru, případně vrací simulovanou polohu. Funkce byla ověřena pro všechny 3 markery. Pro jednoduchost simulace vrací přesné hodnoty pozice markeru, chyby nejsou uvažovány.

Poté začala implementace samotné funkcionality. Simulace se osvědčila a předešla mnoha chybám, následný přechod na reálný hardware byl téměř bezproblémový. Na

Obr. 8.2 je vidět běh simulace s najížděním na stanici, zeleně je vyznačen bod T, na který robot právě najíždí.



Obrázek 8.2: Robot v simulaci najíždí ke stanici.

Do simulace není zahrnuta detekce připojení ke stanici. Naopak je testováno chování, když se robot se stanicí nespojí a musí pokus přerušit.

V rámci simulace byl vytvořen řídící skript, který neustále opakuje těchto 5 příkazů:

1. Jízda do místa daleko od stanice.
2. Jízda ke stanici.
3. Spojení se stanicí.
4. Čekání (nabíjení).
5. Odjetí od stanice.

Při ladění není nutné každý příkaz ručně psát, což šetří mnoho času. Stejný skript je pak využit s reálným robotem, v konečné fázi se jednalo o funkcionalitu launch souboru „leela_docking_simulation.launch“.

8.2. Testování

Při implementaci byl systém souběžně se simulací často testován na reálném robotu. Použit byl výše zmíněný skript. Největší rozdíl mezi simulací a realitou byl při ladění algoritmů v najíždění na stanici. Zatímco simulace dává přesné hodnoty a lze odladit základní funkcionalitu, v realitě jsou některé algoritmy velmi nestabilní. To je způsobeno nepřesností v určení pozice markeru. Závěry z testování jednotlivých variant pohybu jsou rozebrány v následujícím textu.

8.2.1. Move_base

Pohyb pomocí move_base a amcl je stabilní. Lze nastavit požadovanou přesnost konečné polohy. S vysokou přesností má robot problém, protože není stavěn na velmi malé rychlosti a často překmitne. Oproti původnímu stavu se požadovaná přesnost pro potřeby dokování neměnila, ve většině případů robot dojede s chybou polohy do 5 cm od požadovaného místa.

8.2.2. Rotace na místě

Při pouhé rotaci o maximální úhlové rychlosti stanovené v části 6.2.3 se robot otáčí trhaně, nicméně dokáže marker detekovat. Se stávajícím hardwarem není lepší regulace dosažitelná.

8.2.3. Najíždění na stanici

Všechny algoritmy navržené v kap. 7.3.3 byly testovány na robotu. Oproti simulaci se silně projevují nepřesnosti měření, chyby regulace a jiné vlivy.

Verze 1: Algoritmus není složitý, pokud robot najede přesně před stanici, tak bez problémů i zadokuje. Problém nastane, když je robot mimo normálu stanice, vždy najede pod úhlem a od určité vzdálenosti není ani schopen trefit konektor. Algoritmus je ideální pro použití s dalším algoritmem, který robot dopraví na normálu stanice.

Verze 2a: Algoritmus funguje dobře, je relativně jednoduchý a robustní. Koncový úhel od kolmice není vždy nulový, což může být způsobeno nepřesnostmi hardwaru.

Verze 2b: Algoritmus funguje, ale není stabilní. Výpočet je náročnější než u 2a, vzniká větší chyba lokalizace. Jsou nutné přídavné podmínky a funguje stejně nebo hůř jako 2a.

Verze 3a: Algoritmus funguje správně v méně než polovině případů. Pokud funguje, dojede robot ke stanici úplně rovně. V opačném případě docházelo k vybočení z trajektorie a přerušení úkolu. Pro robot Leela není algoritmus vhodný kvůli chybám regulace motorů, pro robot Breach by algoritmus mohl fungovat.

Verze 3b: Algoritmus je silně nestabilní, správně funguje jen ve čtvrtině případů. Výpočet je složitý, nestabilní a nemá reálnou výhodu oproti ostatním verzím. Kvůli nutnosti velmi přesné a pomalé jízdy není algoritmus vhodný pro ani jeden robot.

