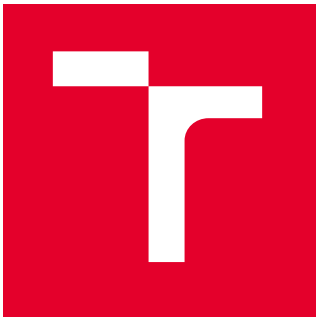


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

VYUŽITÍ HYBRIDNÍCH MAP V ÚLOZE LOKALIZACE A MAPOVÁNÍ MOBILNÍCH ROBOTŮ

USING HYBRID MAPS IN MOBILE ROBOT LOCALIZATION AND MAPPING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Oscar Fajgel**
AUTHOR

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

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Oscar Fajgel**
Studijní program:	Mechatronika
Studijní obor:	bez specializace
Vedoucí práce:	**doc. Ing. Jiří Krejsa, Ph.D.**
Akademický rok:	2023/24

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 bakalářské práce:

Využití hybridních map v úloze lokalizace a mapování mobilních robotů

Stručná charakteristika problematiky úkolu:

V mobilní robotice ve vnitřním prostředí se používají jak 2D tak 3D reprezentace okolního prostoru a to ve vektorové i rastrové podobě. Nevýhodou 2D reprezentace je mimo jiné obtížné použití u vícepodlažních budov, 3D reprezentace zase klade vysoké nároky na datovou propustnost a obecně na výkon palubního počítače. Podobný problém se vyskytuje i u 2D map, pokud zahrnují velký prostor. Podstatou bakalářské práce je navrhnout mechanismus spojení několika lokálních map, který by zachoval výhody a eliminoval nevýhody jednotlivých přístupů.

Cíle bakalářské práce:

1. Proveďte rešerši možných způsobů reprezentace map v mobilní robotice pro vnitřní prostředí
2. Navrhněte mechanismus spojení několika lokálních map pro optimální reprezentaci okolí robotu
3. Navržený mechanismus implementujte v prostředí ROS a simulačně ověřte

Seznam doporučené literatury:

YAMANAKA S, MARIOKA K: Mobile robot navigation using hybrid simplified map with relationships between places and grid maps, IFAC, vol 45, Issue 22, pp 616-621, Elsevier, 2012

HAA W et al: Spatial semantic hybrid map building and application of mobile service robot, Robotics and Autonomous Systems, vol 61, Issue 6, pp 923-941, Elsevier, 2014

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2023/24

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

Abstrakt

Bakalárska práca sa zaoberá návrhom hybridnej mapy pre účely navigácie a lokalizácie mobilného robota vo vnútorných priestoroch viacposchodovej budovy. Vykonaná je rešerš ohľadom aktuálne používaných spôsobov mapovania v oblasti mobilnej robotiky, na základe ktorej sú navrhnuté samostatné ciele tvorby hybridnej mapy. Preskúmané sú viaceré metódy, z ktorých je vybraná najvhodnejšia a následne je otestovaná simulačne v prostredí ROS. Na otestovanie spoľahlivosti metódy sú vykonané aj testy na reálnom robotovi.

Summary

This bachelor's thesis deals with analysis of current approaches to mapping used in mobile robotics and explores a hybrid mapping approach that could be used in mobile robot navigation and localization in multi-level indoor environments of buildings. Multiple methods are examined and a suitable approach is tested using ROS framework simulations. A real life test on a real robot is also concluded to test the usability of this approach.

Kľúčové slová

mobilný robot, mapovanie, viacposchodová vnútorná navigácia, ROS

Keywords

mobile robot, mapping, multi-level indoor navigation, ROS

Bibliografická citácia

FAJGEL, Oscar. *Využití hybridních map v úloze lokalizace a mapování mobilních robotů.* Brno, 2024. Dostupné také z: https://www.vut.cz/studenti/zav-prace/detail/157445. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce: Jiří Krejsa.

Prehlasujem, že som bakalársku prácu vypracoval samostatne s využitím jednotlivých zdrojov uvedených v zozname na konci tejto práce.

Oscar Fajgel

Brno . . . .

Ďakujem vedúcemu práce doc. Ing. Jiřímu Krejsovi Ph.D za cenné rady a postrehy počas vypracovania bakalárskej práce. Taktiež ďakujem Ing. Miroslavovi Čeplovi a jeho kolegom z firmy Bender Robotics za pomoc pri spojazdňovaní simulačného prostredia a robota. Ďakujem svojej rodine a blízkym za podporu a za príležitosť venovať sa tejto práci.

Oscar Fajgel

Obsah

1 Úvod

V súčasnosti je v otázke modernizácie priemyslu snaha zakomponovať mobilné roboty do jeho širokého spektra odvetví. Tieto roboty postupne začínajú nachádzať uplatnenie v rozličných úlohách distribúcie v skladoch a na výrobných linkách, v otázkach dopravy tovaru, ale aj pre vojenské účely. Cieľom je uľahčiť prácu, ktorá je pre ľudí fyzicky a psychicky namáhavá.

Bez ohľadu na to, v akom prostredí je mobilný robot nasadený, hlavnou požiadavkou pri jeho návrhu je, aby bol schopný efektívne vykonať svoje zadanie. V mnohých prípadoch vychádza schopnosť vykonania týchto zadaní z predpokladu, že robot je schopný navigácie daným prostredím a lokalizácie v ňom. Pre obe úlohy je charakteristická práve potreba vhodnej reprezentácie prostredia.

Podľa druhu prostredia sú kladené jednotlivé podmienky pri návrhu jeho reprezentácie. Z úvah ohľadom ideálnej implementácie vychádzajú prevedenia v podobe samostatných máp, ktoré popisujú dané prostredie v požadovanej miere. Môže ísť napríklad o mapu jednej z časti skladov alebo o mapu poschodia viacposchodovej budovy.

Vytvorené mapy bývajú často vo forme 2D, ale existujú aj 3D prístupy k tvorbe máp. Ak je zadaná požiadavka na reprezentáciu celej viacposchodovej budovy prostredníctvom máp, otázka ohľadom správnej implementácie začína byť nejasná. Použitie 2D reprezentácie na viacposchodovú budovu je obtiažne, zatiaľ čo 3D reprezentácia kladie väčšie nároky na dostupnú pamäť a výkon riadiaceho systému robota. Tieto nároky sú pri oboch prístupoch o to väčšie, čím sú väčšie mapované priestory a počet poschodí danej budovy.

Cieľom bakalárskej práce je návrh vhodného spôsobu spojenia viacerých lokálnych máp do jednej hybridnej mapy, ktorá bude vhodná pre nasadenie v úlohách navigácie a lokalizácie pozemného mobilného robota vo vnútorných priestoroch viacposchodovej budovy. Navrhnutý spôsob je následne nutné simulačne overiť.

Pred návrhom hybridnej mapy bola vykonaná rešerš súčasných, najčastejšie používaných reprezentácií prostredí v podobe máp pre účely navigácie mobilných robotov, ako aj rešerš úzkej oblasti lokalizácie. Pre simulačné účely bola spravená rešerš vhodného simulačného softvéru, potrebných balíkov pre prácu s ním a balíkov potrebných pre navigáciu simulačného modelu mobilného robota, ktoré sú dostupné v rámci prostredia ROS - Robot Operating System.

Pri návrhu sa vychádzalo z kódu prevzatého z projektu zalievacieho robota Zaleela. Daný kód bol upravený a obohatený o potrebné prvky.

2 Rešerš

Pred samotným návrhom hybridnej mapy bolo potrebné preskúmať základnú problematiku ohľadom navigácie a lokalizácie mobilných robotov, ako aj súčasné prístupy k mapovaniu a tiež základné koncepčné náležitosti prostredia Robot Operating System. Táto kapitola sa zaoberá danou problematikou.

2.1 Navigácia a lokalizácia

So slovom navigácia prichádza ľudstvo do kontaktu už od čias moreplavby a objavovania svetadielov. Pojem reprezentuje optimálne riadenie polohy a rýchlosti objektu pri jeho pohybe z bodu A do bodu B.

Problematika navigácie predstavuje komplexný študijný odbor, ktorý je možné rozdeliť na pododbory aj podľa toho, v akom prostredí objekt navigujeme. Menovitými príkladmi je hlavne pozemná, vzdušná, podmorská či vesmírna navigácia.

V oblasti robotiky je najbežnejším príkladom navigácie práve pozemná navigácia mobilného robota. Aj napriek jednoduchej predstave častokrát nejde o jednoduchú činnosť. Navigovať človeka, ktorý dokáže racionálne premýšľať a spájať veci do súvislostí, je zvyčajne jednoduchšie ako navigácia mobilného robota.

Činnosť navigácie je možné rozdeliť na postupnosť troch základných krokov: [1]

- **Vnímanie**
  Ide o získavanie informácií zo svojho okolia pomocou snímačov, ktorými je robot vybavený. Snahou je umožniť robotovi získať čo najviac informácií o svojom prostredí vo vhodnej podobe. Nie každá informácia môže byť pre robot užitočná. Pri návrhu senzorického rozhrania robota je nutné zamyslieť sa nad tým, aké všetky druhy informácií je možné z prostredia získať a vybrať z nich tie, ktoré sú pre činnosť robota najpodstatnejšie. V opačnom prípade môže dôjsť k zahlteniu systému dátami a k nesprávnemu chovaniu robota, ak je jeho riadenie nevhodne navrhnuté.

- **Lokalizácia**
  Lokalizácia spočíva v spracovaní získaných dát o prostredí vhodnými metódami. Výstupom je hrubý odhad pozície, v ktorej sa robot práve nachádza. Tento odhad môžeme spresniť použitím viacerých metód lokalizácie naraz, napr. pridaním modulu na lokalizáciu pomocou GPS a i. Rozličné metódy lokalizácie ponúkajú rôznu presnosť odhadu polohy, čo predstavuje dôležité kritérium pri návrhu navigácie. [2]

- **Plánovanie**
  Plánovanie trasy spočíva v rozhodovaní systému robota o nasledujúcej akcii, ktorej vykonaním sa má robot priblížiť k svojmu cieľu. V praxi býva plánovanie často rozdelené na globálne a lokálne. Globálne plánovanie predstavuje hrubý odhad potrebnej trasy, zatiaľ čo lokálne sa zoberá plánovaním len najbližšieho počtu krokov.

Robot je takto z globálneho hľadiska navigovaný k svojmu cieľu aj napriek možným zmenám trasy na povel lokálneho plánovača, napríklad kvôli vyhýbaniu sa detegovanej prekážke na ceste. [3]

Problémy lokalizácie

Problematika lokalizácie nemusí byť pre užívateľa spočiatku jednoznačná. Jej jednotlivé prevedenia sa môžu drasticky líšiť podľa prostredia, v ktorom ju má robot vykonávať, a aj podľa nároku na jej presnosť.

Pri návrhu lokalizačného prístupu je potrebné brať do úvahy aj chyby vychádzajúce zo samotných vlastností jednotlivých častí hardvéru robota. Menovite ide hlavne o: [1]

- **Šum senzorov**

  Každý zo snímačov ma svoju snímaciu časť vyrobenú s určitou presnosťou garantovanou výrobcom, ktorá má svoju citlivosť na snímanú veličinu. Pri väčšej citlivosti môže výstup kmitať okolo určitej hodnoty, čo sa prejaví v podobe šumu.

  Operačná činnosť snímača môže byť taktiež ovplyvnená vonkajšími vplyvmi. Napríklad surové dáta z infračerveného snímača budú ovplyvnené infračerveným žiarením zo Slnka, prípadne iných zdrojov. Premenlivosť v meraniach vzdialenosti laserovým snímačom bude spôsobená rozdielmi vonkajšieho vzhľadu snímaného objektu - jeho farba, drsnosť, uhol natočenia, a pod.

- **Nepresnosti end-efektorov**

  End-efektor je časť robota pomocou ktorej interaguje so svojim prostredím. Pri navigácii mobilného robota ide hlavne o kolesá, v súvislosti s ktorými dochádza k chybám v odometrii. Motory sú vybavené enkodérmi, ktoré umožňujú riadiacemu systému robota prepočítať prejdenú vzdialenosť pomocou rotácie kolies. Chyba tohto prepočtu bude narastať so zväčšujúcou sa prejdenou dráhou. Ide o takzvaný „drift“, ktorý je spôsobený hlavne nerovnosťami povrchov pri styku s kolesami, chybami prepočtov a nepresnosťami z výroby.

  Spomenutý jav je možné vidieť na Obrázku č. 2.1.

Obrázok č. 2.1: Jav „drift“ - nezhoda medzi mapou a detegovanou hranicou priestoru

Implementácia lokalizácie

Pred návrhom samotného navigačného systému je vhodné si položiť otázku, či sa dané pracovné využitie robota dokáže zaobísť bez implementácie lokalizácie. Ak je pracovné nasadenie vhodné, jeho navigácia môže byť riešená „primitívnymi“ metódami.

Príkladom riešenia je sada inštrukcií, aby robot nasledoval určitý prvok vo svojom prostredí - namaľovaná čiara alebo nasprejované identifikačné body, a pohyboval sa pozdĺž nich. Následne je jeho konečný cieľ možné realizovať ľahko snímateľným objektom, napríklad feromagnetom. [1]

Tento prístup je vhodný iba za predpokladu, že podmienky prostredia sú dostatočne vhodné a robot je vybavený adekvátnym hardvérom. Proces implementácie môže byť časovo náročný, v niektorých prostrediach nemožný, a použiteľný iba pre dané nasadenie.

V moderných aplikáciách je preferovaným prístupom pri návrhu navigácie mobilného robota zakomponovanie lokalizačných algoritmov, ktoré sú robustné a umožňujú väčšiu modularitu použitia.

