SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-100863-98355

IMPLEMENTÁCIA ROS PRE POTREBY RIADENIA ROBOTICKÝCH SYSTÉMOV KUKA

BAKALÁRSKY PROJEKT

Bratislava 2021 **Tomáš Nyiri**

SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-100863-98355

IMPLEMENTÁCIA ROS PRE POTREBY RIADENIA ROBOTICKÝCH SYSTÉMOV KUKA

BAKALÁRSKY PROJEKT

Študijný program:	Robotika a kybernetika
Študijný odbor:	kybernetika
Školiace pracovisko:	Ústav robotiky a kybernetiky
Vedúci záverečnej práce/školiteľ:	prof. Ing. František Duchoň, PhD.
Konzultant: (ak je určený)	Ing. Miroslav Kohút

Bratislava 2021 **Tomáš Nyiri**

STU FEI

ZADANIE BAKALÁRSKEJ PRÁCE

Študent:	**Tomáš Nyiri**
ID študenta:	98355
Študijný program:	robotika a kybernetika
Študijný odbor:	kybernetika
Vedúci práce:	prof. Ing. František Duchoň, PhD.
Konzultant:	Ing. Miroslav Kohút
Miesto vypracovania:	Ústav robotiky a kybernetiky

Názov práce: **Implementácia ROS pre potreby riadenia robotických systémov KUKA**

Jazyk, v ktorom sa práca vypracuje : slovenský jazyk

Špecifikácia zadania:

Robotic Operating System (ROS) je výkonným a pružným rámcom pre písanie algoritmov pre robotické systémy. ROS predstavuje zbierku nástrojov, knižníc a konvencií, ktoré pomáhajú zjednodušiť vytvorenie komplexného a robustného robotického správania sa naprieč širokým spektrom robotických platforiem. V oblasti slovenskej priemyselnej robotiky sú často nasadené robotické systémy od firmy KUKA, ktoré sa uplatňujú pri rôznych výrobách (automobilový, potravinársky, chemický priemysel, a pod.). Cieľom tohto projektu je implementovať do robotického zariadenia vlastné algoritmy riadenia pohybu s využitím ROS.

Úlohy:
1. Analyzujte možnosti nasadenia ROS pre priemyselné roboty KUKA.
2. Preverte funkcionalitu modulov riadenia robotov dostupných v ROSe.
3. Vybrané moduly implementujte vo vybranom simulačnom nástroji (RViz, Gazebo, a pod.)..
4. Verifikujte funkcionalitu vybraných modulov.
5. Spracujte a vyhodnoťte výsledky.
6. Navrhnuté riešenie verifikujte a zdokumentujte v práci.

Dátum zadania:	21. 09. 2020
Dátum odovzdania:	04. 06. 2021

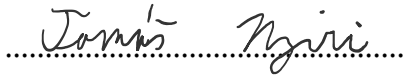
Tomáš Nyiri
študent

prof. Ing. Jarmila Pavlovičová, PhD.
vedúci pracoviska

doc. Ing. Eva Miklovičová, PhD.
garantka študijného programu

Čestné vyhlásenie

Čestne vyhlasujem, že som záverečnú prácu s názvom Implementácia ROS pre potreby riadenia robotických systémov KUKA vypracoval samostatne pod vedením prof. Ing. Františka Duchoňa, PhD. a že som uviedol všetku použitú literatúru.



podpis autora

Poďakovanie

Dovoľujem si poďakovať vedúcemu práce prof. Ing. Františkovi Duchoňovi, PhD. za užitočné pripomienky, ochotu a usmernenie, ktoré som zužitkoval pri písaní bakalárskej práce. Taktiež ďakujem Ing. Miroslavovi Kohútovi za odborné konzultácie potrebné pre vypracovanie praktickej časti bakalárskej práce.

ABSTRAKT

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Robotika a kybernetika
Autor:	Tomáš Nyiri
Bakalárska práca:	Implementácia ROS pre potreby riadenia robotických systémov KUKA
Vedúci záverečnej práce:	prof. Ing. František Duchoň, PhD.
Konzultant:	Ing. Miroslav Kohút
Miesto a rok predloženia práce:	Bratislava 2021

Táto práca sa zameriava na využitie systému Robot Operating System (ROS) pri lokalizovaní a riadení mobilných robotov od spoločnosti KUKA. ROS je výkonný softvér, ktorý obsahuje veľké množstvo nástrojov a knižníc a iných doplnkov, ktoré zabezpečujú vykonávanie aj komplexných robotických úloh. V prvej časti práce sme sa zoznámili s ROS-om, a všetkými odvetviami ktoré nám ponúka. Následne sme preverili použitie tohto softvéru na všetky dostupné roboty KUKA, a opísali sme si ich základné parametre. Oboznámili sme sa s metódami mapovania a lokalizácie, pričom pri prvom mapovaní sme použili teleovládanie robota. Povieme si ako ovládať rameno robota za pomoci nadstavby nástroja v ROS-e. A v poslednom rade si vysvetlíme aké balíčky je potrebné použiť aby sme robota vedeli navigovať v známom prostredí a bez kolízií. V poslednej kapitole sa zameriavame na vykonanie testov s navigovaním robota pričom sa mu snažíme zadávať rôzne navigačné úlohy a ciele, ktoré následne vyhodnocujeme.

Kľúčové slová: ROS, KUKA YouBot model, SLAM GMapping, Gazebo, RVIZ, navigovanie robota, AMCL,

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND IFNORMATION TECHNOLOGY

Study programme:	Robotics and cybernetics
Author:	Tomáš Nyiri
Bakalárska práca:	Implementation of ROS into the Control of Robotic System from KUKA
Supervisor:	prof. Ing. František Duchoň, PhD.
Consultant:	Ing. Miroslav Kohút
Place and year of submission of work:	Bratislava 2021

This work focuses on the use of the Robot Operating System (ROS) in the location and control of mobile robots from KUKA. ROS is a powerful software that contains a large number of tools and libraries and other add-ons that ensure the performance of even complex robotic tasks. In the first part of the work, we got acquainted with ROS and all the industries it offers us. Subsequently, we checked the use of this software on all available KUKA robots, and stated their basic parameters. We got acquainted with the methods of mapping and localization, while in the first mapping we used telecontrollers. We'll talk about how to control the robot arm with the help of a tool superstructure in ROS. And last but not least, explain what packages need to be used so that we can navigate the robot in a familiar environment and without collisions. In the last chapter, we focus on performing tests with navigating robots, while trying to enter various navigation tasks and goals, which we then evaluate.

Keywords: ROS, KUKA YouBot model, SLAM GMapping, Gazebo, RVIZ, robot navigation, AMCL

Obsah

Zoznam obrázkov a tabuliek

Zoznam použitých skratiek

ROS	Robot Operating System
Amr	Autonómny mobilný robot
Amcl	Adaptive Monte Carlo Localization
SLAM	Simultaneous Localization And Mapping
URDF	Unified Robot Description Format
SMPA	Sense, map, plan, act

Úvod

Z dôvodu zvyšovania popularity manipulátorov a mobilných robotov prichádza omnoho väčší dopyt po ich riadení, čo bol hlavný dôvod spracovania našej práce, ktorá sa zameriava na skúmanie možností riadenia a následnej implementácie systému do modelu KUKA robota. Hlavný cieľ našej práce je úspešne navigovať KUKA robota v nástroji ROS, s využitím lokalizačných a riadiacich balíčkov. Hlavný cieľ bolo potrebné rozdeliť do niekoľkých menších cieľov, pričom každá kapitola sa venuje jednému z nich.

V prvej kapitole našej práce sa oboznámime so systémom ROS. Uvedieme ako prebiehal vznik tohto systému zameraného na využitie v robotike a hlavne za akým účelom bol systém vyvíjaný. Taktiež sa zameriame na výhody, ktoré nám táto platforma ponúka, a čo znamená pojem ROS Industrial, následne sa oboznámime s jeho jednotlivými úrovňami ako sú napríklad súborová úroveň, výpočtová úroveň a komunitná úroveň, pričom si bližšie rozoberieme možnosti, ktoré nám každá z daných úrovní ponúka.

Druhá kapitola sa zameriava na nasadenie ROS-u do priemyselných robotov KUKA. V tejto kapitole sa oboznámime so všeobecným pojmom čo je to robot, zameriame sa na delenie priemyselných robotov a následne na podporu robotov KUKA v ROS-e. Podrobnejšie sa zoznámime s dostupnými robotmi, ktoré sú zároveň podporované a uvedieme si aké sú potrebné ovládače na použitie ROS-u s týmito robotmi.

Všeobecným fungovaním lokalizácie a navigácie pri mobilných robotoch sa zaoberá tretia kapitola v našej práci. Zameriame sa na lokalizáciu a tvorenie mapy prostredia pomocou SLAM. Rozdelíme si navigácie na Globálnu a Reaktívnu, a objasníme si, prečo tieto navigácie nemôžu bez seba fungovať. Podrobnejšie si vysvetlíme Globálnu navigáciu, vrátane troch podmnožín, ktoré budeme skúmať z hľadiska mapy prostredia tejto navigácie. Posledná podkapitola je zameraná na lokalizáciu mobilného robota pomocou štatistickej metódy Monte Carlo.

Štvrtá kapitola sa zameriava na model youBota a všetky jeho konfigurácie, pomenovanie kĺbov modelu a taktiež jeho rozmiestnenie v súboroch. V tejto kapitole je bližšie vysvetlený model lidaru, a to konkrétne Hokuyo URG-04 a balíčky v ROS-e, ktoré zabezpečujú teleovládanie robota. Vysvetlíme si ako pracujú balíčky na lokalizáciu

robota ako GMapping alebo AMCL, balíčky, ktoré sú určené na riadia robota, ale taktiež aj balíček, ktorý zabezpečuje ukladanie mapy do súborov a taktiež so všetkými ostatnými nástrojmi využívanými v systéme ROS.

V piatej kapitole sa nachádza vysvetlenie použitia a nastavenia jednotlivých báčikov a nástrojov ROS-u, pričom sme si za pomoci nástroja Gazebo vytvorili vlastný model prostredia, v ktorom celý proces mapovania a navigácie prebiehal.

Posledná kapitola sa zameriava na samotné dosiahnuté výsledky, obsahuje testy, ktoré boli v simulátore vykonané a následne zdokumentované.

1.Charakteristika systému ROS

Pre potrebu implementovať systém ROS do robotických systémov KUKA, je nutné si ROS priblížiť. Opíšeme si za akým úmyslom bol systém vyvíjaný, ako systém pracuje, a z čoho sa skladá.

1.1 História ROS

V roku 2007 dvaja doktorandi Eric Berger a Keenan Wyrobek sa rozhodli vyvinúť vlastný softvér, zameraný na ovládanie robotov, ktorý by bol dostupný na akademickej pôde, pretože si všimli, že ich spolužiakov a kolegov brzdí rozmanitosť robotiky.

Vo svojich prvých krokoch vyvinuli prototyp mobilného robota s názvom PR1 a začali pracovať na softvéri, ktorý by bol vhodný na použitie, pričom využívali svoje skúsenosti a postupy, ktoré nadobudli v predošlých open sorce projektoch.

Po roku 2007 sa výskum ROS-u presunul do Willow Garage. Willow Garage bolo výskumné centrum pre robotiku a technologický inkubátor zameraný na vývoj hardvéru a otvoreného softvéru pre robotiku. Tu začali spolu s tímom technikov vyvíjať robota s názvom PR2, ktorý nadväzoval na PR1 a ROS, ktorý bol jeho riadiaci softvér. V tomto období prispeli do vývoja ROS-u skupiny z viac ako dvadsiatich inštitúcii, a to ako do základného softvéru tak aj s novými balíčkami ktoré rozširovali tento ekosystém.