Na Obr. 8.3 je vidět příklad navigace ke stanici, algoritmus verze 2a. Pro názornost je bod začátku pohybu ke stanici záměrně posunutý vůči normále stanice.

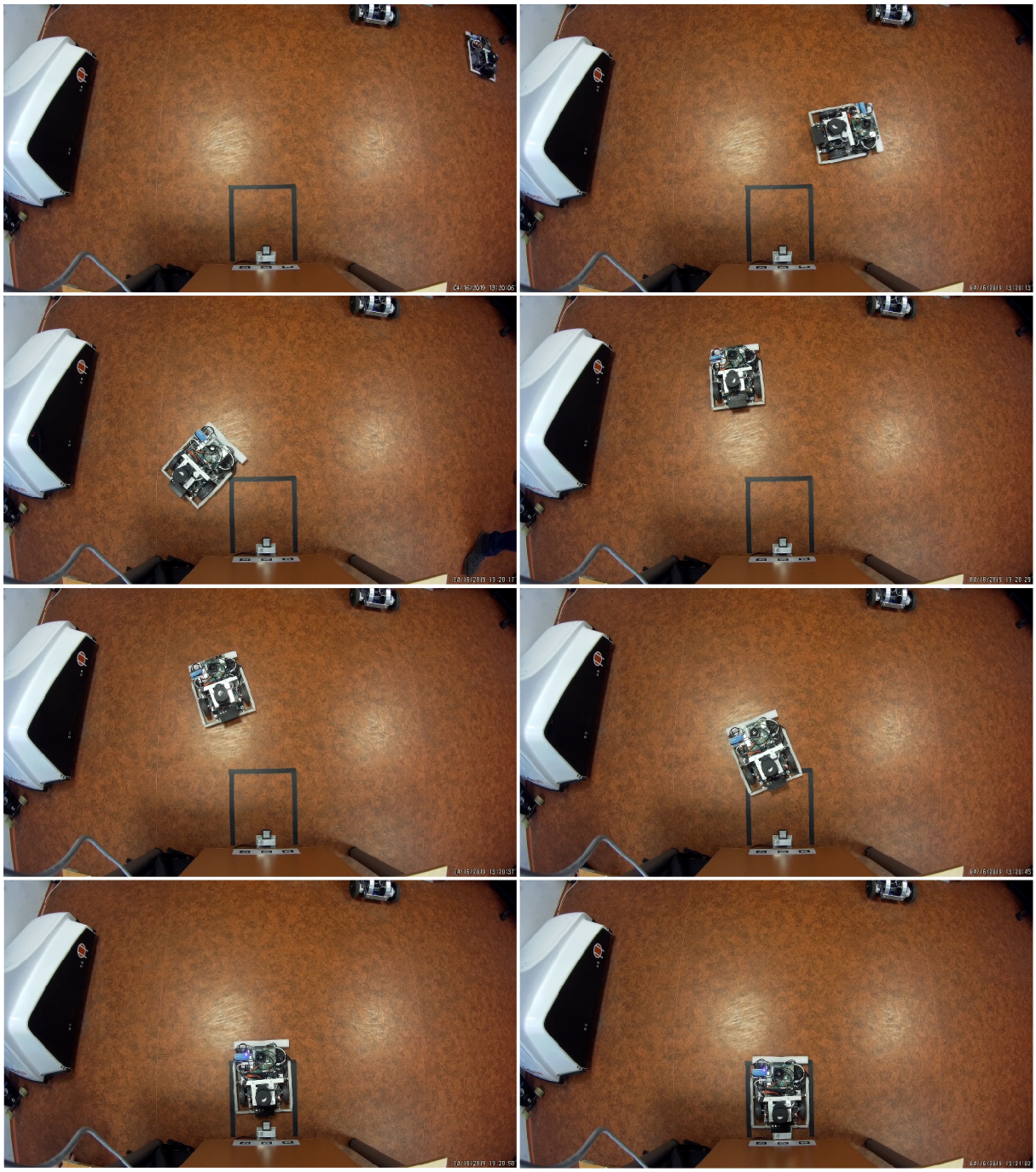
Hodně velký obrázek

8.2.4. Dlouhodobé testy

Kromě verifikace probíhaly dlouhodobé testy i během vývoje. Tím bylo možné odladit chyby, které neodhalila simulace.