Táto motivácia vychádza z hlavného faktu, že je prakticky nemožné, aby robot pomocou jeho vnútornej logiky a jednej sady vyhodnotených dát dospel práve k jedinému odhadu jeho aktuálnej pozície, keďže neistota tohto odhadu je obrovská. [1]

Únikom z danej situácie je zavedenie pravdepodobnostnej logiky vedúcej k viacerým hypotézam, ohľadom teoreticky nekonečného množstva pozícií, v ktorých sa robot môže práve nachádzať. Niektoré pozície sú viac pravdepodobné než iné.

Dve hlavné pravdepodobnostné prístupy k tomuto druhu lokalizácie sú Markovova lokalizácia a Kalmanova lokalizácia. V praxi bežne využívané lokalizačné algoritmy, ktoré vzišli z týchto dvoch techník, sú založené na báze metódy SLAM - Simultaneous Localization and Mapping. [1]

Robot spočiatku nemá žiadnu informáciu o tom, v akej pozícii sa práve nachádza, a teda pravdepodobnosti pre všetky body mapy sú v tomto čase rovnaké. Jeho postupným pohybom v navigovanom priestore sa na základe detegovaných unikátnych prvkov prostredia upravujú pravdepodobnostné hodnoty bodov. Robot má postupom času lepší odhad, kde sa práve nachádza.

2.2 Reprezentácia prostredia

Prostredie, v ktorom sa má robot pohybovať je často komplikované - je potrebné ho do určitej miery zjednodušiť a vhodne reprezentovať. Najčastejší spôsob reprezentácie prostredia je prostredníctvom mapy, ktorá vhodne popisuje závislosti medzi objektmi v priestore.

Pri návrhu mapy je potrebné brať ohľad na vybavenie robota a zvoliť optimálne rozlíšenie mapy podľa jeho špecifikácií. Zároveň je vhodné, aby bola primerane zjednodušená kvôli úspore dostupnej pamäte.

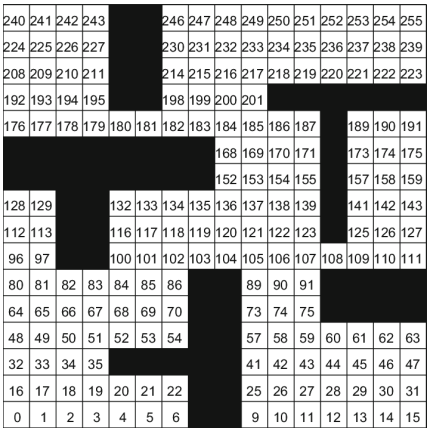
Dva hlavné druhy máp, ktoré sa v oblasti mobilnej robotiky najčastejšie využívajú sú rastrová - mriežková mapa a topologická mapa. [4]

2.2.1 Okupačná mriežková mapa

Ide o pevnú reprezentáciu prostredia, ktorá delí priestor na sieť samostatných buniek, štvorcov s určitým rozmerom, reprezentujúcich danú časť prostredia. Každému štvorcu je priradená hodnota v intervale <0;1> podľa toho, ako je obsadený - okupovaný. [1]

Ak robot môže daným priestorom prejsť, štvorcu sa priradí hodnota 0. Ak je v priestore prekážka, priradí sa mu určitá nenulová hodnota. Ak je hodnota daného štvorca väčšia než užívateľom preddefinovaná prahová hodnota, robot štvorec vyhodnotí ako prekážku a bude sa jej snažiť vyhnúť.

Podstata okupačnej mriežkovej mapy je znázornená na Obrázku č. 2.2. Čierne bunky reprezentujú prekážku, zatiaľ čo biele reprezentujú voľný priestor.



Obrázok č. 2.2: Príklad okupačnej mriežkovej mapy [5]

Tvorbu okupačnej mriežkovej mapy je možné realizovať nasledovnými spôsobmi: [1]

- **Dekompozícia máp**
  Ak už existuje vopred dostupná reprezentácia prostredia - napr. plán budovy, možeme ju rozdeliť na jednotlivé bunky presne - Exact Cell Decomposition [6], alebo približne - Approximate Cell Decomposition [7]. Výsledkom činnosti je mriežková mapa s rovnomerným delením priestoru alebo s premenlivým delením priestoru podľa geometrie.

- **Snímanie prostredia**
  Tento spôsob je možné vykonať meraniami, napr. pomocou laserového skeneru. Robot spočiatku začína s prázdnou mapou. Jeho postupným prechodom priestorom sníma objekty okolo seba. Pri detekcii objektu sa príslušnej skupine štvorcov inkrementuje určitá preddefinovaná hodnota. Pri prekonaní prahovej hodnoty začne byť daný štvorec vyhodnocovaný ako prekážka. [1]

- **Ručná tvorba**
  Ručným prístupom je možné prekresliť existujúcu mapu do jej zjednodušenej monochromatickej podoby - čierna bunka reprezentuje prekážku, biela bunka reprezentuje voľný priestor. Táto podoba obsahuje iba dôležité prvky a objekty prostredia tak, aby bol čo najpresnejšie vystihnutý jeho charakter.

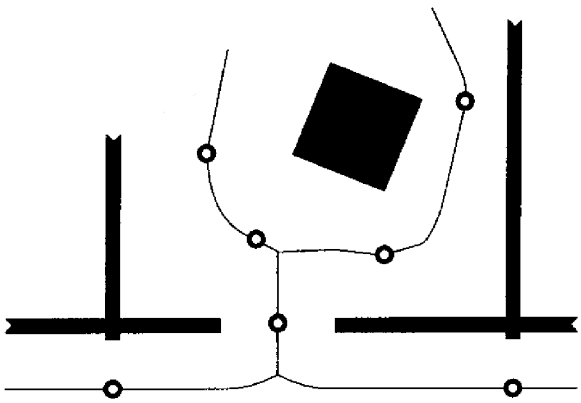
Hlavnou výhodou mriežkovej mapy je jednoznačnosť reprezentácie prostredia a možnosť úpravy jej rozmerov. Vytvorená mapa má zároveň názorný charakter pre človeka. S narastajúcimi rozmermi ide o druh mapy náročný na dostupnú pamäť a výkon riadiaceho systému.

Princíp okupačnej mriežkovej mapy je možné využiť ako na 2D, tak aj na 3D reprezentáciu priestoru, ale vzhľadom na spomenutú výpočtovú náročnosť nejde o vhodné riešenie, ak si to dané pracovné nasadenia priamo nevyžaduje. Príkladom 3D prístupu je framework Octomap [8].

2.2.2 Topologická mapa - graf

Podstatou topologickej mapy je reprezentácia každého unikátneho miesta v priestore pomocou uzlov a vzájomných prepojení medzi nimi. Vhodným príkladom je mapa metra alebo poschodie kancelárskych priestorov. Uzly môžu predstavovať jednotlivé miestnosti a časti chodieb, a prepojenia vyjadrujú možnosť prechodu medzi uzlami. Takto je priestor reprezentovaný stromovou štruktúrou - v tomto prípade ide o graf. [9]

Príklad topologickej mapy je na Obrázku č. 2.3.



Obrázok č. 2.3: Topologická mapa získana dekompozíciou prostredia [10]

Tvorbu topologickej mapy je možné vykonať nasledovne: [1]

- **Dekompozícia**

  Už existujúcu mriežkovú mapu je možné prerobiť na topologickú. K tomu sú zvyčajne využívané metódy matematickej teselácie [11]. Podstatou je mapu rozložiť na majoritné prvky na základe určitých princípov použitej metódy dekompozície, ktoré budú slúžiť ako hlavné navigačné uzly.

  Jedným z hlavných spôsobov je metóda Voroného diagramov [12], ale bežná je aj tvorba vlastnej metódy teselácie, príkladom je [13, 14].

- **Extrakcia prvkov robotom**

  Pomocou snímania prostredia robotom je možné za chodu vytvárať uzly na základe zadefinovaných podmienok. Väčšinou ide o komplexnejší proces, keďže je potrebné použiť pokročilejšie techniky spracovania získaných dát.

Vhodný prístup je rozpoznávanie obrazu z kamery a hľadanie unikátnych vlastností okolia robota. Ak robot viackrát úspešne rozpozná prvky prostredia v ktorom sa práve nachádza, môže určitú oblasť okolo jeho pozície považovať za uzol topologickej mapy, ktorá je takto postupne vytváraná. [15]

- **Strojové učenie**
  Metódami strojového učenia je možné vytvoriť algoritmy, ktoré z poskytnutých dát vyextrahujú uzly topologickej mapy. Častá je kombinácia týchto metód s predchádzajúcimi dvomi prístupmi. [16]

Navigácia čisto iba pomocou topologickej mapy je náročná na implementáciu, keďže neumožňuje robotovi vykonať priamy odhad jeho aktuálnej polohy. Nevýhodou je taktiež strata prehľadnosti pre človeka pri narastajúcom počte uzlov - názornosť mapy je prakticky nulová, ak sa využije na 3D reprezentáciu prostredia. Avšak pri správnej implementácii môže byť topologická mapa použitá spolu s inou, napr. mriežkovou mapou, pre dokonalejšiu metódu navigácie.

Mriežkové a topologické mapy často vychádzajú z predpokladu statickej reprezentácie prostredia. Tento predpoklad je v rozpore s dynamickým prostredím, v ktorom sú často roboty nasadené, čo môže viesť k navigačným chybám.

Záchranou môžu byť riešenia využívajúce SLAM algoritmy v konjunkcii s mriežkovou mapou, ktoré sú upravené tak, aby fungovali v mierne premenlivých prostrediach.

Príkladom je zavedenie algoritmu postupnej „degradácie“ jednotlivých oskenovaných častí okupačnej mapy. Jednotlivé bunky mriežkovej mapy postupom času strácajú svoju hodnotu a je nutné ich obnoviť opätovným prechodom prostredím. [1]

Príkladom prístupu k mapovaniu pre dynamické prostredia je algoritmus DOGMA (Dynamic Occupancy Grid Mapping Algorithm), ktorý využíva viacero okupačných máp a algoritmy na rozpoznanie premenlivých objektov. [17]

Z vlastností mriežkových a topologických máp vyplýva motivácia ich vzájomného prepojenia do jednej hybridnej mapy, čím je čiastočne možné znegovať ich nevýhody. Túto hybridnú mapu a celkovú navigáciu je ďalej možné vylepšiť implementáciou ďalších procesov, napríklad už spomenuté rozpoznávanie obrazu alebo tvorba hĺbkovej mapy zo snímku získaného kamerou. [10, 15]

2.3 Robot Operating System

Robot Operating System, ďalej ako ROS, je súbor open-source softvérových nástrojov pre vývoj robotických systémov. Ide o middleware vrstvu spadajúcu medzi riadiaci operačný systém a hardvér robota. Vývoj ROS začal v roku 2007 v robotickom laboratóriu na Stanfordskej Univerzite v Kalifornii, USA a aktuálne je pod správou neziskovej organizácie Open Robotics.

Cieľom projektu je tieto nástroje sprístupniť širokej verejnosti. Dôraz sa kladie na rozsiahlu podporu knižníc a programovacích jazykov, jednoduchú škáľovateľnosť a testovanie, pričom samotný softvéreový balíček ostáva jednoducho integrovateľný do nových projektov. [18]

V rámci bakalárskej práce bola použitá distribúcia ROS Noetic Ninjemys bežiaca vo virtualizovanom systéme Ubuntu 20.04 cez softvér Oracle VirtualBox 7.0.12. Ide o poslednú hlavnú ROS1 distribúciu s ukončením podpory v máji 2025. Postupnou náhradou sa stávajú distribúcie pod označením ROS2, ktoré majú plánované vydania každý júl a december.

ROS2 prináša viacero vylepšení kvality života. Hlavným rozdielom je rozšírenie podpory pre operačné systémy Windows a MacOS. Jeho základná knižnica je písaná v jazyku C, na ktorej sú stavané ostatné knižnice. Pre ne je okrem jazykov Python a C++ podpora rozšírená aj napr. pre Javu a C#.

Riadenie je sprostredkované službou DDS - Data Distribution Service, ktorá umožňuje efektívnejšiu a spoľahlivejšiu komunikáciu jednotlivých procesov. Umožnená je aj ich paralelizácia vďaka podpore multi-threadingu viacjadrových procesorov, čo s už spomenutými zmenami umožňuje lepšie uplatnenie v real-time aplikáciách. [19, 20]

2.3.1 Výpočtový graf

Podstatou fungovania prostredia ROS je „computation graph" - výpočtový graf. Ide o peer-to-peer sieť jednotlivých procesov, ktoré medzi sebou komunikujú a podieľajú sa na fungovaní systému robota ako celku. Možné je nasadenie širokej škály dostupných knižníc.

Najpodstatnejšie koncepty výpočtového grafu sú nasledovné: [21]

- **Nodes - Uzly**
  Uzly sú jednotlivé výpočtové procesy, programy bežiace na pozadí, komunikujúce medzi sebou vo vzťahu „publisher - subscriber", teda jeden uzol publikuje dáta a druhý ich odoberá. Jeden uzol môže zároveň odoberať aj publikovať dáta.

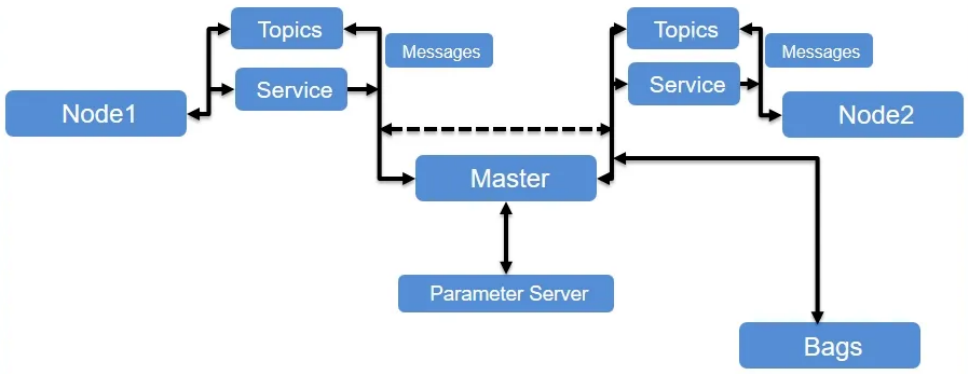
- **ROS Master**
  ROS Master je riadiaci uzol nadradený všetkým ostatným uzlom, ktorým umožňuje komunikovať medzi sebou. Jeho hlavnou úlohou je zaobstaranie menovacej služby, ktorej jednotlivé uzly poskytnú informácie o sebe. Master im spätne poskytne informácie o ostatných práve bežiacich uzloch, s ktorými môžu nadviazať spojenie.