Prelomovým rokom pre robota PR2 a ROS bol rok 2008, kedy PR2 zvládol počas dvoch dní fungovať na navigácii bez prerušenia. Začiatkom leta 2009 nechali PR2 navigovať v kancelárii, pričom robot si sám otvoril dvere a sám sa zapojil do elektrickej siete. (zásuvky)[20]. V auguste 2009 nasledovalo spustenie stránky ROS.org, pričom prvé návody boli zverejnené v decembri a ROS s verziou 1.0 v januári 2010. Toho roku Willow Garage darovala 11 robotov PR2 akademickým inštitúciám.

Rok 2012 začal vo Willow Garage vytvorením Open Source Robotics Foundation (ďalej už iba OSRF), pričom už v roku 2013 sa stal hlavným správcom softvéru pre ROS. V rokoch keď OSRF prevzal vývoj ROS-u, bola každý rok vydávaná nová verzia[4]

Prvého vesmírneho robota, ktorý bežal na systéme ROS oznámila NASA 1.septembra 2014. Tento robot mal názov Robonaut 2 a nachádzal sa na Medzinárodnej vesmírnej stanici (ISS) do roku 2015, kedy ho po neznámej hardvérovej chybe dopravili naspäť na zem. [26]

1.2 Charakterizácia ROS

V samej podstate je ROS publish-subscribe softvér vytvorený pre použitie v robotike. Z pohľadu užívateľa je ROS sada open-source softvérových nástrojov, ovládačov a knižníc. ROS je vytvorený za účelom zrýchliť a hlavne zjednodušiť vývoj aplikácii zameraných na robotiku. Vieme v ňom nájsť všetky základné prvky, ktoré by sme pri tvorbe robotickej aplikácie potrebovali, ako sú napríklad ovládače pre rôzne typy senzorov, knižnice na spracovanie obrazu, balíčky na plánovanie trajektórii, alebo balíčky podporujúce riešiace rôzne kinematické úlohy. Taktiež sa v ňom nachádzajú knižnice pre mapovanie priestorov až po vizualizáciu a ukladanie nazbieraných dát.

1.3 Výhody systému ROS

ROS, celým názvom Robot operating system má veľkú konkurenčnú výhodu hlavne vďaka tomu, že funguje na princípe opensource software. Tento spôsob fungovania mu zabezpečuje veľkú komunitu angažovaných vedcov a vývojárov, vďaka čomu ho táto výhoda robí príťažlivejším voči konkurenčným robotickým softvérom, ktorým je napríklad software ABB, ktorý je na rozdiel ROS primárne vyvíjaný len pre ABB roboty s uzavretými programami, z čoho vyplýva, že sa u nich stretávame s menším spektrom možností využiteľnosti a pomalšie sa vyvíjajúcimi komponentami. Pre ROS sú typické rýchlo sa vyvíjajúce možnosti a podpora nových typov snímačov a senzorov. Bohužiaľ v ROS môžeme taktiež nájsť určité nedostatky akými sú napríklad nefunkčné riadenie v reálnom čase, ktoré však má byť vyriešené v ROS 2. [31]

1.4 Odvetvie ROS Industrial

ROS- industrial sa zameriava na rôzne typy priemyselných robotov, kde podporu softvéru ROS majú roboty od ABB, Universal Robots, Fanuc a pre našu prácu najdôležitejšie KUKA manipulátory. ROS industrial obsahuje prepojenie s MTConnect bridge, pomocou ktorého vie ROS komunikovať s obrábacími strojmi a zisťovať ich stav. ROS- indrustrial nie je vhodné využívať pre robotov, určených na vykonávanie monotónnych činnosť, z hľadiska zbytočnosti marenia ich potenciálom. Správne sú využívaný v situáciách, v ktorých je potrebné dynamicky generovať trajektóriu pohybu, čiže v prípade komplikovanejšieho pracoviska s viacerými prekážkami alebo presúvateľnými komponentami. Ako príklad v praxi si môžeme uviesť manipuláciu

s neusporiadanými výrobkami, paletizáciu alebo depaletizáciu a následný presun na iné miesto za pomoci robota. [20]

1.5 Distribúcie systému ROS

ROS pravidelne vychádzajú nové verzie systému. Tieto verzie vychádzajú priemerne raz do roka, pričom finálna verzia by sa mala objaviť na ich oficiálnej stránke každý rok v máji. My máme na výber z troch podporovaných verzii a to sú ROS Kinetic Kame, ROS Melodic Morenia a ROS Noetic Ninjemys. V roku 2017 bola vydaná aj verzia ROS Lunar ale stratila podporu v roku 2019.

Tab. 1 Distribúcia verzií systému ROS [18]

Názov verzie	Dátum vydania	Koniec podpory	Prislúchajúca verzia Ubuntu
Noetic Ninjemys	23.5.2020	Máj, 2025	20.04
Melodic Morenia	23.5.2018	Máj, 2023	18.04
Kinetic Kame	23.5.2016	Máj, 2021	16.04
Lunar Loggerhead	23.5.2017	Máj, 2019	17.04
Jade Turtle	23.5.2015	Máj, 2017	15.04
Indigo Igloo	22.7.2014	Apríl, 2019	14.04

V Tab. 1 vidíme zhrnutie posledných piatich vydaných distribúcii ROS-u, pri čom podporu majú zatiaľ prvé tri. Všetky uvedené verzie podporujú starší kompilátor s názvom rosbuild ale aj novší, ktorý sa stáva štandardom v systéme ROS s názvom catkin.

Z dôvodu zamerania našej práce na roboty KUKA, a potrebu podpory ovládačov, sme zistili že kuka_experimental a youbot_driver sú odporúčané pre verziu ROS-u s názvom Kinetic Kame.[18]

1.6 Základné rozdelenie systému ROS

Pre začínajúcich open-sorce programátorov v ROSe je zo začiatku dosť namáhavé sa v prostredí ROS zorientovať a preto je najskôr nutné pochopiť princíp jeho fungovania. Rozoberieme ho do troch navzájom nadväzujúcich levelov, ktorými sú [28]:

1.FileSystem Level (Úroveň súborového systému)

2.Computation Graph Level (Výpočtová úroveň)

3.Ros Comunity level (Komunitná úroveň)

1.6.1 Úroveň súborového systému

Tu sa nachádzajú všetky súbory ktoré súvisia s ROS, a užívateľ ich má uložené na svojom harddisku, okrem inštalácie sa tieto súbory nachádzajú v tkz. workspace, ktorých môže byť ľubovoľné množstvo. Nachádza sa tu :

1.Packages – Hlavná jednotka pre organizáciu. Nachádzajú sa tu všetky dôležité súbory pre chod systému ROS ako sú napríklad zdrojové kódy, knižnice, spúšťacie súbory a ovládače.

2.Package Manifest – uchováva metadáta o danom balíčku vrátane jeho názvu, veľmi stručného popisu, verzie a rôznych iných ktoré sú pre dané spustenie nevyhnutné.

3.Message types (msg)- aj keď v ROS-e je veľa preddefinovaných správ občas je potrebné si vytvoriť vlastný typ správy je nutné zadefinovať ich štruktúru. Na to nám slúži táto časť súborového systému.

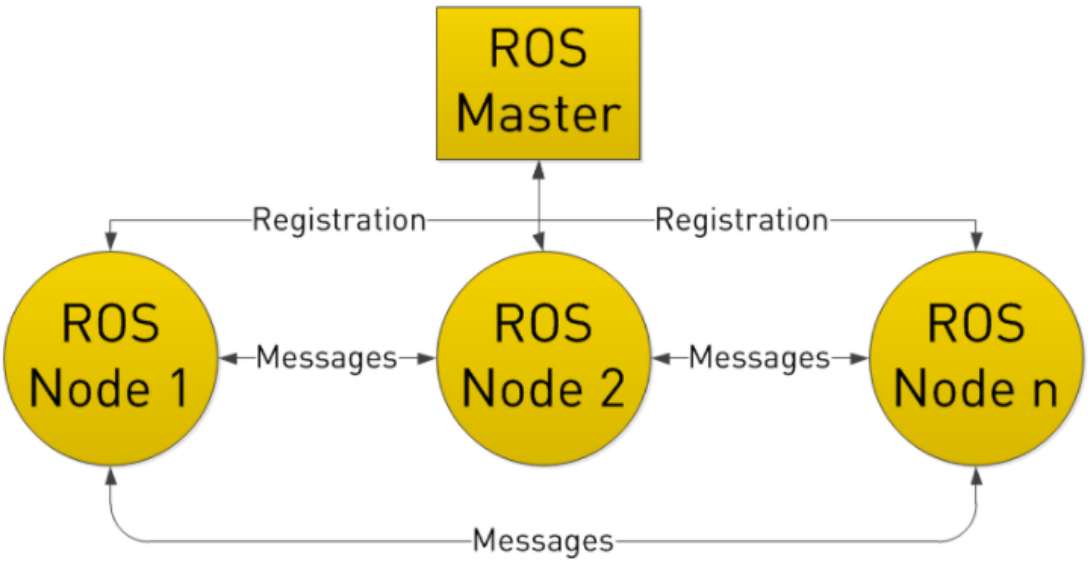
4.Services types- Tu sa nachádzajú preddefinované interakcie na požadované služby ktoré prebiehajú medzi procesmi.[31]

1.6.2 Výpočtová úroveň

Comuputating Graph Level je tiež známy pod názvom Runtime level, ktorý môžeme taktiež nájsť spomínaný v odbornej literatúre. Nachádzajú sa v ňom jednotlivé navzájom prepojené časti, ktorými sú:

1. Master- je v ROSe jadrom každej aplikácie alebo napísaného programu , pretože Master má za úlohu prepojiť medzi sebou všetky uzly, a všetky komunikačné kanály ktoré sú vytvorené. Jednoduchšie povedané ak chcú uzly medzi sebou komunikovať musia najskôr o sebe vedieť a to je úlohou Mastra, vieme ho zavolať jednoduchým príkazom „*roscore*“.
2. Uzol(„Node“)- predstavenie proces, ktorý vykonáva výpočtové operácie, pre užívateľa je ľubovoľné koľko procesov naraz spustí, a akú komunikáciu si zadefinuje v programovacom jazyku C++ alebo Pythone. V rámci ROSu beží ľubovoľné množstvo uzlov paralelne a komunikujú medzi sebou. Pre príklad si môžeme uviesť rôzne tri uzly pričom prvý spracováva dáta z kamery, druhý lokalizáciu a tretí pohyb robota.

3. Správy(„Messages“) - sú štruktúry ktoré sú naplnené komunikačnými dátami, preposielajú si ich uzly za pomoci tém.
4. Téma(„Topics“) – Uzly dokážu medzi sebou komunikovať aj pomocou tém. Z pohľadu tém sa uzly delia na subscriber-a publisher-a pričom jeden uzol danú tému vytvorí (publisher) a ďalší uzol túto tému číta (subscriber), napríklad z pohľadu slam_gmapping, číta tému /tf, a publikuje tému /map.
5. Služby(„Services“) – Služby sú tiež jedným druhom komunikácie ktorá sa nachádza v ROS-e, z pohľadu tejto komunikácie sa uzly delia na server a klienta. Komunikácia funguje tak že klient pošle požiadavku na server a server mu odpovie, pričom sa využívajú rovnaké dátové typy ako používajú správy.
6. Bags – Občas pri ladení aplikácii, nemáme pri sebe vždy reálneho robota, a na túto maličkosť myslia aj bagy, v ktorých si vieme uložiť reálne hodnoty zo senzorov z robota a pracovať s nimi aj v rámci simulácie. [31]



Obr. 1 Schéma 1 [11]

1.6.3 Komunitná úroveň

Ros má organizáciu, ktorá spravuje kľúčové komponenty systému a jednotlivé distribúcie, no aj tak väčšinu balíčkov a obsahu vytvorila práve robotická komunita ktorá využíva overené zdroje medzi ktoré patria:

Distribúcie- Organizácia spravujúca kľúčové komponenty vydáva každý rok v máji nové distribúcie ROSu pre jeho správny chod.