8.2.5. Testy bezpečnosti

Zvlášť byla testována i bezpečnost zajišťovaná lidarem. Robot je schopný zastavit před člověkem i překážkou v dostatečné vzdálenosti.



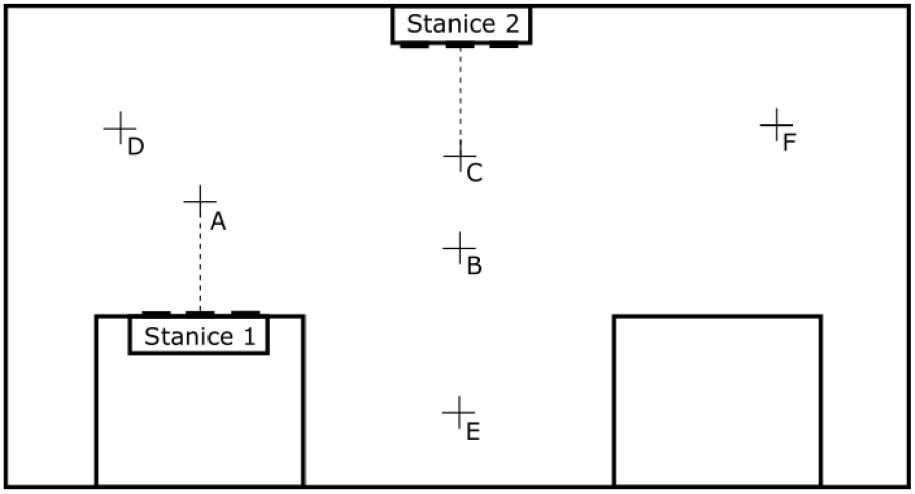
Obrázek 8.3: Ukázka celého procesu dokování.

8.3. Verifikace celku

Závěr této práce se věnuje verifikaci funkčnosti systému. Proběhly dvě sady testů po 50 pokusech. Jako jeden pokus je myšleno to, že robot začal v poloze daleko od stanice, musel ke stanici dojet a správně zadokovat, bez možnosti opravy jednotlivých kroků.

První sada pokusů probíhala stejně jako dosavadní testování, uspořádání bylo stejné jako na Obr. 8.3, rozložení v místnosti je schematicky naznačeno na Obr. 8.4, A je bod

před stanicí Stanice 1, B je vzdálený bod. Toto uspořádání má nevýhodu v tom, že robot najíždí vždy ze stejného místa. Výsledky jsou v tabulce 8.1 viz. Testovací jízdy 1.



Obrázek 8.4: Rozmístění cílů při testování.

Tabulka 8.1: Verifikační testy

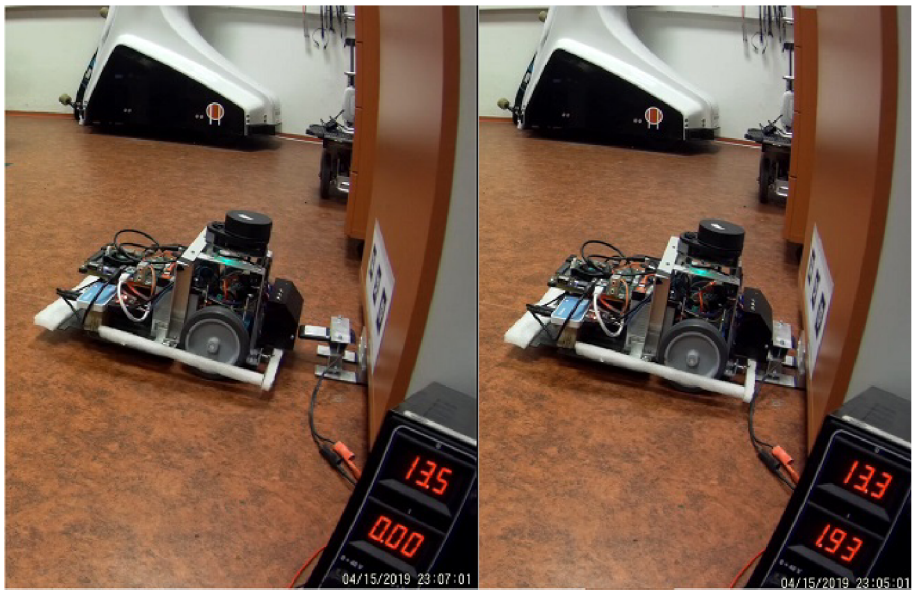
	Počet pokusů	Úspěšných	Neúspěšných
Testovací jízdy 1	50	48	2
Testovací jízdy 2	50	47	3
Celkem	100	95	5