- **Messages - Správy**
  Komunikácia medzi uzlami je možná pomocou správ. Každá správa má svoju definovanú štruktúru, ktorá udáva počet a poradie prvkov správy. Prvky správy majú podobu bežných dátových typov - napr. int, string, bool, atď. Pri komunikácii je potrebné štruktúru danej správy dodržať, aby bola správne spracovaná prijímacími uzlami.

- **Topics - Témy**
  Správy je možné medzi uzlami posielať cez jednotlivé témy. Názov témy umožňuje identifikovať správu. Na jednu tému je možné publikovať správy od viacerých uzlov naraz.

- **Services - Služby**
  Služba je definovaná párom správ - jedna pre požiadavku a druhá pre odpoveď. Táto komunikácia je možná, keď je jeden uzol komunikácie nastavený ako poskytovateľ služby. Tento uzol príjme dáta v podobe správy od klientského uzla a na základe vnútornej logiky mu poskytne odpoveď hneď, ako bude dostupná. Takto je umožnená asynchrónna komunikácia medzi uzlami v prípadoch, kedy odpoveď nie je dostupná okamžite kvôli akciám, ktoré sa medzičasom potrebujú vykonať.

- **Bags - Bagy**
  Ide o súbor - záznam správ z určitej témy a dát uzlov z určitého časového úseku. Zber dát je uskutočnený na základe požiadavky užívateľa a poskytuje základný spôsob debugovania činnosti uzlov.



Obrázok č. 2.4: Graf ROS komunikácie medzi uzlami [22]

Podstatu výpočtového grafu vyjadruje Obrázok č. 2.4 popisujúci komunikáciu medzi dvomi ROS uzlami.

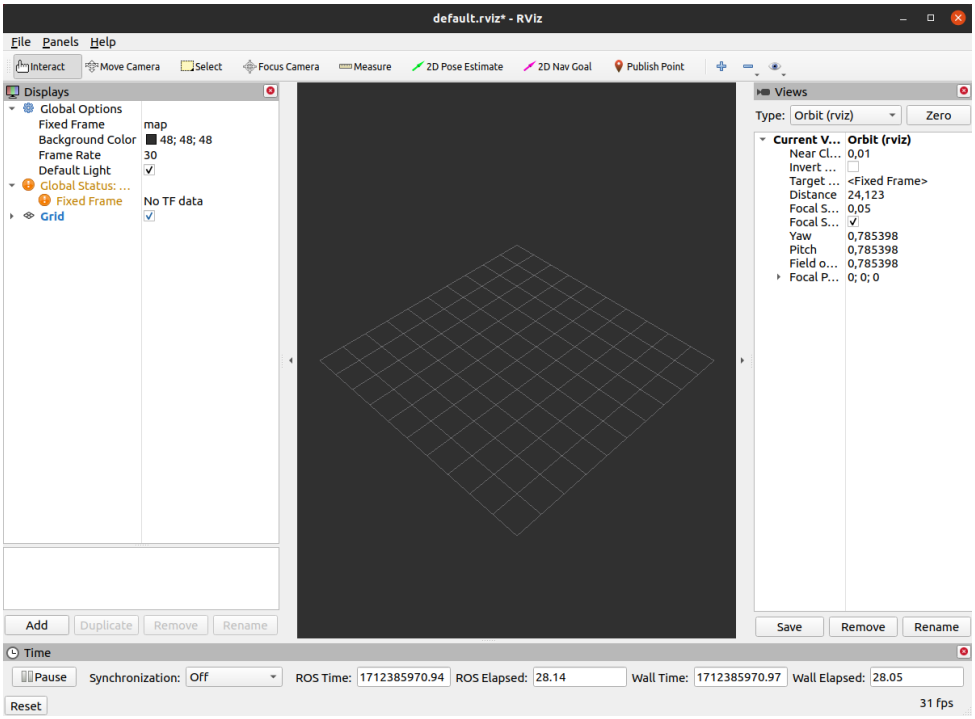
2.3.2 Simulačné nástroje

Prostredie ROS disponuje silným, ľahko integrovateľným softvérom pre testovanie a simuláciu situácií. Medzi najrozšírenejšie simulačné nástroje patrí RViz a Gazebo.

RViz

RViz je softvérový balík pracujúci priamo v rámci prostredia ROS. Jeho hlavnou prednosťou je kompatibilita s viacerými bežne používanými balíkmi. To prináša možnosť vizualizácie dát v 2D aj v 3D.

Príkladom je vizualizácia objektov vyhodnotených ako prekážky z detekcie laserom, ale aj vykreslenie aktuálne zvolenej cesty od plánovača ciest alebo transformačných vzťahov medzi súradnicovými systémami častí robota. Softvér disponuje aj možnosťou zadávania 2D navigačného cieľa priamo prostredníctvom užívateľského rozhrania programu, ktoré je na Obrázku č. 2.5. [23, 24]



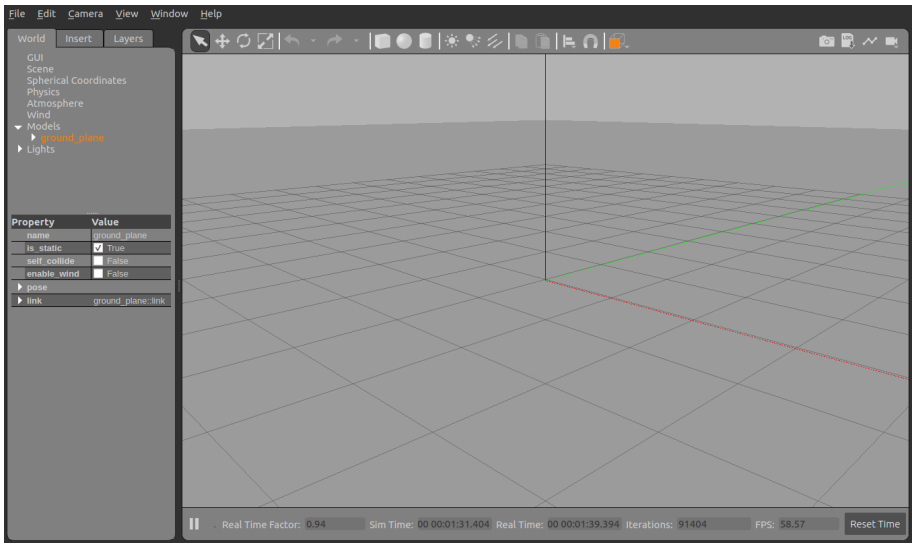
Obrázok č. 2.5: Základné užívateľské rozhranie programu RViz

Gazebo

Gazebo je samostatný simulačný softvér s úzkou integráciou s ROS cez API. Určený je predovšetkým na tvorbu 3D simulačných svetov, čo je umožnené vďaka palete nástrojov. Tie zároveň poskytujú prostriedky na tvorbu a pridávanie objektov do týchto svetov. Vytváranie svetov je uľahčené vďaka integrácii s cloud prostredím disponujúcim knižnicou objektov.

Softvér je vhodný pre simulácie, ktorých súčasťou je interakcia robota so svojim okolím. Nevýhodou oproti RViz je menej možností vizualizácie dát z prostredia [25].

Obrázok č. 2.6. znázorňuje užívateľské prostredie Gazebo.



Obrázok č. 2.6: Prázdny simulačný svet v užívateľskom prostredí Gazebo

Najelegantnejší spôsob, ako spustiť simuláciu s požadovanými vstupnými parametrami je pomocou .launch súboru. Ide o súbor formátu XML, v ktorom je možné zadefinovať vstupné parametre a potrebné uzly, ktoré sa majú spustiť.

Do jedného hlavného .launch súboru je možnosť vnoriť samostatné .launch podsúbory, čo umožňuje užívateľovi sprehľadniť svoj kód. Ide o jednoduchý spôsob definovania procesov potrebných pre spúšťanie simulácie, ale aj samotných robotických systémov.

2.3.3 ROS balíky

ROS balíky predstavujú samostatné sady programov pre jednotlivé situácie použitia. Medzi uzly balíkov, ktoré boli kľúčové v rámci vypracovania zadania patria:

- **map_server**
  Ide o ROS uzol publikujúci dáta zvolenej mapy ako službu. Tieto dáta sú publikované na tému **/map**. Pri spúšťaní uzla je potrebné špecifikovať systémovú cestu k .png súboru reprezentujúcemu mapu a .yaml súboru mapy. Ten popisuje parametre mapy, ako je jej rozmer a požadované rozlíšenie, vďaka čomu je súbor .png spracovaný ako mriežková mapa. [26]

- **move_base**
  Tento uzol odoberá informácie ohľadom transformácie súradnicových systémov jednotlivých častí robota a publikovanej mriežkovej mapy. Na tému **move_base/goal** mu je možné poslať pozíciu, do ktorej má byť robot navigovaný. Následne podľa nastavených vstupných parametrov je pohyb robota riadený lokálnym a globálnym plánovačom. [27]

Ostatné balíky, ktoré boli používané v pozadí práce, ale neboli predmetom jej riešenia sú: tf2, joint_state_publisher, robot_state_publisher, amcl. [28 - 31]

2.3.4 Prevzaté súbory

Pre účely návrhu mechanizmu bol prevzatý repozitár System_ZL a Simulation_ZL z platformy GitHub. Ide o riadiaci kód projektu autonómneho zalievacieho robota Zaleela, navhrnutého na robotickej platforme Breach. Spomínaný robot bol predmetom diplomovej práce z roku 2022. Na celkovom návrhu sa podieľalo päť študentov, ktorých diplomové práce popisujú návrh jednotlivých častí zalievacej platformy. [32]

Repozitáre obsahuje balíky potrebné pre chod skutočného robota a pre jeho simuláciu a vizualizáciu dát. Pre účely bakalárskej práce boli prevzaté nasledujúce súbory:

- System_ZL/zaleela_main

Hlavný simulačný balík na spustenie jednotlivých procesov a uzlov simulačného testu.

- **simulation.launch**
  Hlavný .launch súbor definujúci vstupné argumenty potrebné pre jednotlivé uzly, zároveň zabezpečujúci spúšťanie samostatných .launch podsúborov (navigation.launch a gazebo.launch) a simulačného klienta RViz. Kvôli prehľadnosti boli z prevzatého súboru vymazané riadky spúšťajúce nepoužité uzly a syntax bola mierne prepísaná.

- **planning.launch**
  Úlohou súboru je spustenie uzlov potrebných pre navigáciu robota, menovite hlavne move_base pre posielanie navigačných cieľov a amcl pre lokalizáciu. Súbor bol premenovaný na **navigation.launch**.

- System_ZL/planning

- **transfer.py**
  Súbor spúšťa uzol transform, ktorý v podobe správy preposiela transformáciu súradnicového systému base_footprint do súradnícového systému base_link počas chodu simulácie. Súbor bol premenovaný na **transform.py**.

- **Parametrové súbory**

  - global_costmap_params.yaml

  - local_costmap_params.yaml

  - costmap_common_params.yaml

  - base_local_planner_params.yaml

  Súbory popisujú dodatočné argumenty simulácie. Príkladom sú tolerancie navigačných cieľov, povolené translačné a rotačné rýchlosti a zrýchlenia robota, alebo parametre súvisiace s vizualizáciou dát z laserových snímačov.

- Simulation_ZL/sim_zl

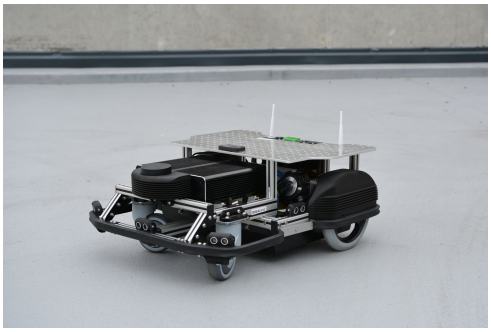
Balík s prostriedkami potrebnými pre spustenie a nastavenie fyzikálnych parametrov simulácie a navigačných uzlov.

- **simulation_gazebo.launch**
  Súbor spúšťa simulačný server a klient Gazebo, transformačný program transform.py a physics_parameter_script.py. Uzol map_server načíta požadovanú mapu podľa vstupných argumentov. Súbor bol premenovaný na **gazebo.launch**.

- **physics_parameter_script.py**
  Program definujúci medze simulačného kroku a frekvencie, a iné parametre súvisiace s chodom simulácie. Premenovaný bol na **physics_parameter.py**.

2.4 Mobilná robotická platforma Breach

Mobilná robotická platforma Breach je mobilný robot vyvinutý spoločnosťou Bender Robotics, ktorý je znázornený na Obrázku č. 2.7. Ide o robot s diferenciálnym pohonom dvoch kolies a vysokou modularitou, vybavený LIDAR modulom, určený hlavne pre oblasť výskumu a vývoja, a diaľkový prieskum. [33]



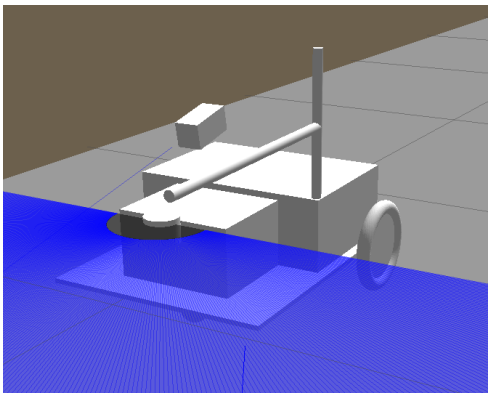
Obrázok č. 2.7: Jeden z mobilných robotov Breach [33]

Špecifikácie robota sú uvedené v Tabuľke č. 2.1.