ROS Wiki – wikipédia písaná komunitou, vieme tu nájsť potrebné dokumentácie k balíčkom

ROS Answers - Q&A fórum pre všetky otázky, je to ROS verzia Stack Overflowu

ROS Blog – Zdroje noviniek o vývoji ROS

2. Nasadenie ROS pre priemyselné roboty KUKA

Pre potreby nasadenia ROS do systémov KUKA je potrebný výber robota, preto je táto kapitola práce zameraná na opis všetkých dostupných KUKA robotov, ktoré podporuje ROS.

2.1 Charakteristika robota

Robot je zariadenie, ktoré je automatizované a vie snímať podnety z okolia a spätne reagovať na ne. Je to vlastne počítačom riadený systém, ktorý je schopný autonómnej a cieľovo orientovanej interakcie buď so známym prostredím alebo prostredím, v ktorom sa vie orientovať po jeho spracovaní za pomoci senzorov (napr. lidar). Interakcia robota priamo spočíva vo vnímaní okolitého prostredia, v jeho rozpoznávaní, a v pohybe po ňom alebo v manipulácii rôznych predmetov bez akejkoľvek kolízie. My sa budeme snažiť o podobný výsledok, aby sa mobilný robot pohyboval v známom prostredí bez kolízie.

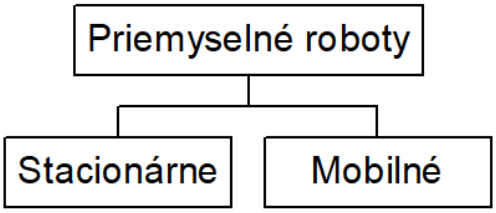
Robotov vieme podľa mobility rozdeliť na stacionárne, mobilné a lietajúce. V oblasti použitia vieme nájsť robotov v rôznych sektoroch ako sú priemyselné roboty, alebo zdravotnícke, armádne atď.

Spoločnosť KUKA sa zaoberá ako vývojom robotov tak aj ich automatizáciou. Nás od tejto spoločnosti budú zaujímať priemyselné roboty a mobilné roboty spoločnosti KUKA.

2.2 Priemyselné roboty

Priemyselný robot, alebo manipulačný priemyselný robot (v skratke manipulátor) je automaticky riadený viacúčelový stroj. Je programovateľný, čiže vie zmeniť svoj program podľa činnosti, ktorú má práve vo výrobe vykonávať. Táto činnosť musí byť stála pri danej výrobe, inak sa daný robot stáva neefektívnym. Ale pri zmene výroby vieme daného robota preprogramovať. Na danú prácu roboty využívajú nástroje rôzneho druhu od zváracích nástrojov cez montážne až po lakovacie. Veľkou výhodou priemyselných robotov v priemysle je ich vytrvalosť, presnosť a rýchlosť, avšak všetky tieto vlastnosti treba vedieť u priemyselného robota aj využiť.

Priemyselné roboty sa delia podľa kinematických štruktúr na KĺbovéKarteziánske, SCARA roboty a Paralelné roboty. Pre nás sú však najviac príťažlivé kĺbové roboty, pretože spoločnosť KUKA má rozsiahli sortiment priemyselných kĺbových robotov. Nájdeme tam napríklad KR 3 AGILUS ktorý má maximálne zaťaženie 3kg a maximálny dosah 541 mm ale aj KR 700 PA ktorého maximálne zaťaženie sa šplhá na 700kg a maximálny dosah na 3320 mm.



Obr. 2 Rozdelenie priemyselných robotov podľa schopnosti premiestňovania

2.3 Mobilné roboty

Mobilné roboty sú schopné sa pohybovať vo svojom prostredí, t. j. nie sú fixované na jednom fyzickom mieste ako statické roboty. Mobilné roboty môžu byť autonómne navigované čo znamená, že sú schopné sa navigovať v priestore za pomoci kamier alebo rôznych senzorov. Mobilné roboty nemusia byť len autonómne a teda pohybovať sa rôzne po priestore, môžu sa pohybovať aj po vopred daných trajektóriách. Od spoločnosti KUKA je nám známy mobilný robot s názvom KUKA youBot, ktorý má umiestnené rameno na mobilnej základni.

2.4 Podpora pre roboty KUKA v prostredí ROS

Roboty od spoločnosti KUKA majú v systéme ROS, vytvorené špeciálne ovládače ktoré prislúchajú k jednotlivým robotom. My sa pozrieme aké jednotlivé roboty od spoločnosti KUKA tieto ovládače podporujú a vysvetlíme si ako pracujú.

2.4.1 Ovládač KUKA_experimental

V prostredí ROS existuje priama podpora pre roboty KUKA. Táto podpora je v podaní zdrojového balíku s názvom kuka_experimental, ktorý sa doinštaluje do ROSu. Pri sťahovaní tohto balíčku máme veľké upozornenie že je stále vo vývoji, čo pre nás znamená, že sa v balíčku nachádzajú známe chyby a neodporúča sa použiť vo výrobe, čo pre nás neznamená problém. Balíček obsahuje podporu pre 8 robotov KUKA.

V balíku sa nachádza podpora pre nasledujúce KUKA roboty :

- KR10
- KR120
- KR150
- KR16
- KR210
- KR5
- KR6
- LBR IIWA

Pre našu prácu sú z daného balíčku zaujímavé iba modely KR6 a LBR IIWA, z dôvodu dostupnosti fyzických manipulátorov.

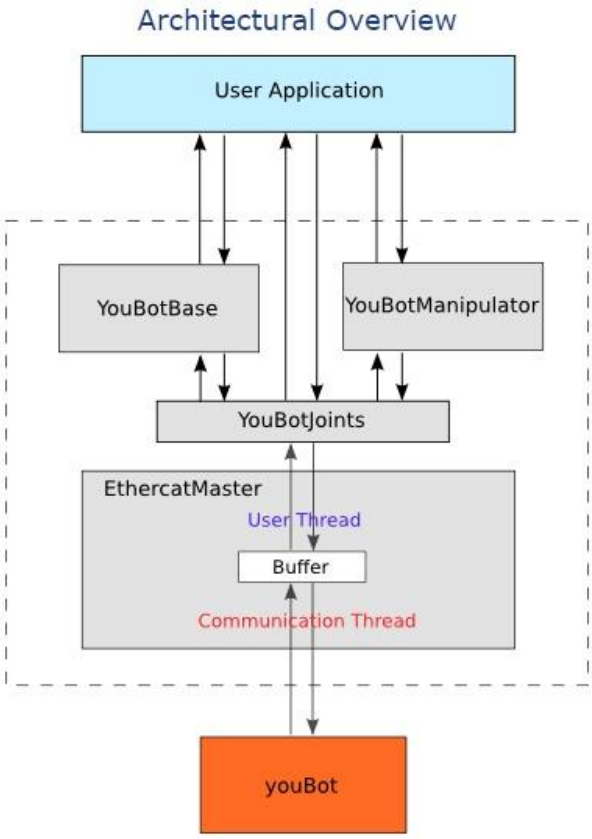
Avšak máme na výber ešte jeden model robota ktorým je Kuka Youbot. Youbot je mobilný robot s robotickým ramenom. Balíček kuka_experimental však tohto robota nepodporuje, preto sme museli preskúmať iné riešenie, ktorým je osobitný ovládač s názvom 2.4.2 Ovládač youbot_driver. [9]

2.4.2 Ovládač youbot_driver

YouBot driver je základný ovládač pre komunikáciu s Youbotom. YouBot_driver slúži na priame ovládanie Youbota, ako ramena tak aj podvozku z čoho nám plynie, že daný ovládač bude v najnižšej vrstve v hierarchii riadiacich softvérov robota. Spomínaný ovládač je napísaný v programovacom jazyku C++. Youbot driver komunikuje výhradne s balíčkom youbot_oodl.

Aby sme sa v tom vedeli lepšie zorientovať musíme si rozdeliť robota do troch pod sekcií alebo tried, ktorými sú :

- YouBotManipulator - reprezentuje rameno robota ako agregáciu niekoľkých kĺbov
- YouBotBase – predstavuje základnú platformu youbota
- YouBotJoint – zastupuje kĺby ako v ramene tak aj v základni robota. Pod kĺby sa zaraďujú aj samotné kolesá robota. [12]



Obr. 3 Komunikácia medzi triedami ovládača [14]

Obrázok znázorňuje komunikáciu medzi jednotlivými triedami, aplikáciou a samotným robotom. Ako si môžeme všimnúť, samotná komunikácia je sprostredkovaná ešte jednou triedou EthercatMaster, ktorá tvorí samostatné komunikačné vlákno. Prenos dát cez toto vlákno sa vykonáva pomocou neblokovanej medzi pamäte. EtherCAT je vlastne internet, ktorý má štandardizovaný protokol v IEC 61158 a je vhodný pre použitie ako nástroj na komunikáciu medzi robotom a riadiacim počítačom.[12]

2.4.3 Balík Yboubot_oodl

Tento balík primárne komunikuje s youbot ovládačom pomocou spoločného uzla s názvom youbot_wrapper, ktorý obsahuje rôzne uzly, ako prijímacie tak aj odosielacie pre správnu komunikáciu s youbot ovládačom. Taktiež obsahuje softvérové núdzové zastavenie robota a bezpečnostný uzol, ktorý kontroluje rýchlosť robota.

Tento uzol sa nazýva cmd_vel_safety a číta všetky správy, ktoré riadia pohyb robota, pričom používa knižnicu tf na nájdenie polohy robota a pomocou mriežky si vypočítava nasledujúcu polohu robota. Ak by sa táto poloha náhodou zhodovala s polohou nejakej prekážky, tak sa príkazová rýchlosť robota spomalí.

Medzi uzly ktoré prijímajú správy patria uzly o pohybe kĺbov, kolies, a uchopovača.[13]

2.5 Dostupné roboty KUKA

Pre úspešnú implementáciu programovateľného kódu do robota KUKA je potrebný samotný robot. Na Fakulte elektrotechniky a informatiky Slovenskej technickej univerzity v Bratislave máme dostupné 3 roboty KUKA, ktorými sú:

Stacionárne roboty :

- KR6
- LBR IIWA,

Mobilný robot :

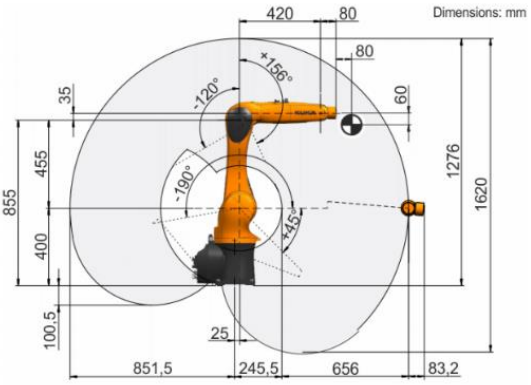
- KUKA youBot

2.5.1 KUKA KR6

KUKA KR6 je stacionárny robot, ktorý má šesť stupňov voľnosti, pričom jeho maximálny dosah činí 901.5mm a jeho maximálne zaťaženie je 6kg. Pre toto zaťaženie je však potrebné optimalizovať dynamický výkon robota alebo znížiť ťažisko vzdialenosti, kedy je možné použiť vyššie zaťaženie až do hodnoty maximálneho zaťaženia.

Tento priemyselný manipulátor disponuje odolnosťou IP54, čo je čiastočná ochrana pred prachom a ochrana pred striekajúcou vodou z každého smeru. Do koncového bodu robota sa dajú umiestniť rozličné nástroje pre rôzne použitia robota.