Poté byla stanice přesunuta doprostřed místnosti do polohy Stanice 2 s bodem C před stanicí. Robot pak mohl najíždět ze všech stran z bodů D, E, F. Přesun stanice sloužil i k otestování funkčnosti parametrů. Celý proces přesunu stanice a konfigurace nových bodů zabral méně než 15 minut. Výsledky jsou v tab. 8.1 viz. Testovací jízdy 2.

Dohromady mělo testování trvat několik hodin, a proto bylo zapojeno provizorní nabíjení. V obou případech byl ke konektoru připojen laboratorní zdroj a po připojení robot vždy 90 sekund čekal, aby se baterie mírně dobila, poté se odpojil. Verifikace byla nahrávána, výřez s pohledem na zdroj je na Obr. 8.5. S nabíjením stoupla celková doba verifikace na asi 5 hodin, nicméně po skončení obou jízd byla úroveň nabití baterie stejná jako na začátku, což de facto potvrzuje funkčnost systému.

Jednou chyba nastala při najíždění ke stanici, kdy regulace rychlosti způsobila nadměrné překmitnutí a robot ztratil stanici z dohledu. V jednom případě robot správně zadokoval, ale chybně vyhodnotil úspěšné spojení se stanicí. Jedenkrát navigace move_base špatně vyhodnotila vzdálenost od překážky a robot se zasekl při jízdě ke stanici. Poslední dvě chyby způsobil plánovač move_base, který z neznámých důvodů odmítl naplánovat jakoukoliv trasu, robot zůstal stát uprostřed místnosti a bylo nutné ho manuálně postrčit.

Z analýzy chyb vyplývá, že pouze ve dvou případech selhal dokovací balíček, zbylé tři chyby způsobil plánovač move_base. Chybu najíždění by vyřešily silnější motory, chyba detekce stanice se dala očekávat, protože implementace konektoru je provizorní. Chyby plánování by měly jít odstranit vhodnou změnou parametrů move_base.



Obrázek 8.5: Ukázka reálného nabíjení.

9. Závěr

Cílem této diplomové práce bylo navrhnout a implementovat systém autonomního dokování pro mobilní robot. Po úvodní rešerši již realizovaných řešení a algoritmů byl s ohledem na požadavky navržen systém dokování. Předpokládalo se použití robotu s diferenciálním podvozkem, který je řízen frameworkem ROS. Pro potřeby lokalizace robotu vůči nabíjecí stanici bylo zvoleno použití detekce vizuálních markerů.

Validita metody lokalizace pomocí markerů byla ověřena testovacím měřením. Nejdříve byla vybrána kamera, metoda detekce a vhodný marker. Zkoumány byly rozsahy a limity metody detekce, absolutní přesnost i uživatelská přívětivost. Na základě výsledků byla navržena úprava rozložení markeru, místo jednoho bylo zkoumáno použití více markerů zároveň. Nová konfigurace byla otestována a porovnána s původní. Výsledné zlepšení je zásadní, a proto byla nová varianta ponechána.