Celkové rozmery (DxŠxV)	620 × 540 × 255 mm
Typ podvozku	diferenciálny s pohonom dvoch kolies
Hmotnosť platformy	34 kg
Maximálna nosnosť	20 kg
Prevádzková rýchlosť	až 3,0 km/h
Dojazd na jedno nabitie	až 10 km
Výdrž batérie na jedno nabitie	až 8 h
Určenie	interiér

Tabuľka č. 2.1: Tabuľka špecifikácií platformy Breach [33]

Pri návrhu hybridnej mapy bol pre účely simulácie prevzatý URDF model robota Breach a jeho vybavenia. Ide o textové súbory popisujúce jednotlivé časti robota a ich špecifikácie pre účely simulácie, napríklad rozmery a momenty zotrvačnosti [34]. URDF model je možné vidieť v rámci simulačného prostredia Gazebo na Obrázku č. 2.8.



Obrázok č. 2.8: URDF model mobilného robota Breach

Z vykonanej rešerše vzišla myšlienka návrhu hybridnej mapy, ktorá vznikne vzájomným prepojením samostatných mriežkových máp jednotlivých poschodí pomocou jednej topologickej mapy vyjadrujúcej prechody medzi týmito poschodiami a ich miestnosťami.

Pri prechode robota medzi poschodiami by na pokyn riadiaceho programu bola zmenená aktuálne publikovaná mapa podľa potreby, čo by umožnilo robotovi pokračovať v navigácii na novom poschodí.

Na základe tejto myšlienky boli stanovené dielčie ciele práce:

(a) Návrh mechanizmu zmeny mriežkovej mapy pri zmene poschodia

(b) Návrh testovacích mriežkových máp poschodí

(c) Návrh topologickej mapy

(d) Návrh testovania funkčnosti mechanizmu

(e) Otestovanie navrhnutého mechanizmu

Ich vypracovanie je popísané v kapitole č. 3.

3 Návrh hybridnej mapy

Pre reprezentáciu priestorov viacposchodovej budovy bol navrhnutý následovný spôsob mechanizmu hybridnej mapy pre navigáciu robota medzi poschodiami:

- Jednotlivé poschodia budovy sú reprezentované samostatnými okupačnými mriežkovými mapami.
- V každej mriežkovej mape je definovaná prechodová zóna pre účely zmeny mapy. Táto prechodová zóna reprezentuje výťah budovy.
- Jedna topologická mapa vzájomne myšlienkovo prepája mriežkové mapy pomocou prechodových zón.
- Zastavením robota v prechodovej zóne nastane zmena aktuálnej mriežkovej mapy prostredníctvom riadiaceho kódu podľa topologickej mapy. Zároveň je vykonaný reštart riadiacich uzlov robota.
- Robot následne môže pokračovať v činnosti na novom poschodí.

V následovných podkapitolách je popísaná konkrétna problematika a postrehy pri návrhu mechanizmu hybridnej mapy. Súčasťou kapitoly je popis navrhnutého testovacieho a riadiaceho kódu simulácie.

3.1 Návrh metódy zmeny mapy

Hlavným krokom pri návrhu hybridnej mapy bolo zistenie možných spôsobov zmeny mriežkovej mapy pri prechode medzi poschodiami. Podľa dostupných článkov [35] bola preskúmaná vhodnosť implementácie štyroch metód:

- **map_restart**
  V rámci prostredia ROS existuje API pre integráciu s Python kódom - rospy [36], ktoré umožňuje písať programy ovládajúce prvky prostredia ROS. Možné je spustiť a ukončiť uzly, poskytovať služby či nastavovať vstupné parametre a i.
  Týmto spôsobom je možné reštartovať jeden hlavný map_server uzol, ktorý po každom reštarte publikuje aktuálne potrebnú mapu.

- **multiplexer**
  Spustením hlavného programu sú spustené individuálne map_server uzly pre všetky potrebné mapy poschodí. Tie sú dostupné na jednotlivých, vhodne označených /map témach. V teórii následne môžeme vyberať momentálne potrebnú mapu spomedzi spustených uzlov pomocou nástroja twist_mux [37], ktorá začne byť publikovaná na hlavný /map topic.
  Problémom je, že mapy sú z jednotlivých map_serverov uverejňované na príslušné témy len raz, a to pri spustení uzla. Multiplexovaním medzi témami nie je teda možné získať aktuálne potrebnú mapu pokiaľ je simulácia už spustená. Pre správnu funkciu je nutné po každom multiplexingu reštartovať príslušný map_server uzol. Tento spôsob riešenia vedie k prvej metóde, ktorá je výhodnejšia kvôli menšej náročnosti na pamäť systému.

- **load_map**
  V rámci ROS dokumentácie balíka nav_msgs [38] sú spomenuté služby LoadMap a SetMap, ktoré majú poskytnúť jednoduchý spôsob zmeny mapy poslaním správy na dané služby. Kvôli nedostatočnej dokumentácii sa v rámci riešenia aj napriek viacerým pokusom nepodarilo nájsť spôsob, ako túto metódu spojazdniť a preto bola zamietnutá. Predpokladom je, že tieto služby sú viac rozpracované a funkčné až v distribúciach ROS2.

- **map_islands**
  Najjednoduchšou metódou bez využitia akýchkoľvek služieb uzlov je vytvorenie jednej veľkej mapy obsahujúcej dielčie mapy poschodí v podobe samostatných oddelených ostrovov. Pri zmene poschodia by bolo možné podľa potreby nastavovať aktuálne potrebnú pozíciu robota na danú časť mapy. Táto metóda je najnáročnejšia na simulovanie kvôli nárokom na pamäť a výpočtový výkon, a teda pri väčších mapách by bolo nevhodné ju nasadiť.

Na základne výhod a nevýhod jednotlivých metód bola vybraná metóda **map_restart**, ktorá predstavovala najvhodnejší kompromis medzi výpočtovou náročnosťou a jednoduchosťou použitia.

Na otestovanie metódy bol navrhnutý testovací program **test_map_restart.py.** V ňom je inicializovaný uzol map_restart, ktorý je periodicky zapínaný a vypínaný podľa toho, aká mapa je aktuálne požadovaná. Publikovaná mapa je vizualizovaná pomocou RViz. Cez .launch súbor bola spustená simulácia so všetkými potrebnými uzlami.

Pseudokód testovacieho programu map_server je následovný:

```
Inicializácia

Zadefinovanie požadovanej mapy

WHILE rospy je spustený:

    IF požadovaná mapa je poschodie 1:

        Spustenie uzla map_server s mapou poschodia 1

        Čakanie 5 sekúnd

        Vypnutie uzla map_server

    ELSE:

        Spustenie uzla map_server s mapou poschodia 2

        Čakanie 5 sekúnd

        Vypnutie uzla map_server

    Čakanie 2 sekundy
```

Horná hranica spoľahlivosti tejto metódy bola pri publikovaní mapy s rozmerom 6000 na 6000 pixelov pri rozmerovom rozlíšení 0.05 metra na pixel. Zmena mapy bola vykonaná do sekundy.

Pri publikovaní väčšej mapy alebo pri použití väčšieho rozmerového rozlíšenia začala spoľahlivosť metódy značne klesať. Simulácia dokázala správne fungovať najviac šesť opakovaní, po ktorých prestal reagovať testovací program. Pri niektorých testoch bol zaznamenaný pád celého virtualizovaného systému.

Tieto obmedzenia sú zapríčinené virtualizáciou.

3.2 Návrh metódy reštartu uzlov

Pri zvolenom spôsobe zmeny mapy je potrebné pre správnu funkciu systému reštartovať všetky používané uzly. Existuje viacero spôsobov, ako túto činnosť vykonať. Príkladom sú tieto prístupy:

- **roslauncher.py**
  Podobne ako pri zmene mapy, výhodným spôsobom je využitie rospy knižnice. Pomocou programu môžeme reštartovať uzly spúšťané cez .launch súbor a predať im nové vstupné argumenty. Zároveň sú reštartované procesy simulačných softvérov.

- **Konzolové príkazy**
  Pomocou príkazov zadávaných do systémovej konzoly Ubuntu môžeme vypínať a spúšťať nové ROS uzly bez nutnosti vypnutia simulačného softvéru. Zadávanie týchto príkazov môžme vykonať aj použitím dostupných knižníc v rámci kódu.

  Spomenutý spôsob nie je vhodný kvôli premenlivosti v počte uzlov, čo sa prejaví v premenlivosti potrebných príkazov na správne vykonanie reštartu. Aj samotný simulačný softvér mal pri testoch metódy premenlivé správanie.

- **Viacero svetov**
  Teoretickým prístupom je možnosť spustenia viacerých simulačných svetov pre každú z máp a nasledovné prepínanie medzi nimi. Metóda je náročná na implementáciu, nie je k nej dostupná žiadna dokumentácia a predstavuje vysokú výpočtovú náročnosť.

Zvolenou metódou pre reštart uzlov je **roslauncher.py**. Ide o program riadiaci činnosť hlavného simulačného súboru simulation.launch, ktorý je v podobe objektu. Daný objekt je podľa potreby reštartovaný, čo umožňuje reštart všetkých simulačných uzlov.

Pred plným nasadením všetkých funkcií bol navrhnutý program **test_roslauncher.py**. Ním bolo otestované správanie jednotlivých uzlov a simulačných klientov Gazebo a RViz pri viacerých opakovaniach pokynu na reštart prostredníctvom while cyklu.

Pseudokód programu test_roslauncher je následovný:

```
WHILE rospy je spustený:
    Vyvolanie objektu "Svet 1" s požadovanou pozíciou robota
    Čakanie 20 sekúnd
    Volanie metódy na ukončenie simulácie
    Čakanie 20 sekúnd
    Vyvolanie objektu "Svet 2" s požadovanou pozíciou robota
    Čakanie 20 sekúnd
    Volanie metódy na ukončenie simulácie
    Čakanie 20 sekúnd
```

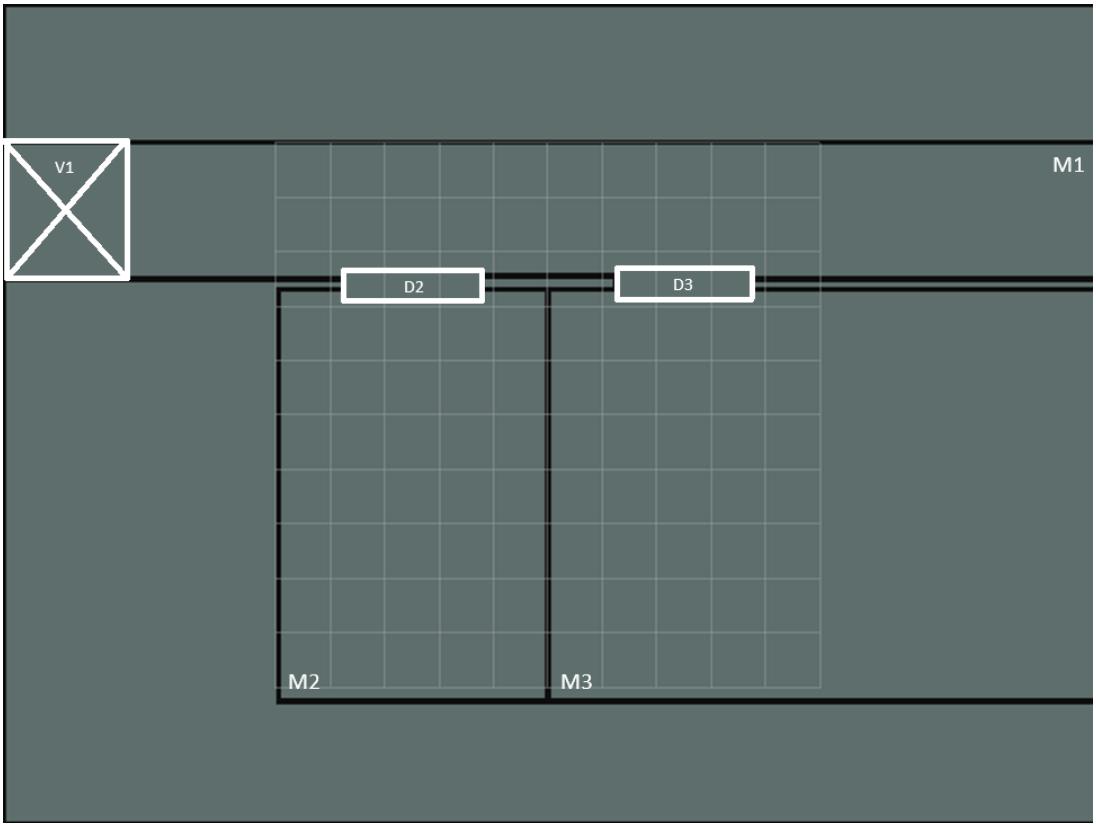
Aj po viacerých opakovaniach nebola objavená chyba, ktorá by nejak narušovala priebeh simulácie. Jediným spozorovaným problémom je nesprávne ukončenie simulácie pri jej manuálnom prerušení. To vedie k potrebe otvorenia nového okna terminálu pred opätovným spustením simulácie.