Balík kuka_experimental podporuje všetky roboty KR 6 a aj ich varianty, pričom tento balík obsahuje všetky konfiguračné údaje aj s 3D modelmi robotov. Limity spojov a maximálne zaťaženie vychádzajú z priamych informácii uvedených v dokumentácii KUKA, ktoré sú uvedené na Obr. 4. [6]

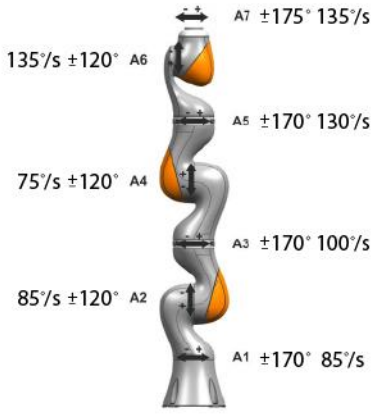


Obr. 4 Limity KR6[6]

2.5.2 KUKA LBR IIWA

LBR IIWA je sériovo vyrábaný senzitívny stacionárny robot a je vhodný pre spoluprácu človeka s robotom. V preklade LBR znamená „Leichtbaurobote“ čo je v preklade robot s ľahkou konštrukciou a IIWA je skratka pre „intelligent industrial work assistant“. Tento robot je vhodný pre veľmi citlivé úlohy, pričom LBR IIWA existuje v dvoch vyhotoveniach so zaťaženiami 7 a 14 kilogramov. Pri práci s človek je potrebné, aby robot spĺňal aj normy, ktoré takúto prácu dovoľujú. IIWA spĺňa požiadavky normy ISO 13849, ktorá informuje o zabezpečení bezpečnej kooperácie človek-robot.

Veľkou výhodou LBR IIWA je, že má 7 stupňov voľnosti s maximálnym dosahom 800 mm a maximálnym zaťažením 7 kg, pričom dokáže pracovať s presnosťou ± 0.1mm. Taktiež disponuje krytím IP54 čo pre neho znamená čiastočnú ochranu pred prachom a striekajúcou vodou z každého smeru. Hmotnosť robota je takmer 24kg.
Balík kuka_experimental podporuje taktiež obe varianty robotov LBR IIWA, pričom balík obsahuje všetky konfiguračné údaje aj s 3D modelmi robota a spúšťacími súbormi. Limity spojov a maximálne zaťaženie vychádza taktiež z priamych informácii uvedených v dokumentácii KUKA obr. nižšie.[7]

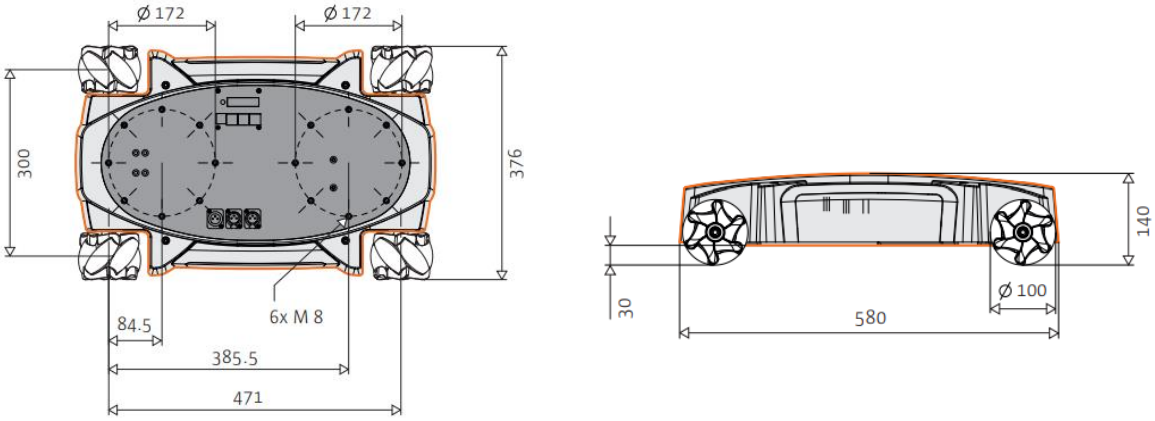


Obr. 5 Limity kĺbov robota LBR IIWA[8]

2.5.3 KUKA youBot

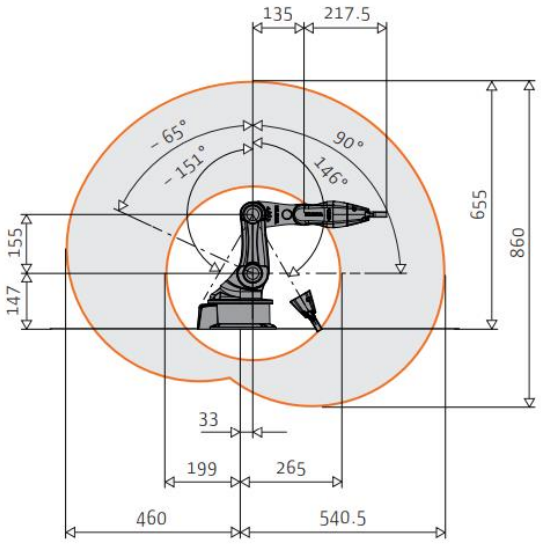
Obr. 6 Kuka youBot[10]

Kuka youBot je mobilný robot, ktorého rameno je umiestnené na všestrannej základni, obsahujúcej 4 všesmerové kolesá, inak nazývané aj Švédske kolesá, ktoré umožňujú robotovi pohyb do všetkých smerov bez mechanického riadenia. Samotná základňa má hmotnosť 20 kilogramov. Nachádza sa v nej 24V baterka s kapacitou 5Ah, pričom dokáže dodávať napätie až 200W. Približná výdrž batérie je asi 1 hodina a 30 minút. Základňa sa vie pohybovať rýchlosťou 0.8 m/s.



Obr. 7 Kuka youBot rozmery základne [10]

Rameno youBota disponuje piatimi osami, čo znamená, že má 5 stupňov voľnosti. Jeho výška je 655 mm pričom jeho maximálny dosah je 540 mm. Rameno dokáže zdvihnúť teleso o hmotnosti maximálne 0.5 kg. Na konci ramena je koncova v podobe paralelného uchopovača s dvoma prstami.



Obr. 8 Kuka YouBot limity ramena [10]

Nevyhnutnosťou mobilného robota je vstavaný riadiaci počítač priamo v základni robota. Počítač obsahuje procesor Intel Atom Dual Core s frekvenciou 1.66GHz, RAM o veľkosti 2GB a externý SSD disk s veľkosťou 32gb. Vstupné porty sú 6x USB typu 2.0, VGA port, ktorý slúži na pripojenie externej obrazovky, a vstup pre LAN. Počítač je napájaný 12V jednosmerného napätia.

YouBot bol primárne navrhnutý na výučbu, výskum a experimentovanie. Používa sa na univerzitách a vysokých školách, kde slúži na vývoj a validáciu nových algoritmov pre mobilné roboty, ktoré budú patriť do tovární zajtrajška, čo bol hlavný dôvod nášho rozhodnutia využiť softvérový model spomínaného robota na demonštráciu navigačných a lokalizačných softwarových balíčkov v ROSe.[30]

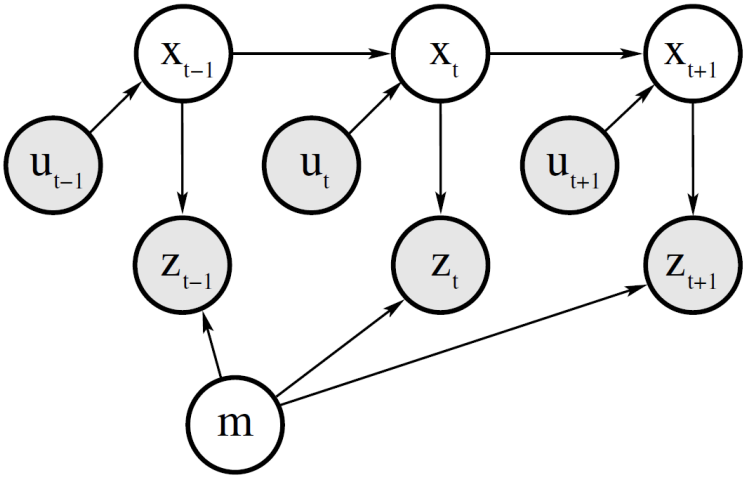
Obr. 9 Kuka youBot rameno[10]

3. Lokalizácia, navigácia a mapovanie

Na lokalizovanie mobilných robotov v známom aj neznámom prostredí vieme využívať metódy ako SLAM a Monte Carlo. Pre riadenie robota využijeme spoluprácu dvoch navigácií, ktorými je globálna a reaktívna navigácia.

3.1 Metóda SLAM

Skratka SLAM, v plnom znení „*Simultaneous Localization And Mapping,*“ sa zameriava na problém navigácie robota v neznámom prostredí, pomocou určenia jeho aktuálnej polohy, ktorej hodnoty sú získané zo snímačov, a pomocou 2D lasera , ktorý na základe vyhodnocovania vyhotovuje priechodnosť terénu. Tento krok možno vynechať v prípade, že už mal vyhotovenú mapu prostredia pomocou ktorej vieme robota navigovať v teda už známom prostredí. Táto mapa môže byť dodatočne upravovaná.[15]



Obr. 10 Grafický model úloh SLAM

Grafický model na Obr. 10 označuje vzťahy medzi jednotlivými uzlami, pričom tmavé uzly sú známe premenné. Označenie U nám reprezentuje odometriu robota a Z dáta zo senzora (napríklad lidar alebo sonar.) Biele sú neznáme premenné pričom X reprezentuje polohu robota a M mapu prostredia. Premenná t nám udáva čas zaznamenania dát, pričom postupnosť zaznamenaných dát môže byť až t_{+n}. Od týchto dát sa potom odvíja výsledná mapa prostredia. [15]

3.2 Navigácia mobilných robotov

Z dôvodu nasadenia mobilných robotov v praxi, vznikla požiadavka na riadení mobilných robotov v známom alebo neznámom prostredí. Z tohto pohľadu sú vyvíjané rôzne algoritmy, ktoré zabezpečujú optimálnu trasu robota, čo môže pozitívne ovplyvniť dĺžka dráhy a zlepšiť spotreba energie na danom úseku.

Navigácie vieme rozdeliť do dvoch nesledujúcich skupín :

- Reaktívna navigácia
- Globálna navigácia

Počas premiestňovania sa robota do cieľovej polohy robot musí reagovať na skutočnosti, ktoré nemusia byť predpovedané, aby bol schopný dosiahnuť cieľ. Ako príklad si môžeme uviesť situáciu, kedy sa objaví prekážka, ktorá nie je na globálnej mape uvedená. Práve preto sa globálna navigácia neuplatní bez reaktívnej navigácie a je potrebné ich vzájomné použitie. Avšak ak by sme použili iba reaktívnu navigáciu robot by nevedel dosiahnuť vzdialený cieľ.[1]

3.2.1 Globálna navigácia

Zo všeobecného hľadiska hlavným cieľom globálnej navigácie je nájdenie optimálnej dráhy medzi aktuálnou polohou robota a cieľovým bodom, pričom by mali byť splnené nejaké podmienky. V týchto podmienkach sa nachádza minimalizácia alebo maximalizácia dĺžky trasy, čas kedy je mobilný robot v pohybe a iné.

Globálna navigácia pracuje s troma základnými odpoveďami na nasledujúce tri otázky „Ako sa tam dostanem ?," „Kde som sa už nachádzal ?", „Aká je pre mňa najlepšia cesta do cieľa ?"