Pro implementaci byl zvolen experimentální robot Leela. Původní senzorická výbava zahrnovala pouze lidar a enkodéry motorů, robot byl schopný autonomní jízdy ve zmapovaném prostředí. Senzorika byla rozšířena o kameru a dokovací elekroniku. Pro potřeby práce byla vytvořena jednoduchá nabíjecí stanice, ke které se robot může připojit. Na původní funkcionalitu bylo navázáno, systém byl doplněn o dokovací balíček a podpůrné skripty.

Postupná implementace byla testována pomocí simulátoru Stage, který nahrazuje hardware robotu. V pozdějších fázích se přešlo na testování s reálným robotem. Celkovou funkcionalitu ověřily verifikační testy, systém dosáhl úspěšnosti 95 % při úloze dokování. Chyby byly analyzovány a byla navržena zlepšení.

Dokovací funkcionalitu může operátor manuálně spustit pomocí předpřipravených launch souborů, balíček je zároveň vhodný pro začlenění do systému s vyšším řízením, čehož bylo využito při autonomním testování.

Dokovací elektronika byla nutná pro detekci spojení se stanicí. Nad rámec práce byl vytvořen funkční konektor mezi stanicí a robotem, který umožňuje nabíjení robotu externím zdrojem. Možnosti nabíjení bylo využito během verifikačních testů. Dále bylo implementováno měření napětí a proudu baterie pro diagnostické účely.

Vytvořený systém není navázaný na konkrétní činnost robot, dalším krokem by mělo být ověření funkčnosti v reálné aplikaci. Také by bylo vhodné vytvořit plnohodnotný ROS balíček včetně dokumentace. Pro budoucí použití je žádoucí umožnit konfiguraci parametrů pomocí launch souborů, v současnosti je implementována podpora pouze pro nejdůležitější parametry.

Podařilo se vytvořit kompletní systém autonomního dokování včetně jednoduchého nabíjení, všechny cíle diplomové práce byly splněny.

Literatura

[1] SONG G., H. WANG, J. ZHANG, T. MENG. Automatic Docking System for Recharging Home Surveillance Robots. *Ieee Transactions On Consumer Electronics* [online]. IEEE-INST ELECTRICAL ELECTRONICS ENGINEERS, 2011, 57(2), 428-435 [cit. 2018-09-06]. ISSN 0098-3063.

[2] CASSINIS, R., F. TAMPALINI, P. BARTOLINI, R. FEDRIGOTTI. *Docking and charging system for autonomous mobile robots.* [online]. 2005. [cit. 2018-09-06]. Dostupné z: https://www.researchgate.net/publication/228800043_Docking_and_charging_system_for_autonomous_mobile_robots

[3] SILVERMAN M., D. NIES, B. JUNG, G. SUKHATME. *Staying alive: a docking station for autonomous robot recharging.* In: Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292) [online]. IEEE, 2002, 1, 1050-1055 vol.1 [cit. 2018-09-06]. DOI: 10.1109/ROBOT.2002.1013494. ISBN 0780372727.

[4] QUILEZ Roberto, Adriaan ZEEMAN, Nathalie MITTON, Julien VANDAELE. *Docking autonomous robots in passive docks withInfrared sensors and QR codes.* In: International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCOM) [online]. 2015 [cit. 2018-09-06].

[5] ACOSTA CALDERON A. C., Buck Sin NG, E. R. MOHAN, Heng Khai NG. *Docking System and Power Management for Autonomous Mobile Robots.* In: Applied Mechanics and Materials [online]. Trans Tech Publications, 2014, 590(Innovative Solutions in the Field of Engineering Sciences), s. 407-412 [cit. 2018-09-06]. DOI: 10.4028/www.scientific.net/AMM.590.407. ISSN 1660-9336.

[6] GUANGRUI Fan, Wang GENG. *Vision-based autonomous docking and re-charging system for mobile robot in warehouse environment.* In: 2017 2nd International Conference on Robotics and Automation Engineering (ICRAE) [online]. IEEE, 2017, s. 79-83 [cit. 2018-09-06]. DOI: 10.1109/ICRAE.2017.8291357.