3.3 Návrh testovacích máp

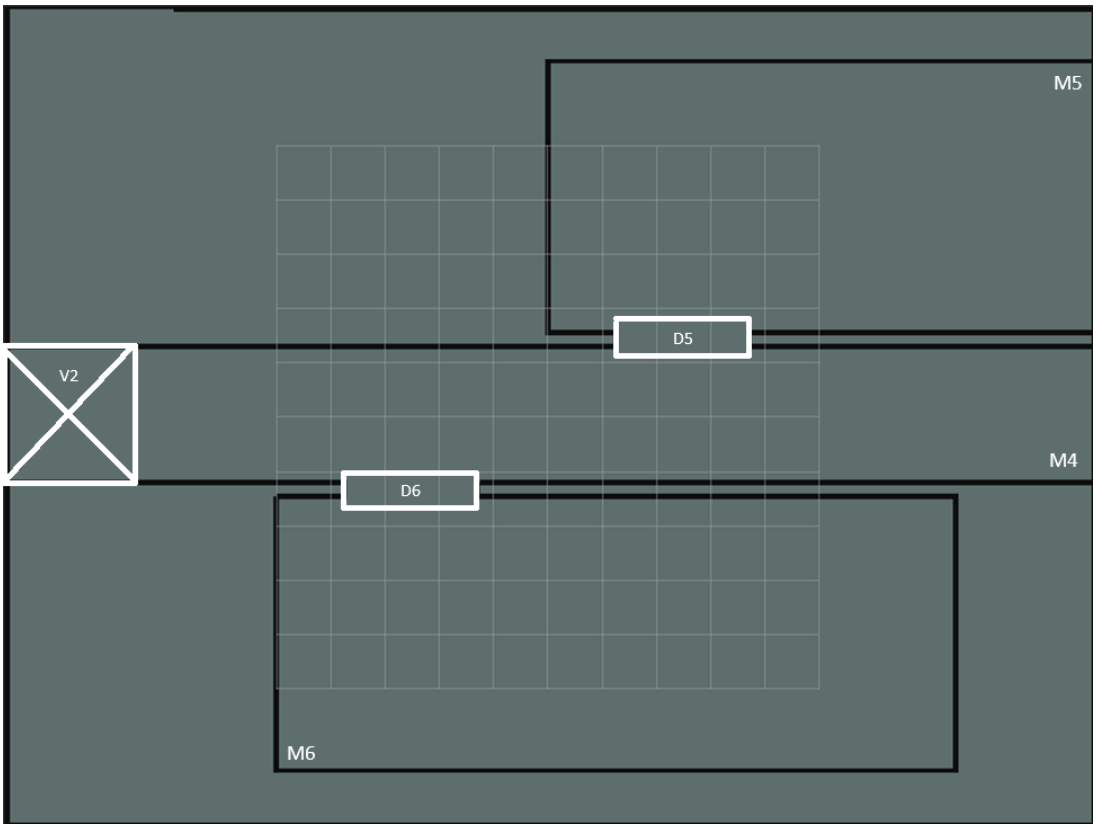
Pre účely testovania mechanizmu hybridnej mapy boli navrhnuté dve mriežkové mapy a k ním príslušné simulačné gazebo svety. Na prepojenie mriežkových máp bola navrhnutá topologická mapa vyjadrujúca vzťahy medzi miestnosťami poschodí.

Navrhnuté rastrové mapy reprezentujú prvé a druhé poschodie viacposchodovej budovy. Obe pozostávajú z centrálnej chodby s výťahom, ktorá vedie do dvoch miestností znázorňujúcich kancelárske priestory a sklad. Mapy sú formátu .png a majú rozlíšenie 401 na 301 pixelov. Pre simuláciu bolo v .yaml súboroch nastavené rozmerové rozlíšenie 0,05 metra na pixel, teda reprezentovaný priestor má rozmery 20,05 na 15,05 metra.

Mriežkové mapy boli navrhnuté pomocou softvéru Inkscape kvôli možnosti ich jednoduchej úpravy. Znázornené sú na Obrázku č. 3.1. a Obrázku č. 3.2.



Obrázok č. 3.1: Mriežková mapa prvého poschodia

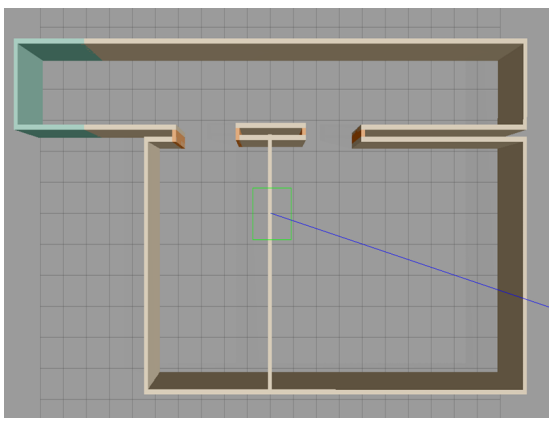


Obrázok č. 3.2: Mriežková mapa druhého poschodia

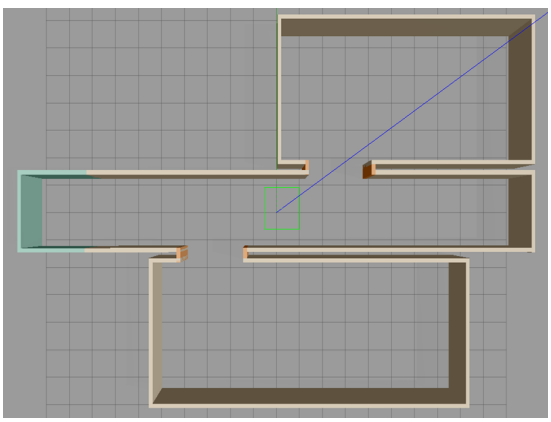
Pre účely zmeny mapy a vhodnú navigáciu medzi miestnosťami boli priestory mriežkových máp myšlienkovo rozdelené na jednotlivé miestnosti. Tieto miestnosti sú, vrátane chodby, pod označením Mx, pričom x reprezentuje príslušné číslo. Medzi týmito miestnosťami boli navrhnuté prechodové zóny „dvere“ s označeniami Dx. Okrem dverí sú na mape vytvorené hlavné prechodové zóny „výťahy“ označené Vx, v ktorých dôjde k zmene používanej mapy.

Kvôli simulácii boli navrhnuté aj dva simulačné svety pomocou nástrojov softvéru Gazebo. Ide o 3D reprezentáciu poschodí pozostávajúce zo stien, v rámci ktorých bude robot navigovaný pomocou príslušnej mriežkovej mapy.

Svety sú určené len pre účely vizualizácie, nie sú súčasťou mechanizmu hybridnej mapy. Znážornené sú na Obrázku č. 3.3. a Obrázku č. 3.4.



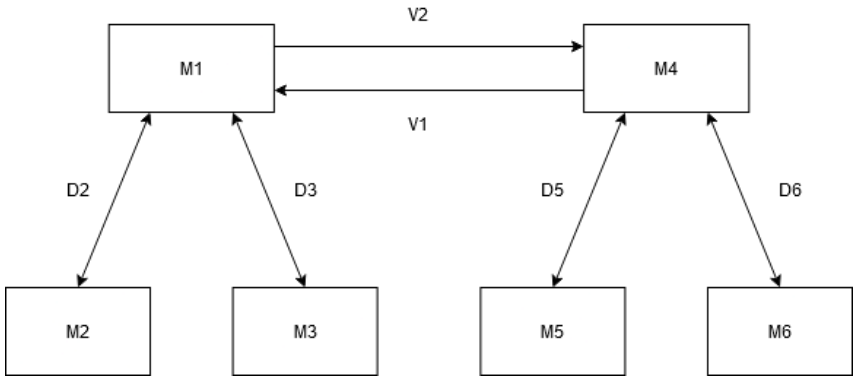
Obrázok č. 3.3: Poschodie 1



Obrázok č. 3.4: Poschodie 2

Takto rozdelený priestor je možné reprezentovať pomocou topologickej mapy vyjadrujúcej možné prechody medzi jednotlivými miestnosťami cez prechodové zóny.

Obrážok č. 3.5. zobrazuje navrhnutú topologickú mapu.



Obrázok č. 3.5: Topologická mapa prepájajúca navrhnuté mriežkové mapy

Mriežkové mapy jednotlivých poschodí je možné rozdeliť na ešte menšie podmapy jednotlivých miestností pri, zachovaní prechodových zón dverí pre účely zmeny mapy, čo umožní zníženie pamäťovej náročnosti na systém robota pri väčších mapách.

Následne by mohlo byť toto odľahčenie použité v konjunkcii so zväčšením rozlíšenia podmáp, ak je navigované prostredie komplexné a obsahuje veľa menších prvkov, ktoré je nutné robotovi reprezentovať ako prekážky.

3.4 Návrh testovania hybridnej mapy

V tejto podkapitole je rozpísaný spôsob testovania spoľahlivosti navrhnutej hybridnej mapy. Pre účely testovania bol vytvorený hlavný testovací program **run_simulation.py**, v rámci ktorého figurujú štyri hlavné funkcie:

- **funInitWorld**

  Funkcia zabezpečuje nastavenie správnych vstupných argumentov pre spustenie simulačného sveta na základe vstupu. Výstupom z funkcie je objekt spúšťajúci simuláciu prostredníctvom súboru simulation.launch s požadovanou štartovacou miestnosťou.

  Vyvolaním objektu sú spustené všetky potrebné ROS uzly pre navigáciu robota, server a klient simulačného sveta Gazebo, a vizualizácia pomocou klienta RViz, ktoré bežia súčasne.

```
def funInitWorld(floor, pose_x, pose_y, pose_f):
    Prepočet "pose_f" z uhlov na radiány
    Inicializácia vstupných parametrov
    Definícia objektu "world" podľa vstupov funkcie
    RETURN "world"
```

- **funMovebaseClient**
  Funkcia spúšťa uzol - klient, ktorý na tému move_base/goal zašle navigačný cieľ s požadovanou pozíciou robota - súradnicami X, Y a natočením. Klient následne čaká na odpoveď, ktorá signalizuje úspešné dosiahnutie navigačného cieľa robotom.

```
def funInitWorld(poseX, poseY, poseOrientation):
    Zadefinovanie služby "move_base"
    Zadefinovanie objektu "goal"
    Zápis "poseX" do "goal"
    Zápis "poseY" do "goal"
    Prepočet "poseOrientation" z uhlov na radiány
    Prepočet a zápis "poseOrientation" do "quaternion"
    Zápis "quaternion" do "goal"
    Poslanie "goal" do "move_base"
    Čakanie na odpoveď
    IF odpoveď je dostupná:
        RETURN odpoveď
```

Pri lokálnom plánovaní trasy sa vychádzalo z metódy Dynamic Window Approach [39]. Použitý algoritmus na hľadanie cesty globálnym plánovačom bol A* [40].

- **funBFS**
  Funkcia konajúca algoritmus prehľadávania do šírky - Breadth First Search [41] a následnu úpravu poľa do požadovaného tvaru. Vstupom funkcie je dátová štruktúra list reprezentujúca topologickú mapu a výstupom je pole reprezentujúce postupnosť uzlov navigačnej cesty podľa začiatočnej a koncovej miestnosti.

```
def funBFS(graph, start, end):
    BFS algoritmus na získanie poľa "path" s miestnosťami
    Pridanie prechodových zón do "path" podľa sledu miestností
    Pridanie koncovej pozície do "path"
    Pridanie počiatočnej pozície do "path"
    Vymazanie označení miestností z "path"
    RETURN "path"
```

BFS algoritmus bol mierne upravený pridaním dodatočnej premennej, vďaka ktorej je možné nájsť navigačnú cestu, aj keď existuje viacero možných ciest medzi dvomi miestnosťami.

- **funGetPoses**
  Funkcia prepája ostatné funkcie. Vstupom je aktuálne požadovaný navigačný cieľ, podľa ktorého je výstupom z funkcie požadovaná pozícia robota. Pre účely zmeny mapy je súčasťou výstupu aj požadované poschodie, podľa ktorého sa nastaví správna mapa pri reštarte sveta.

```
def funGetPoses(node):
    IF "node" == niektorému z uzlov cesty:
        "x" = požadovaná súradnica v smere osi x
        "y" = požadovaná súradnica v smere osi y
        "orientation" = požadované natočenie
        "floor" = požadovaný názov poschodia
    RETURN "(node, x, y, orientation, floor)"
```

Po spustení testovacieho programu a vykonaní prvotnej inicializácie začína hlavný simulačný cyklus. V tomto cykle je robot navigovaný do príslušných miestností podľa určenej navigačnej slučky cez jednotlivé navigačné cesty.

Navigačná slučka reprezentuje požadovanú postupnosť miestností a navigačná cesta reprezentuje postupnosť uzlov pre navigáciu medzi dvoma miestnosťami. Počet opakovaní navigačnej slučky je definovaný na začiatku programu. Hlavný simulačný cyklus, a tým aj samotný program, končí po dosiahnutí požadovaného počtu opakovaní navigačnej slučky.

Základné činnosti, ktoré tento program vykonáva je možné zhrnúť nasledovným sledom činností:

- **Inicializácia**
  Súčasťou inicializácie je kontrola a prípadné ukončenie už spustenej simulácie, uzlov a ROS Mastera. Následne je spustený nový ROS Master, inicializované sú pomocné premenné a dátová štruktúra topologickej mapy - list. Po tomto kroku začína hlavný simulačný cyklus.

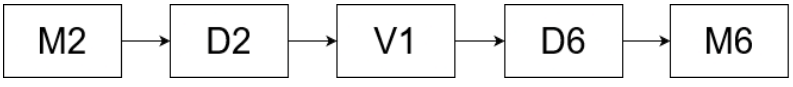
- **Hľadanie cesty**
  Pomocou funkcie funBFS sa nájdu všetky uzly aktuálnej navigačnej cesty pre požadovaný krok navigačnej slučky, ktoré sú potrebné pre navigáciu robota do zadanej miestnosti.

- **Navigácia**
  Ak sa jedna o prvú iteráciu navigačnej slučky, funkciou funInitWorld sa spustí simulačný svet so vstupnými parametrami, ktoré sú výstupom funkcie funGetPoses. Po rozbehnutí všetkých uzlov funkcia funMovebaseClient pošle robotovi nový navigačný cieľ podľa aktuálne požadovaného uzla navigačnej cesty.