Globálnu navigáciu vieme rozdeliť na tri podmnožiny, ktoré sa delia na základe mapy prostredia[1] :

- **Na báze topologickej mapy** – navigácia na tejto báze je najviac podobná navigácii, ktorú používajú ľudia(v konverzáciách najviac) na navigovanie sa navzájom (choď rovno a zaboč doprava pri piatom stĺpe). Navigácia na tejto báze využíva vyznačené body v prostredí, pričom samotné plánovanie trajektórie spočíva v nájdení prepojenia štartu a cieľa na danej mape. Jej veľkou výhodou je nízka náročnosť na výpočtový výkon a pamäť. Pri plánovaní trasy sa využívajú algoritmy Dijkstra alebo A* algoritmus[1].
- **Na báze metrickej mapy** - ak by sme porovnali podmnožinu na základe

topologickej mapy a metrickej mapy, tak druhá spomínaná podmnožina má rovnaký zmysel a tým je nájdenie najlepšej trajektórie do cieľa, avšak ich hlavným rozdielom je uplatnenie rozličných metód, ktoré vykresľujú podľa zvolených kritérií optimálnu trajektóriu. Ďalším rozdielom pri navigácia založej na báze metrickej mapy, je že rozdeľuje celkovú trasu do viacerých bodov s jasne danými súradnicami, čiže výsledná trajektória sa skladá z niekoľkých cieľov. Jej výhodou je jej opísanie prostredia pomocou metrických vlastností, avšak hľadanie optimálnej trasy je výpočtovo náročné. Využívajú sa tu algoritmy ako Ohňový algoritmus alebo D* algoritmus[1].

- **Na báze geometrickej mapy**- táto množina má svoje uplatnenie aj vďaka geometrickej reprezentácii prostredia. Jej využitie je dosť náročné z pohľadu vykreslenia mapy, pretože každý objekt musí mať presne stanovený tvar a polohu, kde sa daný objekt nachádza, pričom pri tvorení tejto mapy je potrebné použiť presné snímače (napr. Hokuyo, Lidar). V tejto podmnožine globálnej navigácie sa využíva napríklad Graf viditeľnosti, Dotyčnicový graf a iné.[1]

3.2.2 Reaktívna navigácia

Cieľom reaktívnej navigácie, inak nazvaná aj lokálna navigácia je riadenie robota tak, aby robot pri prechádzaní prostredím nenarazil do prekážok. Reaktívna navigácia funguje na základe teórie SMPA, čo v preklade znamená snímaj, mapuj, plánuj a konaj. Teória SMPA je dôležitá hneď z dvoch hľadísk, a tým je zabezpečenie, aby robot v prostredí neprišiel do kolízie s prekážkou, a aby lokálna navigácia fungovala v súčinnosti s globálnou navigáciou.

Reaktívna navigácia je na výpočet veľmi jednoduchá z dôvodu, že všetky výpočty musia byť počítané v reálnom čase, aby výpočet vytváral najmenšiu záťaž na pamäť a výpočtový výkon.[1]

3.3 Lokalizácia Monte Carlo

Lokalizácia Monte Carlo sa radí medzi klasické štatistické metódy. Aby sme vedeli použiť tieto metódy je potrebná znalosť prostredia, a to znamená, že musíme mať vyhotovenú mapu, ktorú použijeme pri lokalizácií.

Pri spustení lokalizácie je do roviny umiestnených niekoľko možných pozícii robota, pričom sa ešte nevie, ktorá je správna. Následným hýbaním robota a zbieraním dát zo senzorov (napríklad LIDAR), sa pravdepodobnosť každej možnej pozície upravuje

pomocou štatistického modelu daných snímačov. Za správnu pozíciu sa následne považuje tá pozícia s najvyššou pravdepodobnosťou.[1]

Lokalizáciu Monte Carlo využijeme pri za pomoci uzla AMCL, čo znamená adaptívna Monte Carlo lokalizácia. Budeme pomocou nej lokalizovať robota pri navigovaní v známom priestore a overíme funkčnosť tejto štatistickej metódy použitej v systéme ROS.

4. Poznatky o použitých balíčkoch a modeloch v simulácií

V nasledujúcich pod sekciách si priblížime všetky softvérové súčasti, ktoré využijeme pre zabezpečenie správneho lokalizovania a riadenia robota. Zameriame sa aj na softvérové súčasti, ktoré zabezpečujú mapovanie prostredia. Taktiež si predstavíme model youBota a Hokuyo senzora.

4.1 Použité modely v simulácii

V simulácií sme potrebovali využiť kópie reálnych modelov, aby sme sa dokázali priblížiť reálnym podmienkam. Týmito modelmi sú:

- KUKA youBot
- Hokuyo URG-04

4.1.1 KUKA youBot model

Pre simuláciu potrebujeme do prostredia ROS nahrať model robota youBot. Daný model je veľmi podobný reálnemu robotovi až na malé rozdiely. Jedným z nich sú kolesá v tvare štyroch bielych gulí, ktoré reprezentujú všesmerové Švédske kolesá a vpredu na základni je pripevnený 2D laserový senzor.

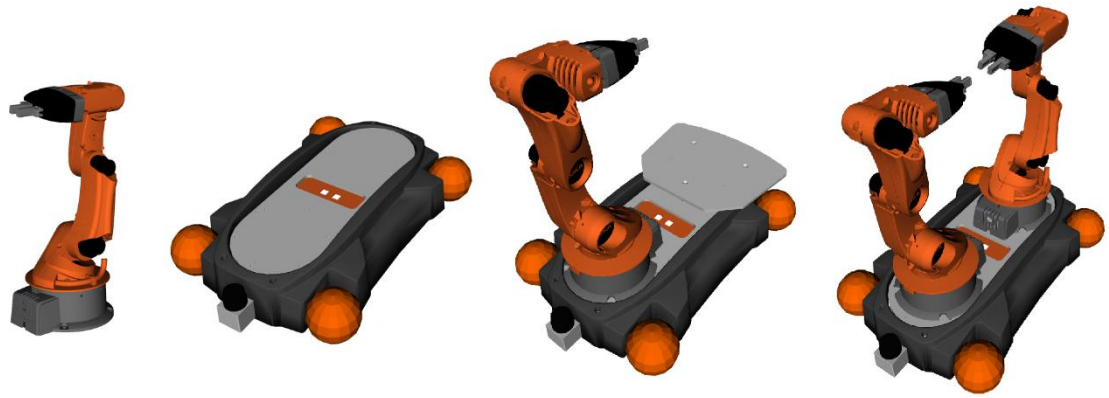
Obr. 11 3D Model robota youBot

Podľa potreby využitia je možné si zvoliť, akú konfiguráciu youBota budeme v simulácii používať, pričom je možné použitie viacerých simulácii súčasne.

Možné konfigurácie robota sú :

- iba rameno youBota
- iba základňa youBota
- základňa s ramenom youBota (štandardná konfigurácia)
- základňa s dvoma ramenami youBota

Bližšia vizualizácia ako môže robot vyzerať v rôznych konfiguráciách je zobrazená na obr. 12.



Obr. 12 Modifikácie modelu robota youBot

Model youBota sa nachádza v priečinku youbot_description. Ak si otvoríme modelový súbor ľubovoľnej konfigurácie youBota všimneme si, že youBot je definovaný rôznymi časťami, ktoré nám spolu dávajú jeden celistvý model robota.

Jednotlivé časti robota sú :

- youbot_base, t. j. definované rozmery základne, a jej pohyblivé časti (všesmerové kolesá)
- youbot_arm, tu je definované rameno robota, avšak nenachádza sa tu uchopovač ktorý je definovaný v osobitnom súbore
- youbot_gripper je definovaný uchopovač, ktorý sa nachádza na konci ramena
- youbot_plate, t. j. definícia rovinnej podložky, ktorá sa nachádza na zadnej časti youbota.
- sensors, v tejto položke je predvolený snímač Hokuyo URG04, avšak v prípade potreby je možné zmeniť predvoľbu napríklad na Microsoft lifecam alebo Kinect kameru.

- ros_controller
- materials, v materiáloch sú definované všetky farby, ktoré sú použité v modely.

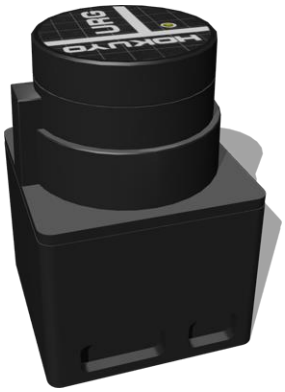
Pre lepšie pracovanie s modelom je taktiež potrebná znalosť mien všetkých kĺbov a kolies, ktoré sa na modeli robota nachádzajú, pričom pri všesmerových kolesách je anglická skratka na rozpoznanie o ktoré koleso sa jedná. (fl- front left, br- back right) pri kĺboch manipulátora to už nie je také zrejmé, preto si treba uvedomiť že dané kĺby sú očíslované od základne smerom k uchopovaču.[30]

Tab. 2 Názvy kĺbov v ROS-e [30]

Kĺb	Meno kĺbu
Všesmerové koleso predné ľavé	wheel_joint_fl
Všesmerové koleso predné pravé	wheel_joint_fr
Všesmerové koleso zadné ľavé	wheel_joint_bl
Všesmerové koleso zadné pravé	wheel_joint_br
Kĺb na manipulátore č.1	arm_joint_1
Kĺb na manipulátore č.2	arm_joint_2
Kĺb na manipulátore č.3	arm_joint_3
Kĺb na manipulátore č.4	arm_joint_4
Kĺb na manipulátore č.5	arm_joint_5
Kĺb pre ľavú stranu uchopovača	gripper_finger_joint_l
Kĺb pre pravú stranu uchopovača	gripper_finger_joint_r

4.1.2 Hokuyo model

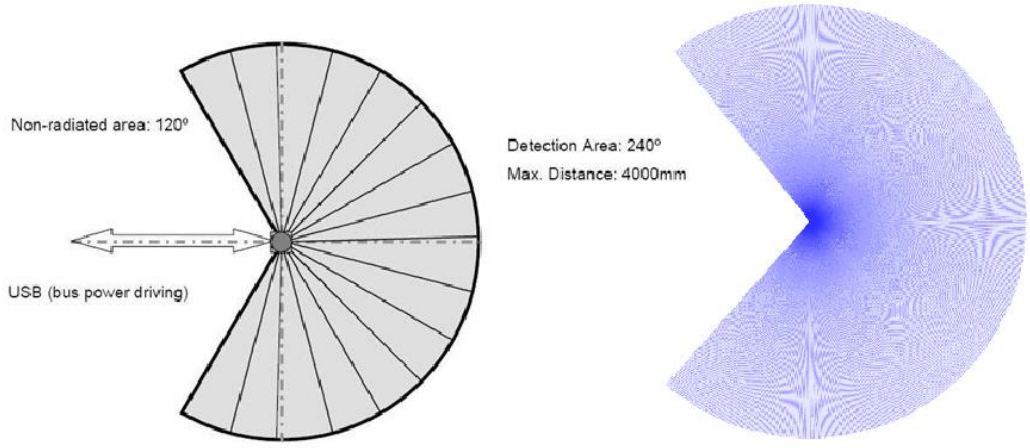
Hokuyo URG-04 je 2D laserový diaľkomer, pomocou ktorého vieme naskenovať plochu za pomoci laserového lúča.



Obr. 13 Hokuyo Lidar URG[16]

Infračervený laser, ktorý sa nachádza v Hokuyo URG-04 je pre človeka neškodný, pretože jeho vlnová dĺžka je 785nm, a spadá do bezpečnostnej triedy laserov 1. Do tejto triedy spadajú všetky lasery, ktoré majú vlnovú dĺžku menej ako 1400nm. Nemusíme sa obávať práce s takýmto laserom pretože je možné dlhodobé sledovania, priame sledovania a dokonca aj v prípade sledovania laseru pomocou optických pomôcok ako sú napríklad okuliare. Plocha, ktorú vie infračervený laser zachytiť je 240° (môžeme vidieť na Obr. 14), pričom minimálny polomer je 20mm a maximálny 5600mm. Uhlové rozlíšenie je 0,36° pretože na 240° vykoná 683 meraní. Hokuyo skenuje prestretie okolo seba vo frekvencii 10Hz čo je 100ms/meranie. Výrobca taktiež v dokumentácií udáva percentuálnu odchýlku lidaru , ktorá činí ± 3% z meraní vzdialeností medzi 1 až 4 metrami.