[7] AMARASINGHE D., G. MANN, R. GOSINE. *Vision-based hybrid control strategy for autonomous docking of a mobile robot.* In: Proceedings of 2005 IEEE Conference on Control Applications, 2005. CCA 2005 [online]. IEEE, 2005, s. 1600-1605 [cit. 2018-09-06]. DOI: 10.1109/CCA.2005.1507361. ISBN 0780393546.

[8] WANG Wei, Zongliang LI, Wenpeng YU, Jianwei ZHANG. *An autonomous docking method based on ultrasonic sensors for self-reconfigurable mobile robot.* In: 2009 IEEE International Conference on Robotics and Biomimetics (ROBIO) [online]. IEEE, 2009, s. 1744-1749 [cit. 2018-09-06]. DOI: 10.1109/ROBIO.2009.5420436. ISBN 9781424447749.

[9] LUO R., C. LIAO, K. SU, K. LIN. *Automatic docking and recharging system for autonomous security robot.* In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems [online]. IEEE, 2005, s. 2953-2958 [cit. 2018-09-06]. DOI: 10.1109/IROS.2005.1545197. ISBN 0780389123.

[10] TAKEUCHI E., T. TSUBOUCHI. *Portable effector docking mechanism for a service mobile robot and its positioning.* In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006 [online]. IEEE, 2006, s. 3380-3386 [cit. 2018-09-06]. DOI: 10.1109/ROBOT.2006.1642218. ISBN 0780395050.

[11] KARTOUN U., H. STERN, Y. EDAN, C. FEIED, J. HANDLER, M. SMITH, M. GILLAM. *Vision-Based Autonomous Robot Self-Docking and Recharging.* In: 2006 World Automation Congress [online]. IEEE, 2006, s. 1-8 [cit. 2018-09-06]. DOI: 10.1109/WAC.2006.375987. ISBN 1889335339.

[12] KIM, Myungsik. Automated Robot Docking Using Direction Sensing RFID. In: Proceedings 2007 IEEE International Conference on Robotics and Automation [online]. IEEE, 2007, s. 4588-4593 [cit. 2018-09-06]. DOI: 10.1109/ROBOT.2007.364186. ISBN 1424406013.

[13] WU Yi-cheng, Ming-chang TENG, Yi-jeng TSAI. *Robot docking station for automatic battery exchanging and charging.* In: 2008 IEEE International Conference on Robotics and Biomimetics [online]. IEEE, 2009, s. 1043-1046 [cit. 2018-09-06]. DOI: 10.1109/ROBIO.2009.4913144. ISBN 9781424426782.

[14] *ROS.org | Powering the world's robots.* [online]. Open Source Robotics Foundation. [cit. 2019-05-22]. Dostupné z: http://www.ros.org/.

[15] ROS.org *ROS Wiki: Distributions* [online]. Open Source Robotics Foundation, poslední aktualizace 10. 5. 2019 [cit. 2019-05-11]. Dostupné z: http://wiki.ros.org/Distributions

[16] THRUN Sebastian, Wolfram BURGARD, Dieter FOX. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series). Intelligent robotics and autonomous agents. The MIT Press, August 2005.

[17] ROS.org *ROS Wiki: amcl* [online]. Open Source Robotics Foundation, poslední aktualizace 24. 9. 2018 [cit. 2019-03-6]. Dostupné z: http://wiki.ros.org/amcl

[18] ROS.org *ROS Wiki: move_base* [online]. Open Source Robotics Foundation, poslední aktualizace 27. 9. 2018 [cit. 2019-03-6]. Dostupné z: http://wiki.ros.org/move_base

[19] ROS.org *ROS Wiki: actionlib* [online]. Open Source Robotics Foundation, poslední aktualizace 30. 10. 2018 [cit. 2019-03-6]. Dostupné z: http://wiki.ros.org/actionlib

[20] HABERMEHL Daniel, Katrin HENKNER, Swantje ECKER, Oliver JÄKEL, Jürgen DEBUS, Stephanie E. COMBS. *Evaluation of different fiducial markers for image-guided radiotherapy and particle therapy.* Journal of Radiation Research [online]. Oxford University Press, 2013, 54(suppl1), i61-i68 [cit. 2019-02-11]. DOI: 10.1093/jrr/rrt071. ISSN 0449-3060.