- **Zmena mapy**
  Po dosiahnutí ľubovoľného navigačného uzla robotom je kontrolovaná podmienka, či dosiahnutý cieľ je jedna z platných prechodových zón. Ak áno, vykoná sa rada príkazov, ktoré ukončia aktuálne spustený simulačný svet a inicializujú nový s požadovanou mapou.

Pri každom úspešnom dosiahnutí koncového uzla navigačnej cesty sa podľa navigačnej slučky vyberie nová navigačná cesta do novej miestnosti. Pri každom úspešnom dokončení navigačnej cesty a pri každom úspešnom vykonaní navigačnej slučky je do dvoch príslušných premien inkrementovaná hodnota. Pri zmene hodnoty jednej z týchto premien sú kvôli signalizácii vypísané obe hodnoty do systémového terminálu.

Obrázok č. 3.6. znázorňuje jednu z možných navigačných ciest vychádzajúcich z navigačnej slučky. Konkrétne ide o uzly, cez ktoré je robot navigovaný pri prechode z miestnosti M2 do miestnosti M6.

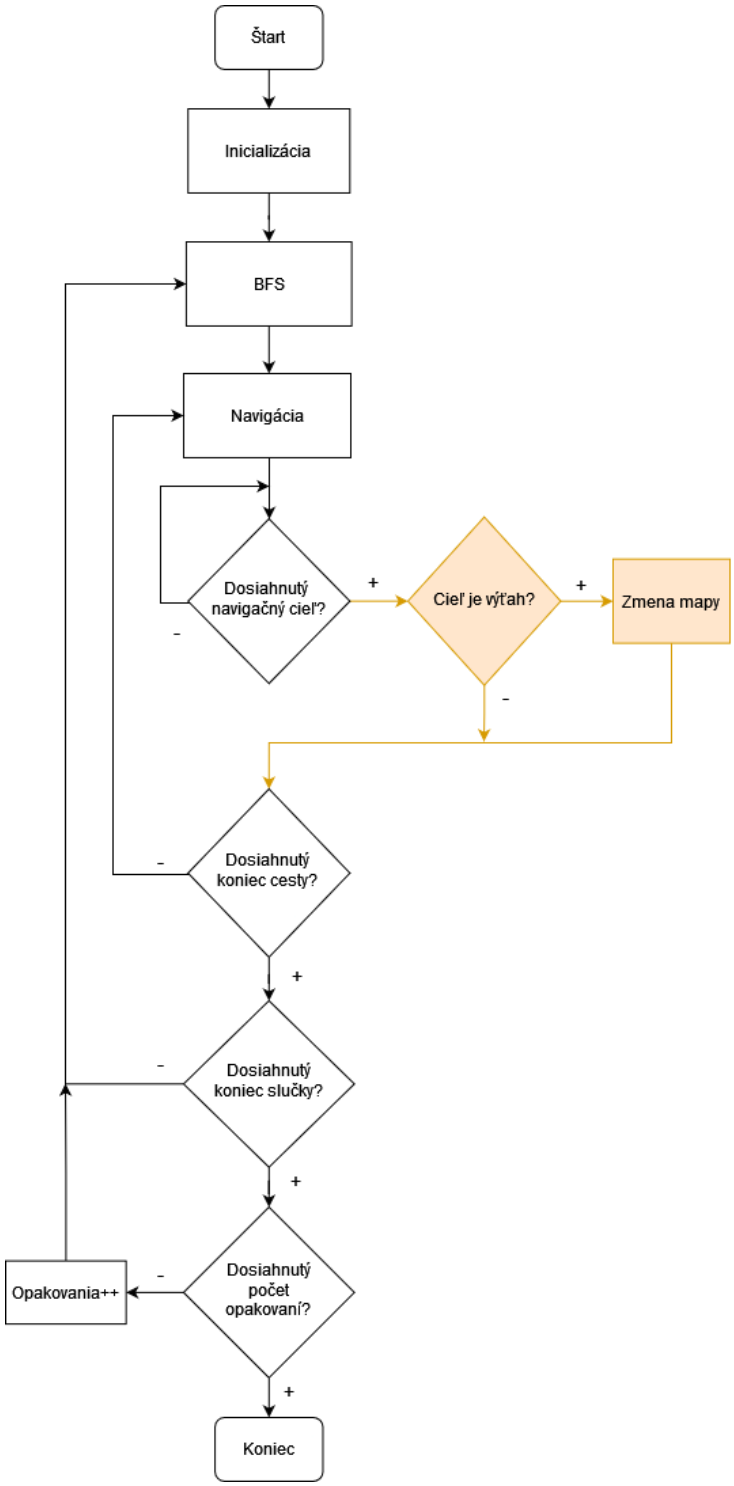


Obrázok č. 3.6: Príklad jednej z navigačných ciest

Program vykonávajúci vymenovaný sled činností je reprezentovaný následovným pseudokódom:

```
Ukončenie spusteného ROS Mastera
Spustenie nového ROS Mastera
Definícia topologickej mapy ako list "Topology"
Definícia navigačnej slučky ako pole "Route"
Inicializácia pomocných premenných
WHILE True:
    Získanie navigačnej cesty "Path" pomocou "funBFS" s definovaným "Route"
    FOR prvky v "Path":
        IF simulačný svet nie je spustený:
            Vyvolanie objektu "World" s vhodnou mapou cez funInitWorld()
        Inicializácia ROS uzla "move_base_client"
        Poslanie požadovanej polohy cez "funMovebaseClient"
        IF robot dosiahol navigačný cieľ:
            Výpis počtu prejdených uzlov do konzoly
            IF navigačný cieľ je prechodová zóna:
                Ukončenie simulačného sveta
                Vyvolanie objektu "World" s vhodnou mapou cez funInitWorld()
    END FOR
    IF dosiahnutý požadovaný počet opakovaní navigačnej slučky:
        BREAK
Ukončenie simulačného sveta
Výpis počtu prejdených uzlov do konzoly
```

Vymenovaný hlavný chod programu je možné znázorniť zjednodušeným vývojovým diagramom:



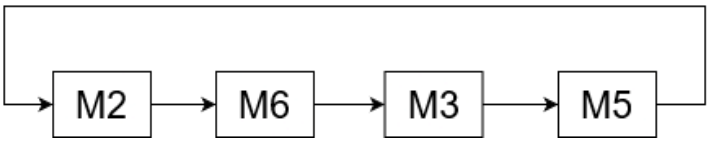
Obrázok č. 3.7: Vývojový diagram riadiaceho kódu simulácie

Prehľad hlavných použitých súborov je v časti Digitálna príloha. Jednotlivé programy sú súčasťou digitálne prílohy.

4 Výsledky simulačných testov

Mechanizmus spojenia viacerých máp do hybridnej mapy bolo nutné overiť. Táto kapitola popisuje konkrétny spôsob testovania, výsledky testov a ich zhodnotenie.

Navigačná slučka navrhnutá pre simulačné testy je znázornená na Obrázku č. 4.1.



Obrázok č. 4.1: Navrhnutá navigačná slučka pre simulácie

Štartovacia pozícia robota je v miestnosti M2 na prvom poschodí. Robot je postupne navigovaný jednotlivými navigačnými cestami do všetkých ostatných miestností prvého a druhého poschodia. Jedna navigačná cesta medzi dvomi miestnosťami obsahuje vždy jeden prechod medzi poschodiami, teda jednu potrebnú zmenu mapy.

Pre testovacie účely boli ako prechodové zóny brané iba uzly výťahov. V realite by bol tento krok reprezentovaný nástupom robota do výťahu a reštartovaním potrebných uzlov pred tým, než bude pokračovať ďalej v ceste.

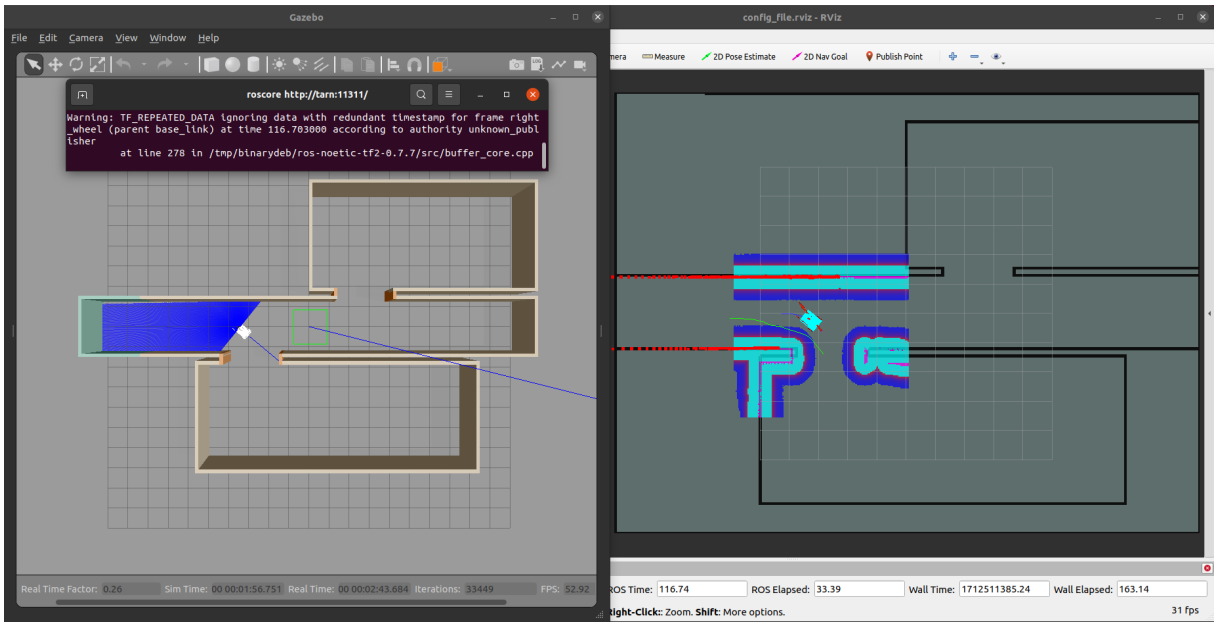
Po dosiahnutí poslednej miestnosti v navigačnej slučke je robot poslaný poslednou navigačnou cestou naspäť na svoju štartovaciu pozíciu. Následne je započaté nové opakovanie navigačnej slučky, ktorá sa má celkovo zopakovať trikrát. Jeden test celkovo obsahuje 12 prechodov medzi poschodiami.

Testy boli vedené za dvoch podmienok, na základe čoho je ich možné rozdeliť na:

- **Sústredené**
  Pred spustením simulácie bol odpojený sekundárny monitor pripojený k notebooku a zminimalizované boli procesy bežiace na pozadí systému Windows a virtualizovaného systému Ubuntu. Počas simulácie bolo v rámci systému Windows aktívne nakliknuté okno klienta VirtualBox, v rámci ktorého bolo aktívne okno systémového terminálu Ubuntu.

- **Nesústredené**
  Pred spustením simulácie nebol kladený dôraz na minimalizáciu procesov bežiacich na pozadí systému Windows a počas priebehu simulácie nebolo aktívne okno s virtualizovaným systémom. VirtualBox klient bol spustený na pozadí, na druhom monitore, kde bolo aktívne okno systémového terminálu Ubuntu.

Kvôli zjednodušeniu vyhodnocovania výsledkov bol pomocou softvéru OBS vedený videozáznam každého z testov. Vyhodnocovanie bolo následne vykonané na základe týchto záznamov. Dokopy bolo vykonaných 23 testov. Na Obrázku č. 4.2. je snímok obrazovky systému Ubuntu zachytený počas chodu jedného zo simulačných testov.



Obrázok č. 4.2: Snímok obrazovky počas simulačného testu

Tabuľky č. 4.1. a č. 4.2. vyjadrujú dáta, ktoré boli získané z niektorých nesústredených a sústredených testov.

Prechod N [s]	**Test 1**	**Test 2**	**Test 3**
Prechod 1	10,37	13,06	14,13
Prechod 2	10,73	15,56	15,97
Prechod 3	10,93	15,53	14,59
Prechod 4	14,49	15,64	14,32
Prechod 5	9,92	16,02	17,75
Prechod 6	15,80	17,80	18,37
Prechod 7	18,01	15,76	19,59
Prechod 8	13,13	15,74	14,13
Prechod 9	12,30	16,12	24.02
Prechod 10	**33,97**	19,74	15,51
Prechod 11	23,07	**83,80**	15,18
Prechod 12	23,67	13,30	19,19
Čas testu [h:min:s]	00:55:09	01:08:50	01:11:14

Tabuľka č. 4.1: Dáta výsledkov z výberu nesústredených testov

Prechod je čas braný od vydania príkazu na ukončenie simulačného sveta, signalizovaného výpisom do konzole po moment, kedy robot dokončil spracovanie navigačného cieľa. To bolo reprezentované zobrazením cesty lokálneho a globálneho plánovača trasy v simulácii a nasledované prvým pohybom robota do nového cieľa.

Čas simulácie je braný od príkazu na spustenie riadiaceho programu v systémovom termináły až po finálne ukončenie simulácie reprezentované výpisom prejdených navigačných ciest a slučiek.

Prechod N [s]	**Test 11**	**Test 20**	**Test 21**	**Test 22**
Prechod 1	8,39	9,41	10,45	9,52
Prechod 2	36,51	10,00	9,93	10,01
Prechod 3	12,20	8,78	9,54	8,77
Prechod 4	13,73	11,07	9,16	9,05
Prechod 5	11,15	11,47	13,88	9,64
Prechod 6	16,09	9,29	9.29	15,34
Prechod 7	8,20	9,14	9,19	9,61
Prechod 8	11,06	20,85	9,96	9,76
Prechod 9	12,03	17,13	21,45	24,27
Prechod 10	11,60	12,43	24,47	14,16
Prechod 11	14,28	18,45	12,49	11,90
Prechod 12	14,80	12,90	18,55	17,68
Čas testu [h:min:s]	00:44:18	00:55:16	00:50:23	00:56:32

Tabuľka č. 4.2: Dáta výsledkov z výberu sústredených testov

Simulačné testy vykonané sústredenou metódou mali vďaka vyššej priorite procesov v systémoch plynulejší priebeh oproti testom z nesústredenej metódy. Pri oboch metódach došlo k občasným výkyvom v časoch prechodov. Ide o prípady, kedy simulačný softvér, obzvlášť Gazebo, predstavoval spomalenie.