Model snímača je veľmi podobný reálnemu modelu. Uhol snímania je rovnaký ako pri reálnom modeli, čiže 240°, čo dokazuje aj obrázok z nástroja Gazebo, pričom modrá farba zobrazuje detekčnú zónu modelového snímača. Snímač je pripevnený na prednom ráme modelu robota, čo mu umožňuje dobre pozorovať prekážky pred sebou a z časti aj do bokov.[32]



Obr. 14 Detekčná zóna pre Hokuyo URG predpísaná a v nástroji Gazebo

Pre potreby použitia reálneho snímača Hokuyo je potrebné mať aj aktívny balíček s názvom Hokuyo_node, ktorý komunikuje so senzorom a dodáva nám potrebné dáta.

4.2 Priame ovládanie

Úplne základné a najjednoduchšie ovládanie robota je pomocou rôznych externých zariadení, napríklad na klávesnici alebo ovládači. Čo sa týka riadenia podvozku KUKA youBot tak ho vieme riadiť pomocou príkazov pre translačnú a rotačnú rýchlosť.

4.2.1 Priame ovládanie youBot ramena

Rameno vieme ovládať pomocou jemne nadstavby nástroja RVIZ-u s názvom MOVE IT. Po spustení nástroja, sa nám zapne prostredie podobné RVIZ-u, s tým rozdielom, že vidíme iba rameno a na jeho koncovom bode je umiestnená súradnicová sústava, pomocou ktorej si vieme nadstaviť kam sa má rameno natočiť. Po navolení cieľovej pozície a stlačení príkazu *Plan and Execute* sa trajektória pohybu ramena naplánuje a rameno sa začne presúvať do cieľovej pozície.

Avšak rameno youBota má 5 kĺbov t. j. 5 stupňov voľnosti (ich názvy môžeme vidieť v Tab. 2) a ak by sme potrebovali riadiť každý kĺb osobitne, tak musíme využiť názvy kĺbov a pomocou publikovania správ, každý kĺb samostatne natočiť na nami zvolenú pozíciu.

MOVE IT nám po spustené ponúka jeden uzol s názvom /move_group, ktorá publikuje témy v ktorých prebieha komunikácia s prostredím Gazebo kde sa vykonáva pohyb ramena.

4.2.2 Uzol teleop_twist_keyboard

Priame ovládanie robota pomocou vstupu na klávesnici je najjednoduchším ovládaním, ak potrebujeme robota niekam presunúť alebo otočiť. Pre ovládanie robota pomocou klávesnice je potrebné spustiť uzol s názvom /teleop_twist_keyboard, ktorý číta vstup z klávesnice.

Pre správne fungovanie je potrebné mať aktívny terminál, kde sme spustili uzol. Pri ovládaní vieme meniť maximálnu rýchlosť a lineárnu rýchlosť otáčania robota. Na samotné ovládanie robota je potrebných 9 kláves, ktoré sa využívajú na chod do všetkých strán, (dopredu, dozadu, doľava, doprava) ale aj na otočenie sa, zastavenie. Nachádzajú sa tu aj klávesy na mierne zabáčanie pri pohybe dopredu.

Po stlačení ľubovoľnej klávesy sa nám pošle správa na tému /cmd_vel ktorá putuje do nástroja Gazebo, kde sa robot pohybuje.[33]

4.3 Použité balíky

V nasledujúcej kapitole si rozoberieme potrebné balíky pre potrebu navigácie robota KUKA youBot, v prostredí ROS.

4.3.1 GMapping

Pomocou balíčku GMapping vieme vyskladať kompletnú mapu, ak náš robot obsahuje 2D laser pomocou ktorého snímame okolie. Taktiež je potrebné vedieť, kde sa robot nachádza čo nám zabezpečuje odometria. Výslednú mapu a jej postupnú tvorbu vieme sledovať v nástroji RVIZ.

Balíček obsahuje nasledujúce :

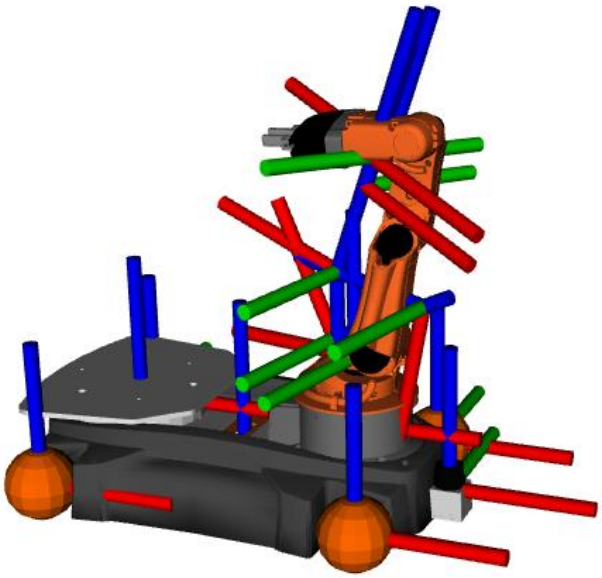
Uzly - gmapping využíva len jeden uzol s názvom slam_gmapping, ktorý číta a zároveň publikuje rôzne témy.

Témy rozdeľujeme na Odoberané témy a Publikované. GMapping odoberá témy s názvamy tf (tf/tfMessage), aby vedel z odometrie robota na akých súradniciach sa nachádza a z scan (sensor_msgs/LaserScan) dostáva dáta zo senzora z ktorých tvorí mapu. Čítať z balíku GMapping môžeme nasledujúce témy map_metadata (nav_msgs/MapMetaData) a map (nav_msgs/OccupancyGrid). Z témy map_metadata vieme vytiahnuť všetky dáta o mape, pričom samotnú mapu ponúka téma s názvom map.

Služby – ponúkanú máme iba jednu službu a ňou je dynamic_map s typom správy nav_msg/GetMap, pričom táto služba nám ponúka uloženie dát o mape.
Parametre – všetky parametre, ktoré nám ponúka GMapping nájdeme na ich oficiálnej stránke s dokumentáciou.[5]

4.3.2 Balík tf

Základným problémom pri mobilných robotoch je zistiť polohu robota. Poloha robota sa dá určovať rôznymi spôsobmi ako napríklad kamerou, ktorú má robot na vrchu tela, pomocou orientačných bodov na strope, ktoré využívajú na orientáciu napríklad niektoré vysávače od značky iRobot. Poloha sa dá určiť aj pomocou knižnice TF, ktoré prepočítava pohyb kĺbov, v našom prípade aj pohyb kolies. Pomocou balíku vieme určiť kde sa robot nachádza v porovnaní so svetovým rámom. Za pomoci tf vieme zistiť, kde bol robot pred piatimi sekundami, aká je aktuálna poloha robota voči predmetu a aká je aktuálna poloha základne robota voči mape. Tieto všetky informácie si balíček dopočítava v 3D súradnicovom systéme pomocou pohybu kĺbov robota. Dané informácie dostáva z témy robot_state_publisher, ktorá publikuje uhly natočenia všetkých pohyblivých častí, (kĺby a všesmerové kolesá) pričom dané informácie publikuje v 3D polohe.[29]



Obr. 15 YouBot ukážka súradnicových systémov vytvorených balíkom tf

4.3.3 Balík Map server

Po vytvorení kompletnej mapy pomocou balíčku GMapping je potrebné pre dokumentáciu mapu uložiť. Na túto akciu využijeme balíček s názvom map_server, ktorý nám túto funkciu ponúka.

Uložený obrázok pomocou map_saver-a popisuje stav obsadenosti každej bunky, (pixelu) ktorý sa na mape nachádza. V našej konfigurácii biele pixely znázorňujú voľný priestor a čierne pixely znázorňujú prekážku. Sivé pixely reprezentujú priestor, ktorý nepoznáme, teda ide o neznámy priestor. Vytvorenú mapu je možné uložiť príkazom, kde *mymap* stojí pre vlastný názov mapy. Príkaz:

```
rosrun map_server map_server mymap.yaml
```

Vytvorenú mapu si uložíme a použijeme ju ako statickú mapu pre globálnu navigáciu, pričom pri spustení ju budeme opätovne nahrávať do map_server-a.
Balíček obsahuje :

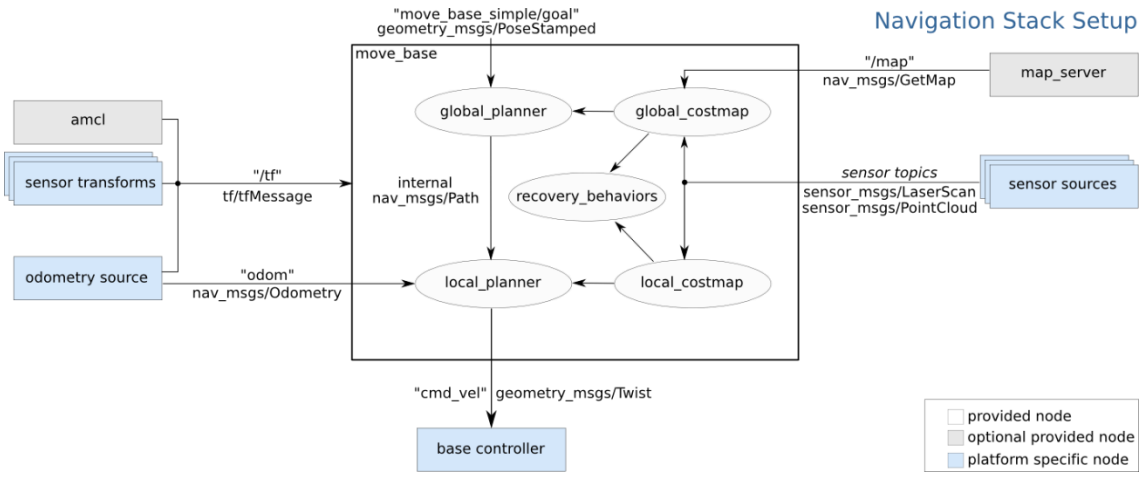
Uzly – map server využíva iba jeden uzol s názvom map_server, ktorý prevádza farebné hodnoty v obrazových údajoch mapy na hodnoty obsadenosti : voľné (0) obsadené (100) a neznáme (-1)

Témy ktoré číta uzol map_server sú map_metadata (správa nav_msgs/MapMetaData) a map (správa nav_msgs/OccupancyGrid), z týchto dvoch tém vie map_server pretvoriť mapu do inej podoby.