[21] ZHANG Xiang, S. FRONZ, N. NAVAB. *Visual marker detection and decoding in AR systems: a comparative study.* In: Proceedings. International Symposium on Mixed and Augmented Reality [online]. IEEE, 2002, s. 97-106 [cit. 2019-02-11]. DOI: 10.1109/ISMAR.2002.1115078. ISBN 0769517811.

[22] ROS.org *ROS Wiki: ar_tools* [online]. Open Source Robotics Foundation, poslední aktualizace 16. 5. 2015 [cit. 2019-03-6]. Dostupné z: http://wiki.ros.org/ar_tools?distro=indigo

[23] ROS.org *ROS Wiki: apriltag_ros* [online]. Open Source Robotics Foundation, poslední aktualizace 22. 4. 2019 [cit. 2019-05-3]. Dostupné z: http://wiki.ros.org/apriltag_ros

[24] ROS.org *ROS Wiki: ar_track_alvar* [online]. Open Source Robotics Foundation, poslední aktualizace 19. 7. 2016 [cit. 2019-03-6]. Dostupné z: http://wiki.ros.org/ar_track_alvar

[25] ROS.org *ROS Wiki: aruco* [online]. Open Source Robotics Foundation, poslední aktualizace 10. 9. 2015 [cit. 2019-03-6]. Dostupné z: http://wiki.ros.org/aruco

[26] ROS.org *ROS Wiki: fiducials* [online]. Open Source Robotics Foundation, poslední aktualizace 27. 8. 2018 [cit. 2019-03-6]. Dostupné z: http://wiki.ros.org/fiducials

[27] Human Interface Technology Laboratory. *ARToolKit* [online]. ARToolworks, Inc., Seattle, WA, USA. [cit. 2019-02-11]. Dostupné z: http://www.hitl.washington.edu/artoolkit/

[28] FIALA M. *ARToolkit Applied to Panoramic Vision for Robot Navigation.* [online] 2003. [cit. 2019-02-11]. Dostupné z: https://www.researchgate.net/publication/44079412_ARToolkit_Applied_to_Panoramic_Vision_for_Robotic_Navigation

[29] FIALA M. *Comparing ARTag and ARToolkit Plus fiducial marker systems.* In: IEEE International Workshop on Haptic Audio Visual Environments and their Applications [online]. IEEE, 2005, 2005, 6 pp. [cit. 2019-02-11]. DOI: 10.1109/HAVE.2005.1545669. ISBN 0780393767.

[30] OLSON E. *AprilTag: A robust and flexible visual fiducial system* E.In: Proceedings - IEEE International Conference on Robotics and Automation [online]. 2011, s. 3400-3407 [cit. 2019-02-11]. DOI: 10.1109/ICRA.2011.5979561. ISBN 9781612843865. ISSN 10504729.

[31] WANG J., E. OLSON. *AprilTag 2: Efficient and robust fiducial detection.* In: IEEE International Conference on Intelligent Robots and Systems [online]. Institute of Electrical and Electronics Engineers, 2016, 2016-, s. 4193-4198 [cit. 2019-02-11]. DOI: 10.1109/IROS.2016.7759617. ISBN 9781509037629. ISSN 21530858.

[32] APRIL Laboratory. *AprilTag* [online]. The APRIL Robotics Laboratory at the University of Michigan, ©2010. [cit. 2019-02-11]. Dostupné z: https://april.eecs.umich.edu/

[33] RICHARDSON A., J. STROM, E. OLSON. *AprilCal: Assisted and repeatable camera calibration.* In: IEEE International Conference on Intelligent Robots and Systems [online]. 2013, s. 1814-1821 [cit. 2019-05-22]. DOI: 10.1109/IROS.2013.6696595. ISBN 9781467363587. ISSN 21530858.