V prípade nesústredeného Testu 1, Prechodu 10 trvalo systému dlhšie vypnúť proces klienta Gazebo, alebo v prípade Testu 2, Prechodu 11 bola simulácia spustená, avšak z nezisteného dôvodu softvér RViz dlho spracovával nastavenie pozície URDF modelu. Tieto časové spomalenia v prípade sústredených testov nastali iba raz. Sústredené testy prebehli rýchlejšie aj z dôvodu menej častých momentov, kedy sa robot zdržal v niektorých častiach poschodia, z dôvodu chyby použitého plánovača trasy.

Počas chodu imulačných testov sa real-time factor u Gazeba pohyboval v rozmedzí 0,14 až 0,28 pri premenlivej frekvencii simulácie 29 až 60 FPS. U RViz išlo o frekvencie pod 31 FPS s bežným poklesom blízko k nule. Pri sústredenej metóde sa tieto spomenuté hodnoty držali stabilnejšie blízko horných hraníc.

Simulácie potvrdili funkčnosť navrhnutej metódy hybridnej mapy, avšak o nameraných časoch pri oboch metódach je možné prehlásiť, že sa nejedná o skutočné časy, ktoré by robot v realite potreboval na prechod prostredím a zmenu mapy. Vzhľadom na virtualizáciu systému nebolo možné využiť plný výkon počítača (s procesorom AMD Ryzen 5 5600H a dedikovanou grafickou kartou NVidia RTX 3060). Simulácie neprebehli plynulo a namerané časy nie je možné stotožňovať s realitou.

Dôležité je podotknúť, že časy prechodov boli merané ručne stopkami. Kvôli už vyššie spomenutým obmedzeniam simulácie nebolo možné časy stopovať v rámci programu, pri pokusoch sa časové výstupy merané týmto spôsobom radovo nepriblížovali ručne meraným časom.

V prípade využitia tejto metódy v realite je v otázke prechodového času nutné brať do úvahy dostupný výpočtový výkon, ktorým by bol robot využívajúci tento mechanizmus hybridnej mapy vybavený.

Pri využití navrhnutej metódy na skutočnom robotovi budú spustené len potrebné výpočtový uzly bez využitia simulačného softvéru. V teórii to umožní značne znížiť čas potrebný na zmenu mapy, ktorý je predĺžený o čas potrebný na reštart simulačných klientov. Zároveň je odľahčená požiadavka na výpočtový výkon riadiaceho systému a eliminované sú chyby, ktoré môžu nastať počas reštartu už spomenutých klientov.

Z tohto dôvodu boli vykonané aj testy zmeny mapy bez využitia simulačného softvéru. V .launch súboroch boli zakomentované simulačné uzly. V testovacom programe bol mierne upravený hlavný simulačný cyklus a pridané boli časové oneskorenia kvôli zjednodušeniu stopovania prechodového času. Simulačná navigačná slučka ostala rovnaká.

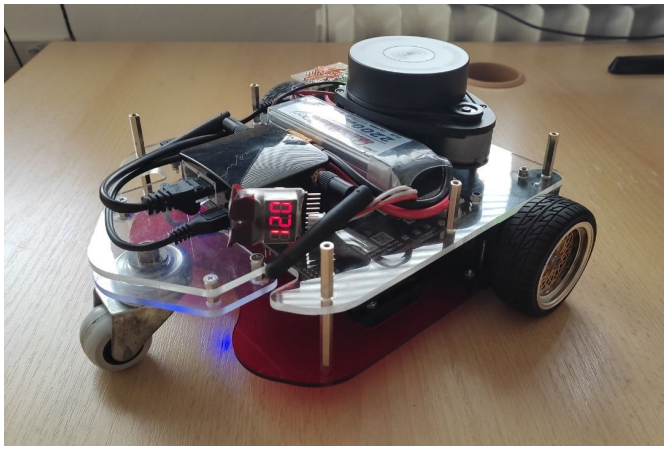
Časy potrebné na reštartovanie uzlov sa pohybovali v rozmedzí dvoch až troch sekúnd. Aby mohli byť celkové časy porovnateľné s časmi z testov so simuláciou, k daným časom je nutné pripočítať ešte ďalšie dve sekundy. Tie sú potrebné pre ďalšiu činnosť spozorovanú počas simulačných testov, napríklad na spracovanie navigačného cieľa. Celkovo sa časy potrebné na zmenu mapy pri týchto testoch pohybujú okolo piatich sekúnd.

5 Testy na robotovi Leela

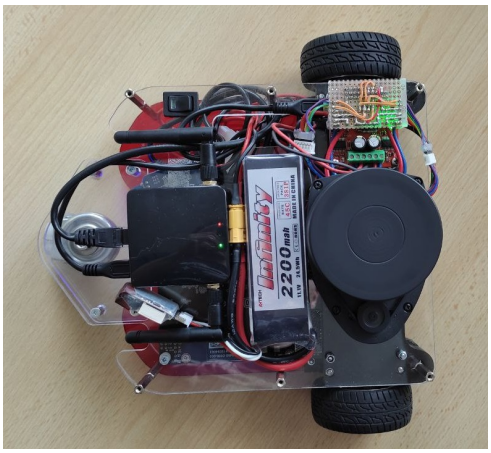
Nad rámec zadania bakalárskej práce bol mechanizmus hybridnej mapy otestovaný na robotovi Leela. Ide o zmenšenú verziu robota Breach určenú na vykonávanie testov.

Jadro robota je tvorené vývojovou doskou Nvidia Jetson Nano s operačným systémom Jetson Linux [42]. Súčasťou vybavenia sú dva DC motory s driverom Roboclaw 2x7A [43] a laserový skener RPLIDAR A1 [44]. Pre prácu s týmito dvomi prvkami boli použité ROS balíky **rplidar_ros** [45] a **roboclaw_node** [46]. Pre bezdrôtovú komunikáciu je súčasťou vybavenia WiFi prístupový bod. Po pripojení naň priamo z terminálu Windows je možné počítačom zadávať riadiace príkazy komunikáciou cez SSH.

Fotky použitej jednotky Leela sú na Obrázku č. 5.1 a č. 5.2.

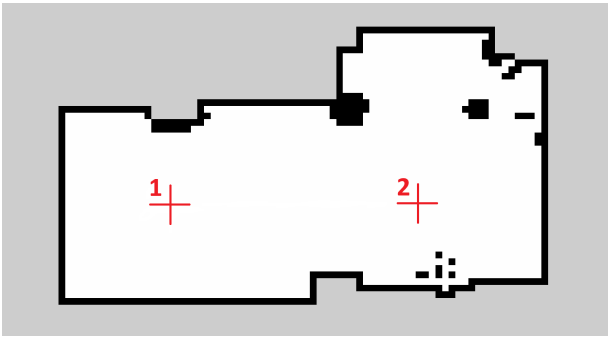


Obrázok č. 5.1: Leela pohľad zboku



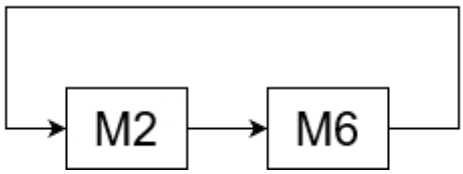
Obrázok č. 5.2: Leela pohľad zhora

Pred vykonaním testov bola otestovaná správna funkčnosť jednotlivých uzlov a následné správanie robota. Pomocou SLAM balíka **gmapping** [47] bola vytvorená testovacia mapa izby. Jej rozmery sú 4000 na 4000 pixelov, ale využitá je len jej malá časť. Rozmerové rozlíšenie bolo nastavené na 0,05 metra na pixel. Mapa je znázornená na Obrázku č. 5.3.



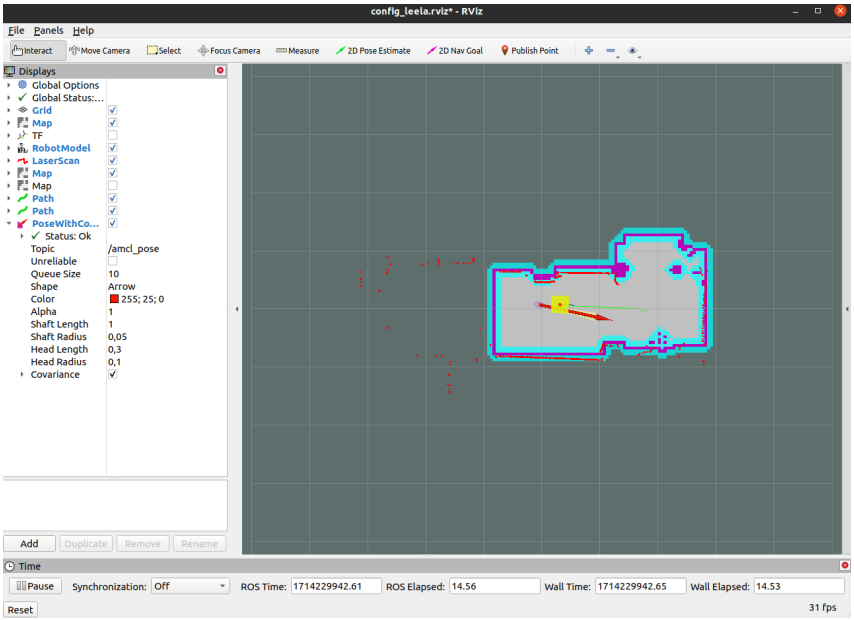
Obrázok č. 5.3: Testovacia mapa izby s vyznačenými navigačnými bodmi

Test pozostával z prechodu robota medzi dvomi pozíciami v miestnosti. Pozícia 1 reprezentovala dve miestnosti s označením M2 a M6, ktoré sú myšlienkovo situované na dvoch rozličných poschodiach. Pozícia 2 reprezentovala výťah, v ktorom nastávala zmena mapy. Počas reštartu je načítaná rovnaká mapa miestnosti, ale zakaždým reprezentujúca príslušné poschodie. Navigačná slučka znázornená na Obrázku č. 5.4. mala byť vykonaná taktiež trikrát.



Obrázok č. 5.4: Navigačná slučka pre testovanie s robotom Leela

Použitý bol rovnaký testovací riadiaci kód, ktorý bol spustený spolu s ostatnými ROS uzlami na spomenutom počítači Jetson. Pred spustením samotného riadiaceho kódu bol najprv spustený samostatný ROS Master. Pri správnej konfigurácii IP adries medzi systémom robota a virtualizovaným systémom Ubuntu je možné vizualizovať chod robota prostredníctvom RViz, znázornený na Obrázku č. 5.5.



Obrázok č. 5.5: Snímka obrázovky počas jedného z testov Leela

Vzhľadom na znížený počet navigačných cieľov je výsledkom jedného testu len šesť prechodov medzi poschodiami. Opäť bola skúmaná spoľahlivosť metódy a časy od začiatku zmeny mapy po prvý pohyb robota. S dostupným výpočtovým výkonom trvala zmena mapy v priemere 8,25 sekundy. V porovnaní s bezsimulačnými testami ide o časy o niečo dlhšie, avšak pri publikovaní desaťkrát väčšej mapy.

Vykonaných bolo 10 testov, počas ktorých robot dokázal pokračovať v navigácii priestorom aj po zmene mapy. Počas každého testu bola každá zmena mapy vykonaná plynule a bez komplikácií.

6 Záver

Bakalárska práca sa venovala návrhu mechanizmu hybridnej mapy pre účely navigácie a lokalizácie mobilného robota vo vnútorných priestoroch viacposchodovej budovy.

Na základe vykonanej rešerše ohľadom aktuálne využívaných spôsobov mapovania v oblasti mobilnej robotiky bol navrhnutý mechanizmus hybridnej mapy, vychádzajúci z myšlienky prepojenia viacerých mriežkových máp reprezentujúcich jednotlivé poschodia budovy pomocou jednej topologickej mapy, vyjadrujúcej vzťahy medzi časťami týchto poschodí.

V rámci riešenia boli navrhnuté mriežkové mapy dvoch poschodí, ktoré boli pomyselne rozdelené na časti podľa miestností, medzi ktorými existujú prechodové zóny vyjadrené navrhnutou topologickou mapou. Topologická mapa prepája mriežkové mapy vyhradenou prechodovou zónou „výťah" situovanou na koncoch chodieb príslušných poschodí.

Pre účely zmeny mapy bol navrhnutý riadiaci program, ktorý pri dosiahnutí prechodovej zóny robotom reštartuje všetky potrebné výpočtové uzly robota. Riadiacemu systému je následne publikovaná nová, aktuálne potrebná mapa pre jeho ďalšiu navigáciu na novom poschodí.

Táto metóda bola overená pomocou simulácie prostredníctvom softvérových balíkov prostredia ROS, menovite RViz a Gazebo. Skúmaná bola spoľahlivosť metódy a čas potrebný na zmenu mapy pri prechode medzi poschodiami. Simulácia i riadiaci program bol spustený v rámci virtualizovaného systému Ubuntu. Pre simulačné účely bol využitý model robota Breach.

Simulačné testy potvrdili vhodnosť využitia navrhnutej metódy hybridnej mapy, aj keď v obmedzenej miere. Výsledky testov poukazujú na premenlivosť v časoch potrebných na reštartovanie uzlov pri prechode poschodiami, spôsobenú chybami reštartu simulačných klientov. Premenlivosť v správaní vychádza priamo zo samotného prístupu k testovaniu, keďže na virtualizovanom systéme nebolo možné využiť plný výkon hardvéru počítača pre chod simulácie.