Služby – ponúkanú máme taktiež iba jednu službu ktorou je static_map (správa nav_msgs/GetMap), ktorá ponúka načítanie už vytvorenej mapy.[19]

4.3.4 Move base

Balík move_base nám implementuje akciu, pomocou ktorej vieme riadiť základňu robota vzhľadom na navigačný cieľ. Tento uzol nám spája globálnu a lokálnu navigáciu, aby sa úspešne robot presunul do cieľovej pozície a splnil tým úlohu globálnej navigácie. Na správny chod je ešte potrebné mať dve mapy ktorými sú globálna mapa obsadenosti a lokálna mapa obsadenosti, ktoré balík využíva na úspešný presun.[25]

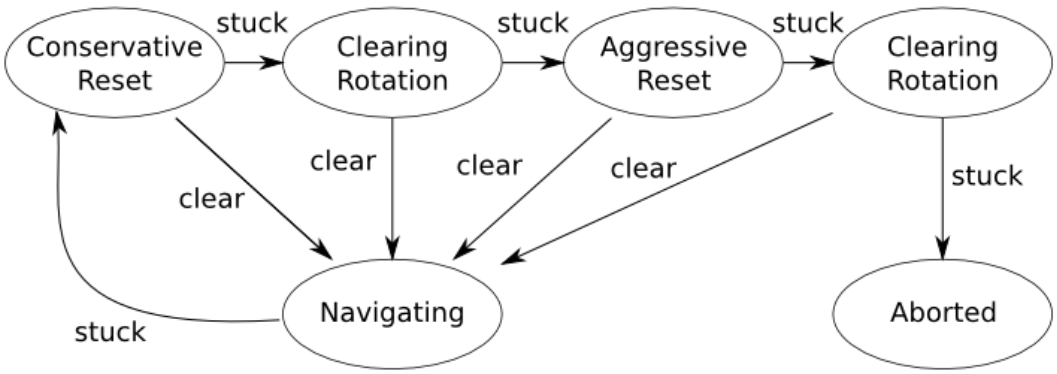


Obr. 16 Pôsobenie uzlov na uzol move_base [25]

Balíček obsahuje :

Uzly – Balíček move base poskytuje iba jeden uzol s názvom move_base, ktorý komunikuje s navigačnými balíkmi. Na Obr. 16 je zobrazený pohľad interakcie uzla move_base.

Pri zlyhaní navigácie uzol postupuje nasledovne. Robot sa pokúsi spraviť reset a začne sa otáčať, ak reštart pomôže, navigácia pokračuje ďalej. Ak nie, skúsi sa agresívnejší reset s následným otáčaním na mieste. Ak ani toto nepomôže, tak bude užívateľovi ohlásené, že akcia bola zrušená, v prípade ak bude reset vykonaný úspešne pokračuje sa v navigácii ďalej.



Obr. 17 Očakávané správanie robota pri zaseknutí[25]

Témy – Uzol move_base nám publikuje jednu tému s názvom /cmd_vel (geometry_msgs/Twist), pomocou ktorej prúdia príkazy ovládajúce rýchlosti pohybu všestranných kolies.

Služby – Uzol nám ponúka službu s názvom /make_plan (nav_msgs / GetPlan) pomocou ktorej vieme sledovať trajektóriu, ktorú sa robot bude snažiť napodobniť.[25]

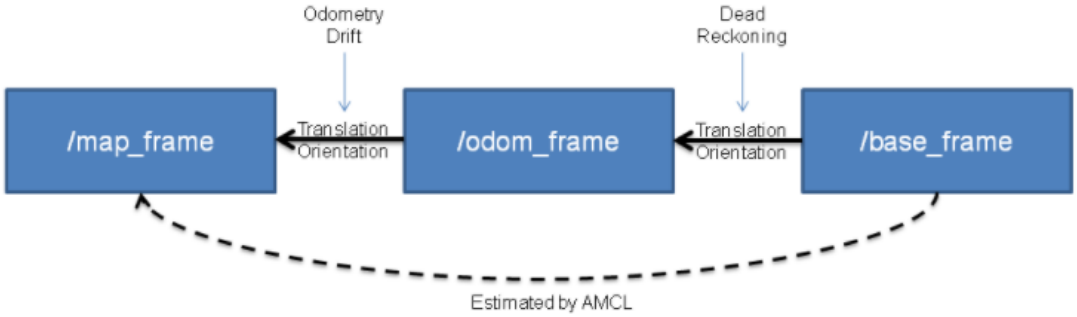
4.3.5 AMCL

AMCL je lokalizačný balík v 2D prostredí, využívajúci pravdepodobnostnú lokalizáciu.

Balíček obsahuje :

Uzly – Balíček AMCL poskytuje iba jeden uzol s názvom amcl, pričom tento uzol prijíma správy z laserového senzora, načítanú mapu z map servera, a transformácie z balíku tf.

Témy – Uzol amcl číta témy scan (sensor_msgs/LaserScan), aby získal dáta z lasera, tf (tf/tfMessage), kvôli dátam z odometrie modelu robota, Pričom naspäť publikuje na tému tf kde odosiela upravenú pozíciu robota.[27]



Obr. 18 Lokalizácia pomocou AMCL[27]

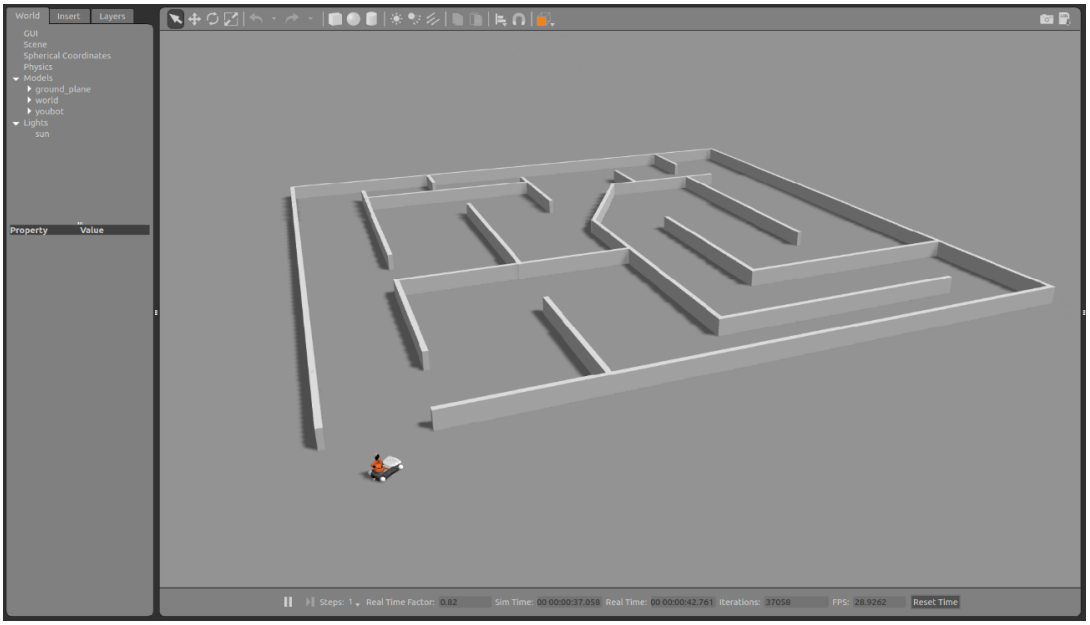
4.4 Nástroje ROS-u

Na vizualizáciu využíva systém ROS nástroje Gazebo a R-VIZ. Tieto nástroje si bližšie rozoberieme v nasledujúcich kapitolách.

4.4.1 Gazebo

Gazebo je 3D simulátor, ktorý s rozhraním ROS vedie k silnému robotickému simulátoru. Pomocou Gazeba si developer robotických systémov môže navrhnúť realistickú simuláciu, pretože simulátor ovláda fyziku, a gravitáciu vďaka ktorej je použitie robota v simulácií zhodné s jeho využitím v realite. Tento nástroj je často využívaný v simulácii situácie, kedy by mohlo prísť k poškodeniu robota. [34]

Za pomoci gazebo buildig editora si môžeme vytvoriť vlastné modely, ktoré potom vieme vložiť do rôznych svetov alebo si ich podľa potreby upraviť. My sme sa rozhodli pre model bludiska s jedným vstupom ktoré je zobrazené na Obr. 19.



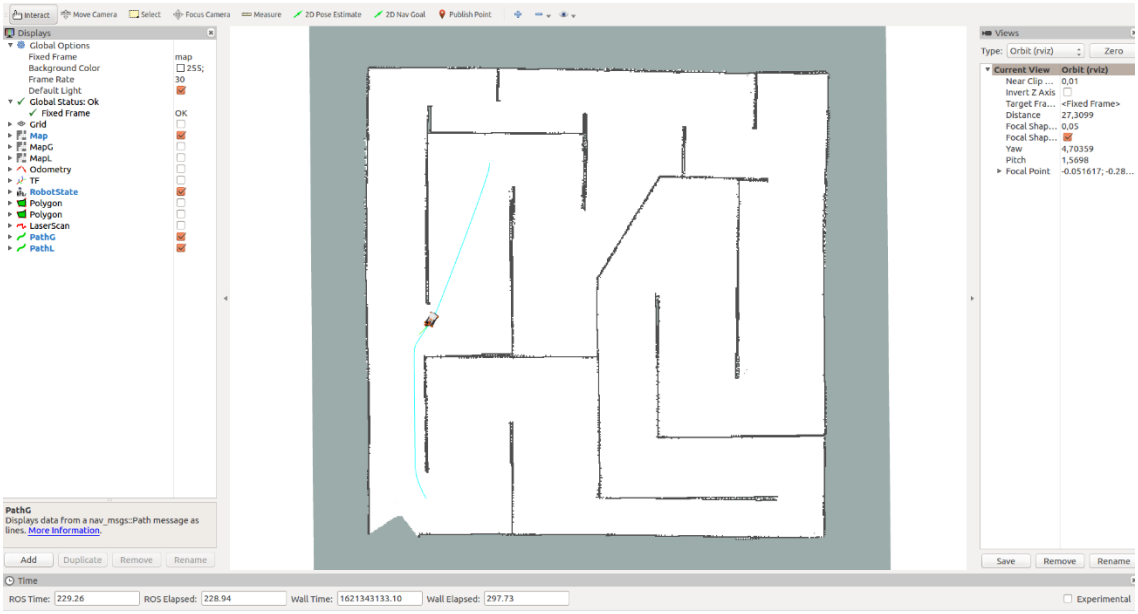
Obr. 19 Gazebo GUI

Simulačné prostredie je jednoduché, na ľavom panely sa nachádzajú základné nastavenia sveta ako napríklad *Physics* alebo *Models,* kde môžeme nájsť model youBota a taktiež model bludiska.

Po spustení Gazeba sa nám spustí iba jeden uzol s názvom /gazebo, ktorý odoberá témy ako napríklad set_model_state, pomocou ktorého môžeme dodatočne nastaviť polohu a natočenie modelu robota. Publikované témy sú ~/link_states (správa gazebo_msgs/LinkStates), ktorý nám publikuje dáta o všetkých polohách kĺbov a pohybov kolies. Pomocou ~/model_states (správa gazebo_msgs/ModelStates) vieme zistiť polohu robota v simulácii.

4.4.2 R-VIZ

Každý programátor potrebuje dostávať spätnú väzbu od robota alebo senzorov. Ako príklad si môžeme uviesť vykreslenie mapy a polohu robota, čo je iba jeden z viacerých využití 3D vizualizačného nástroja R-VIZ. Do R-VIZ-u si vieme navoliť, aké údaje chceme sledovať.



Obr. 20 R-VIZ zobrazenie vizualizačného prostredia

Taktiež pomocou RVIZ-u vieme robota ovládať, na čo nám slúži tlačidlo *2D Nav Goal*, ktoré sa nachádza na hornom panely. Tlačidlo vie robotovi zadať kam sa má na mape dostať, avšak musia byť spustené balíčky, ktoré túto navigáciu zabezpečia, napríklad balíček na globálnu navigáciu.

5. Použitie balíčkov a uzlov v simulácii

Pre správne fungovanie simulačného prostredia je potrebné aj jeho správne nastavenie, ktoré sme docielili pomocou využitia nadobudnutých poznatkov z teoretického využitia balíčkov potrebných na lokalizovanie a riadenie robota.

5.1 Youbot launch súbor

Pri hľadaní ovládača pre youBota sme sa rozhodli použiť kompletný balíček obsahujúci aj model youBota využiteľný v nástroji Gazebo, ktorý bol zároveň funkčný pre verziu ROSU Kinetic. Tento balíček obsahoval ovládač youbot_driver a youbot_oodl, ktoré by nám po implementácii do reálneho robota zabezpečovali komunikáciu s ROS-om. K oficiálnej dokumentácii k youbot_driveru už nie sú vyvíjané novšie verzie od daného autora a posledná aktuálna verzia je podporovaná iba na verziu ROS-u s názvom Hydro.