[34] Augmented Reality / 3D Tracking. *ALVAR* [online]. VTT Technical Research Centre of Finland Ltd., ©2000-2019. [cit. 2019-02-11]. Dostupné z: http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/index.html

[35] ROMERO-RAMIREZ Francisco J., Rafael MUÑOZ-SALINAS, Rafael MEDINA-CARNICER. *Speeded up detection of squared fiducial markers.* Image and Vision Computing [online]. Elsevier B.V, 2018, 76, 38-47 [cit. 2019-02-11]. DOI: 10.1016/j.imavis.2018.05.004. ISSN 0262-8856.

[36] GARRIDO-JURADO S., R. MUÑOZ-SALINAS, F. J. MADRID-CUEVAS, R. MEDINA-CARNICER. *Generation of fiducial marker dictionaries using Mixed Integer Linear Programming.* Pattern Recognition [online]. Elsevier, 2016, 51, 481-491 [cit. 2019-02-11]. DOI: 10.1016/j.patcog.2015.09.023. ISSN 0031-3203.

[37] Aplicaciones de la Visión Artificial. *ArUco: a minimal library for Augmented Reality applications based on OpenCV* [online]. [cit. 2019-02-11]. Dostupné z: https://www.uco.es/investiga/grupos/ava/node/26

[38] DEGOL Joseph, Timothy BRETL, Derek HOIEM. *ChromaTag: A Colored Marker and Fast Detection Algorithm.* ArXiv.org [online]. Ithaca: Cornell University Library, arXiv.org, 2017 [cit. 2019-02-11]. Dostupné z: http://search.proquest.com/docview/2075825571/

[39] SAGITOV Artur, Ksenia SHABALINA, Roman LAVRENOV, Evgeni MAGID. *Comparing fiducial marker systems in the presence of occlusion.* In: 2017 International Conference on Mechanical, System and Control Engineering (ICMSC) [online]. IEEE, 2017, s. 377-382 [cit. 2019-02-11]. DOI: 10.1109/ICMSC.2017.7959505.

[40] KÄLBERER Kai, Melvin KLEIN, Christina PAULE. *Perception packages for Robotics - Analysis and Evaluation* [online]. University of Stuttgart, Institute of Parallel and Distributed Systems, Machine Learning and Robotics Lab, 2015 [cit. 2019-02-11]. Dostupné z: https://ipvs.informatik.uni-stuttgart.de/mlr/papers/15-kaelberer_klein_paule-Fachstudie.pdf

[41] BABINEC Andrej, Ladislav JURIŠICA, Peter HUBINSKÝ, František DUCHOŇ. *Visual Localization of Mobile Robot Using Artificial Markers.* Procedia Engineering [online]. Elsevier, 2014, 96(C), 1-9 [cit. 2019-02-11]. DOI: 10.1016/j.proeng.2014.12.091. ISSN 1877-7058.

[42] Bender Robotics. *Mobile robotic platform Breach* [online]. [cit. 2019-02-11]. Dostupné z: http://www.benderrobotics.com/breach.html

[43] OpenCV. *Camera Calibration and 3D Reconstruction* [online]. ©2011-2014 [cit. 2019-02-11]. Dostupné z: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

[44] ROS.org *ROS Wiki: cv_camera* [online]. Open Source Robotics Foundation, poslední aktualizace 22. 6. 2017 [cit. 2019-03-6]. Dostupné z: http://wiki.ros.org/cv_camera

[45] ROS.org *ROS Wiki: stage* [online]. Open Source Robotics Foundation, poslední aktualizace 18. 9. 2018 [cit. 2019-03-6]. Dostupné z: http://wiki.ros.org/stage

10. Seznam příloh

1. Příloha 1 Obsah CD

Příloha 1 - Obsah CD

Přiložené CD obsahuje elektronickou verzi bakalářské práce, zdrojové soubory a skripty.
Struktura adresářů:

```
/ ..... kořenový adresář přiloženého CD
└── zdrojove soubory ..... zdrojové soubory
    ├── Leela ..... zdrojové soubory pro robot Leela
    └── PC ..... zdrojové soubory pro obslužné PC
```