K eliminácii spomenutého javu boli uskutočnené testy riadiaceho programu iba s potrebnými riadiacimi uzlami robota, bez spustených simulačných klientov. Testy vykazovali minimálne o polovicu kratší čas potrebný pre zmenu mapy oproti najkratším časom pozorovaným pri testoch so simuláciou.

Nad rámec zadania boli vykonané testy na skutočnom robotovi Leela. Ide o zmenšenú verziu robota Breach. Všetok riadiaci kód bol bez simulačného softvéru spustený na riadiacom počítači robota. S dostupným výpočtovým výkonom vykazovali testy časy v okolí ôsmych sekúnd bez zistených chýb počas chodu. Časy potvrdili predpoklad vyvodený z bezsimulačných testov. Funkčnosť metódy bola overená.

Celkovo možno prehlásiť, že ciele bakalárskej práce boli dosiahnuté a výstupom práce je metóda hybridnej mapy vhodná pre účely navigácie a lokalizácie mobilného robota vo vnútorných priestoroch viacposchodovej budovy.

Zdroje

[1] SIEGWART, Roland a NOURBAKHSH, R. Illah. *Introduction to autonomous mobile robots*. Massachusetts: MIT Press, 2004. ISBN 978-0-262-19502-7.

[2] CHIKURTEV, Denis; CHIVAROV, Nayden; CHIVAROV, Stefan a CHIKURTEVA, Ava. Mobile robot localization and navigation using LIDAR and indoor GPS. Online. *IFAC-PapersOnLine*. 2021, roč. 54, č. 13, s. 351-356. ISSN 2405-8963.
Dostupné z: https://doi.org/10.1016/j.ifacol.2021.10.472. [cit. 2024-03-24].

[3] ZHOU, Chengmin; HUANG, Bingding a FRÄNTI, Pasi. A review of motion planning algorithms for intelligent robotics. Online. *ArXiv*. 2021, č. 2102.02376. Dostupné z: https://doi.org/10.48550/arXiv.2102.02376. [cit. 2024-03-08].

[4] THRUN, Sebastian. Learning metric-topological maps for indoor mobile robot navigation. Online. *Artificial Intelligence*. 1998, roč. 99, č. 1, s. 21-71. ISSN 0004-3702. Dostupné z: https://doi.org/10.1016/S0004-3702(97)00078-7. [cit. 2023-10-23].

[5] SONG, Baoye; WANG, Zidong a ZOU, Lei. On Global Smooth Path Planning for Mobile Robots using a Novel Multimodal Delayed PSO Algorithm. Online. *Cognitive Computation*. 2017, roč. 9. Dostupné z: https://doi.org/10.1007/s12559-016-9442-4. [cit. 2024-04-06].

[6] *Approximate Cell Decomposition*. Online. SHENGCHEN, Liu. 2021. Dostupné z: https://shengchenliu.github.io/robotics_gitbook/path_planning/approximate_cell_decomposition.html. [cit. 2024-03-08].

[7] *Exact Cell Decomposition*. Online. SHENGCHEN, Liu. 2021. Dostupné z: https://shengchen-liu.github.io/robotics_gitbook/path_planning/exact_cell_decomposition.html. [cit. 2024-04-08].

[8] HORNUNG, Armin; WURM, Kai; BENNEWITZ, Maren; STACHNISS, Cyrill a BURGARD, Woolfram. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. Online. *Autonomous Robots*. 2013. Dostupné z: https://doi.org/10.1007/s10514-012-9321-0. [cit. 2024-03-18]. Kód dostupný na: https://octomap.github.io.

[9] *Graphs*. Online. Brilliant. Dostupné z: https://brilliant.org/wiki/graphs/. [cit. 2024-04-07].

[10] HAO, Wu; GUO-HUI, Tian; YAN, Li; PENG, Duan a FENG-YU, Duan. Spatial semantic hybrid map building and application of mobile service robot. Online. *Robotics and Autonomous Systems*. 2014, roč. 62, č. 6, s. 923-941. ISSN 0921-8890. Dostupné z: https://doi.org/10.1016/j.robot.2013.01.001. [cit. 2023-07-22].

[11] *Tessellation*. Online. In: Wikipedia: the free encyclopedia. San Francisco (CA): Wikimedia Foundation, 2024. Dostupné z: https://en.wikipedia.org/w/index.php?title=Tessellation&oldid=1220709618. [cit. 2024-04-07].

[12] WEISSTEIN, Eric. *Voronoi Diagram*. Online. Wolfram Mathworld. 2024. Dostupné z: https://mathworld.wolfram.com/VoronoiDiagram.html. [cit. 2024-03-09].

[13] SAFFIOTTI, A a FABRIZI, E. Extracting topology-based maps from gridmaps. Online. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. 2000, roč. 3, s. 2972-2978. Dostupné z: https://doi.org/10.1109/ROBOT.2000.846479. [cit. 2024-01-13]

[14] ZHAO, Weiwei; LIN, Rui; DONG, Shuai a CHENG, Yuhui. A Study of the Global Topological Map Construction Algorithm Based on Grid Map Representation for Multi-robot. Online. *IEEE Transactions on Automation Science and Engineering*. 2023, roč. 20, č. 4, s. 2822-2835. Dostupné z: https://doi.org/10.1109/TASE.2022.3198801. [cit. 2024-01-13].

[15] YAMANAKA, Satoshi a MORIOKA, Kazuyuki. Mobile robot navigation using hybrid simplified map with relationships between places and grid maps. Online. *IFAC Proceedings Volumes*. 2012, roč. 45, č. 22, s. 616-621. Dostupné z: https://doi.org/10.3182/20120905-3-HR-2030.00145. [cit. 2023-07-22].

[16] ROCHA, Francisco; LIMA, Bruno; R, Wilson; ROCHA, Diego; FARIAS, Karoline et al. Machine Learning Applied to Topological Mapping for Structure Recognition. Online. *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2020, s. 1872-1877. Dostupné z: https://doi.org/10.1109/SMC42975.2020.9283475. [cit. 2024-03-09].

[17] THRUN, Sebastian. Robotic Mapping: A Survey. Online. *Exploring Artificial Intelligence in the New Millenium*. 2002. Dostupné z: http://robots.stanford.edu/papers/thrun.mapping-tr.pdf. [cit. 2024-10-23].

[18] DATTALO, Amanda. *ROS Introduction*. Online. Wiki.ros.org. 2018. Dostupné z: https://wiki.ros.org/ROS/Introduction. [cit. 2023-09-15].

[19] LALANCETTE, Chris. *ROS Distributions*. Online. Wiki.ros.org. 2023. Dostupné z: https://wiki.ros.org/Distributions. [cit. 2023-09-15].

[20] VENKATADRI, Arun. *ROS 1 vs ROS 2 What are the Biggest Differences?* Online. Model-Prime. 2023. Dostupné z: https://www.model-prime.com/blog/ros-1-vs-ros-2-what-are-the-biggest-differences. [cit. 2023-09-15].

[21] AZAB, Ammar. *ROS Concepts*. Online. Wiki.ros.org. 2022. Dostupné z: https://wiki.ros.org/ROS/Concepts. [cit. 2023-09-15].

[22]ARSALAN, Anwar. *Part 1: Getting Started with ROS - Overview, Installation and ROS Computational Graph Model*. Online. In: Medium. 2021. Dostupné z: https://medium.com/analytics-vidhya/getting-started-with-ros-overview-installation-and-ros-computational-graph-model-e94d7a16187f. [cit. 2024-04-05].

[23] WOODAL, William. *rviz*. Online. Wiki.ros.org. 2018. Dostupné z: https://wiki.ros.org/rviz. [cit. 2024-03-10].

[24] *RViz User's Guide*. Online. Docs.ros.org. 2012. Dostupné z: https://docs.ros.org/en/indigo/api/rviz/html/user_guide/index.html#. [cit. 2024-03-10].

[25] *Gazebo*. Online. Dostupné z: https://gazebosim.org/home. [cit. 2024-03-10].

[26] *map_server*. Online. Wiki.ros.org. 2020. Dostupné z: https://wiki.ros.org/map_server. [cit. 2024-03-11].

[27] FRAGALE, Nick. *move_base*. Online. Ros.org. 2020. Dostupné z: https://wiki.ros.org/move_base. [cit. 2024-03-11].

[28] *tf2*. Online. WIki.os.org. 2019. Dostupné z: https://wiki.ros.org/tf2. [cit. 2024-04-07].

[29] JAGADEESHAN, S. *joint_state_publisher*. Online. Ros.org. 2022. Dostupné z: https://wiki.ros.org/joint_state_publisher. [cit. 2024-04-07].

[30] *robot_state_publisher*. Online. Wiki.ros.org. 2020. Dostupné z: https://wiki.ros.org/robot_state_publisher. [cit. 2024-04-07].

[31] *amcl*. Online. Wiki.ros.org. 2020. Dostupné z: https://wiki.ros.org/amcl. [cit. 2024-04-07].

[32] PODOLINSKÝ, Ondřej. *Komplexní návrh zalévacího mobilního robotu*. Online, diplomová práca, vedoucí Jiří Krejsa. Brno: Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky, 2022. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/140183. [cit. 2023-07-27].

[33] *Breach*. Online. Benderrobotics. Dostupné z: https://www.benderrobotics.cz/breach.html. [cit. 2024-02-29].

[34] *urdf*. Online. Wiki.ros.org. 2023. Dostupné z: https://wiki.ros.org/urdf. [cit. 2024-03-11].

[35] *How to change map in map_server?* Online. In: Answers.ros.org. 2017. Dostupné z: https://answers.ros.org/question/270129/how-to-change-map-in-map_server/. [cit. 2023-12-11].

[36] *rospy*. Online. Wiki.ros.org. 2017. Dostupné z: https://wiki.ros.org/rospy. [cit. 2024-04-07].

[37]*twist_mux*. Online. Wiki.ros.org. 2018. Dostupné z: https://wiki.ros.org/twist_mux. [cit. 2024-04-07].

[38] TOSSELL, Ken. *nav_msgs*. Online. Wiki.ros.org. 2010. Dostupné z: https://wiki.ros.org/nav_msgs. [cit. 2024-04-07].

[39] MOTA, Jash. *dwa_local_planner*. Online. Wiki.ros.org. 2020. Dostupné z: https://wiki.ros.org/dwa_local_planner. [cit. 2024-03-17].

[40] BELWARIAR, Rachit. *A* Search Algorithm*. Online. Geeksforgeeks.org. 2024. Dostupné z: https://www.geeksforgeeks.org/a-search-algorithm/. [cit. 2024-03-17].

[41]*Breadth First Search or BFS for a Graph*. Online. Geeksforgeeks.org. 2024. Dostupné z: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/. [cit. 2023-12-29].

[42] *Get Started With Jetson Nano Developer Kit*. Online. Developer.nvidia.com. 2024. Dostupné z: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit. [cit. 2024-04-27].

[43] *Roboclaw 2x7A Motor Controller*. Online. Basicmicro.com. Dostupné z: https://www.basicmicro.com/Roboclaw-2x7A-Motor-Controller_p_55.html. [cit. 2024-04-27].

[44] *RPLIDAR A1*. Online. Slamtec.ai. 2024. Dostupné z: https://www.slamtec.ai/product/slamtec-rplidar-a1/. [cit. 2024-04-27].

[45] KANNING, Theo. *rplidar*. Online. Wiki.ros.org. 2019. Dostupné z: https://wiki.ros.org/rplidar. [cit. 2024-04-27].

[46] *roboclaw_ros*. Online. Github.com. 2017. Dostupné z: https://github.com/sonyccd/roboclaw_ros. [cit. 2024-04-27].

[47] *gmapping*. Online. Wiki.ros.org. 2019. Dostupné z: https://wiki.ros.org/gmapping. [cit. 2024-04-27].

Zoznam skratiek

ROS	Robot Operating System
GPS	Global Positioning System
SLAM	Simultaneous Localization and Mapping
ECD	Exact Cell Decomposition
ACD	Approximate Cell Decomposition
DOGMA	Dynamic Occupancy Grid Mapping Algorithm
DDS	Data Distribution Service
API	Application Programming Interface
XML	Extensible Markup Language
LIDAR	Light Detection and Ranging
URDF	Unified Robot Description Format
BFS	Breadth-First Search
AMCL	Adaptive Monte Carlo Localization
DWA	Dynamic Window Approach
OBS	Open Broadcaster Software
SSH	Secure Shell

Digitálna príloha

Príloha bakalárskej práce je tvorená ROS balíkom **tarn_thesis** obsahujúcim jednotlivé zdrojové kódy a súbory použité pri návrhu hybridnej mapy a pri testoch.

Medzi hlavné priečinky a súbory patria:

tarn_thesis

\data		- parametrové súbory uzlov
\launch		
	\simulation.launch	- hlavný simulačný súbor
	\gazebo.launch	- spúšťanie gazebo uzlov
	\navigation.launch	- spúšťanie navigačných uzlov
	\leela.launch	- uzly pre real-life test
\model		
	\breach.urdf.xacro	- URDF model Breach
	\leela.urdf	- URDF model Leela
	\...	- URDF modely výbavy robotov
\rviz		
	\config_sim.rviz	- parametre vizualizácie simulácie
	\config_leela.rviz	- parametre vizualizácie Leela
\world \map		
	\floor1	- prvé poschodie simulácie
	\floor2	- druhé poschodie simulácie
	\real_test	- mapa pre test Leela
\script		
	\testing	- testovacie kódy funkcií
	\physics_parameter.py	- parametre simulácie
	\transform.py	- transformačné vzťahy
	\run_simulation.py	- riadiaci kód simulácie
	\run_leela.py	- riadiace kódy Leela
\...		- ostatné pomocné súbory