Prvé nastavenie, ktoré sme zmenili v launch súbore youbota je spúšťanie nášho sveta, ktoré docielime tým že pridáme cestu k spúšťaciemu súboru sveta :

```
<include file="$(find youbot_gazebo_worlds)/launch/myMaze.launch" />
```

Po spustení launch súboru sa nám automaticky rozbehne nástroj Gazebo s prostredím, v ktorom je implementované naše bludisko na súradnice 0 0 0 (x,y,z), ktoré môžeme vidieť na Obr. 21 a modelom youBota. Ďalšie nastavenie, ktoré sme upresnili bolo inicializačné umiestnenie youBota v simulovanom svete. Za cieľové miesto sme zvolili vstup do bludiska ktorého súradnice sú:

- x = -9
- y = -9.2
- z = 0

Taktiež tu nájdeme aj cesty určené na spustenie ovládača pre rameno youBota /arm_controller.launch a samostatný ovládač pre základňu /joint_state_controller. Na začiatku sme si museli preveriť, ktorú modifikáciu youBota spúšťame, aby boli zapnuté tie ovládače, ktoré potrebujeme, pretože pri modifikácii s dvoma ramenami budeme potrebovať samostatný ovládač aj pre druhé rameno.

Posledný uzol ktorý sa spúšťa je /robot_state_publisher, ktorý číta z URDF modelu robota parametre a tvorí z nich strom transformácií, ktorých výsledky zverejňuje pomocou balíka tf.

5.2 Vizualizácia prostredia

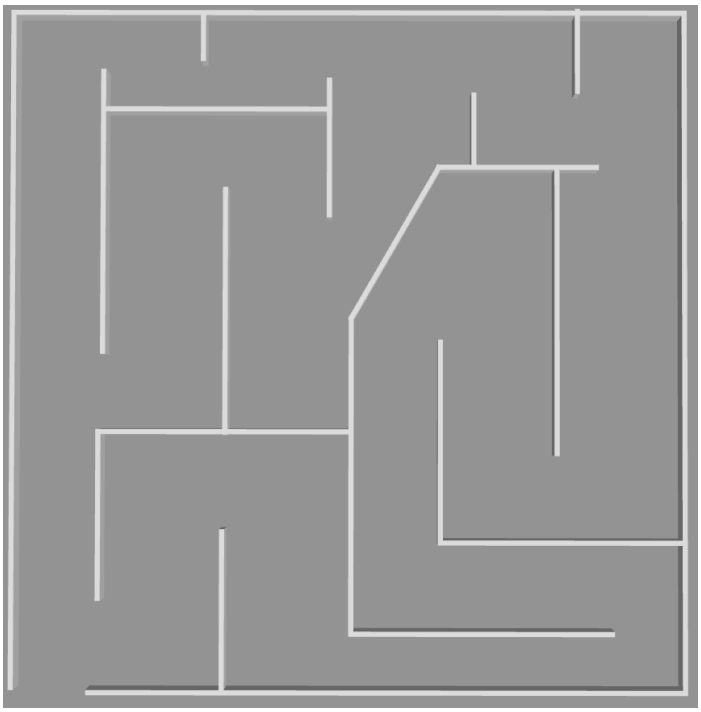
Pomocou využitia Gazebo simulátora a RVIZ konfigurácie dokážeme funkčnosť nami zvoleného robota v simulačnom prostredí a jeho schopnosti rozpoznávať prekážky a dostať sa z bodu A do bodu B.

5.2.1 Využitie Gazebo simulátora

Na preukázanie funkčnosti mapovacích balíčkov sme museli do prázdneho simulačného sveta pridať nami vytvorené modely prekážok.

Najskôr bol nami vytvorený model primitívneho bludiska, ktoré sme následne implementovali do prázdneho sveta, ktorý sme prepojili s funkčným prostredím, v ktorom sa bude youBot pohybovať.

Model sme si vytvorili na základe Gazebo building editora. Jeho simulačné rozmery sú 20 m^2 a v ľavom dolnom rohu sa nachádza vstup do bludiska, pričom sa v modeli nachádza veľa pravých uhlov, no taktiež aj slepé miesta a jedna šikmá stena, ktoré budú pre nášho robota prekážkou.



Obr. 21 Model bludiska

Po vytvorení modelu bola potrebná implementácia do prázdneho sveta, ktorú sme docielili nasledovne. Po otvorení .launch súboru sveta, ktorý obsahoval základné nastavenia sme vložili nasledovné príkazy na načítanie modelu:

```
<param name="world_description" command="$(find xacro)/xacro.py $(find
youbot_gazebo_worlds)/myMaze/model.sdf" />

<node pkg="gazebo_ros" type="spawn_model" name="gazebo_world_model"
args="-sdf -param world_description -model world -x 0.0 -y 0.0 -z 0.0"
respawn="false" output="screen" />
```

V prvom riadku je zadané, kde v súborovom systéme sa nachádza uložený model bludiska, a v druhom riadku je nadstavené, kde presne sa má daný model vo svete umiestniť, vzhľadom k tomu, že do sveta umiestniť viacero objektov.

5.2.2 RVIZ konfigurácia

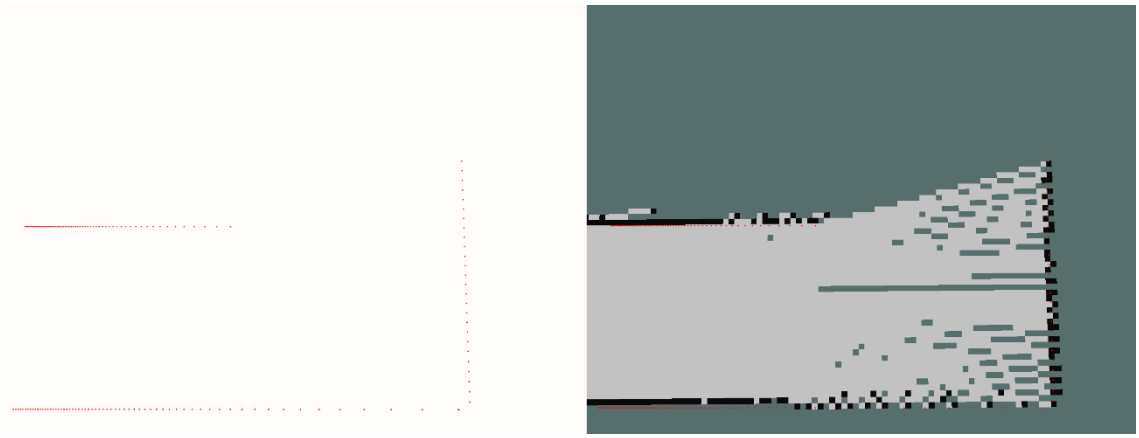
Pre efektívnejšiu prácu s nástrojom RVIZ bola nami vytvorená vlastná konfigurácia sledovaných hodnôt. Ako sme si mohli všimnúť na Obr. 20 v ľavom menu môžeme vidieť nami navolené sledovanie konkrétnych údajov, ktoré sa nám následne zobrazujú. Pre potreby vizualizácie mapy je potrebné pridať prvok s názvom /map vďaka ktorej vidíme zobrazenie na mapovanej oblasti pomocou 2D lasera, pričom máme na výber z lokálnej mapy, ktorá zobrazuje len okolie robota, globálnej mapy a statickej mapy načítanej z map servera. (porovnanie môžeme vidieť na Obr. 26)

Parametre s názvom *Odometry* sa nám zobrazujú pomocou červených šípok a hovoria nám kam robot smeruje, pričom sa uschováva niekoľko posledných krokov.

Pomocou údajov s názvom *RobotState* môžeme robota detegovať, kde na mape sa nachádza, pričom je možné vidieť aj model sledovaného robota a vďaka *tf* si vieme skontrolovať či ROS vidí všetky kĺby ako bolo zobrazené na Obr. 15.

Planner plan nám vie zobraziť trajektóriu robota z bodu A do bodu B, pričom si vieme navoliť či chceme vidieť celú trajektóriu alebo len lokálnu. Lepšie vizuálne porovnanie globálnej a lokálnej trajektórie sa nachádza na Obr. 25.

Dáta zo senzora sú uložené v *LaserScan* a vyobrazujú laser v prostredí R-VIZ a následne pomocou neho vytvárajú mapu, ktorá je zobrazená na Obr. 22.



Obr. 22 LaserScan v prostredí R-VIZ

Po úspešnom nastavení všetkých sledovaných parametrov boli nami vybrané nastavenia uložené do súboru s názvom *rvizCNFG.rviz*, pričom z prípony, nám je jasné že sa jedná o súbor patriaci k RVIZ-u. Následne sme si túto konfiguráciu nastavení zadali aj do Youbot_bc.launch súboru, z dôvodu, aby sa nám vždy pri spustení zobrazili nami navolené nastavenia.

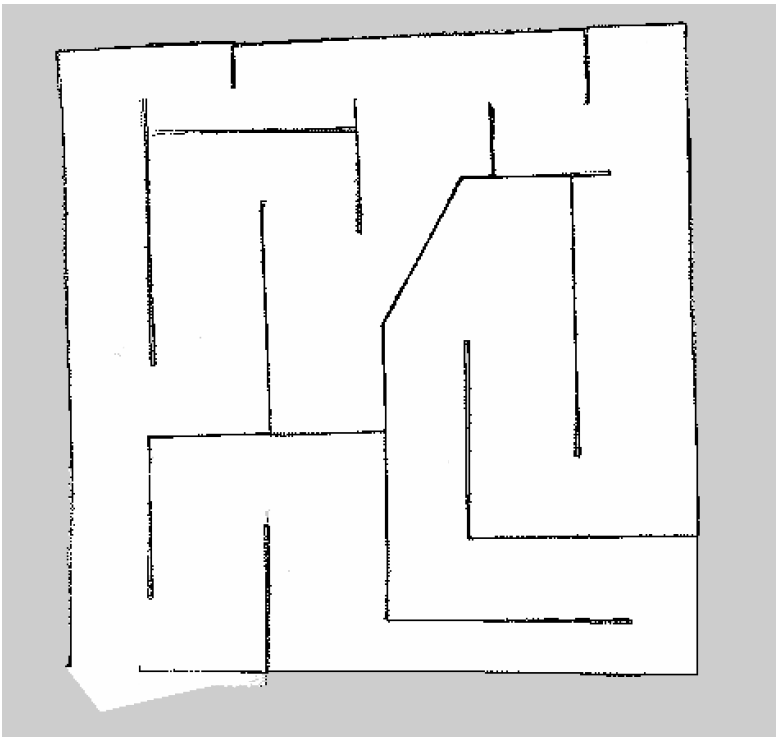
5.3 Mapovanie prostredia pomocou balíčku GMapping

Po umiestnení YouBota na vstupe do bludiska je potrebné pre začatie mapovania zapnúť uzol s názvom /slam_gmapping.

Pred samotným spustením je potrebné nastaviť priradenie do serveru s parametrami, ktoré boli nastavené nasledovne scna:=/base_scan. Následne sme ešte upravili parametre mapovania, ktoré sa týkali veľkosti mapy. Vieme že nami vytvorené bludisko má rozmery 20x20m, veľkosť výslednej mapy sme určili na 24x24m. Zmena sa týkala nasledujúcich parametrov :

- xmin=-12
- xmax=12
- ymin=-12
- ymax=12

Po spustení uzla GMapping potvrdí zmeny parametrov zopakovaním do terminálu. Následne sme robota pomaly previedli pomocou teleovládania celým bludiskom na najnižšej rýchlosti, aby nevznikali zbytočné chyby pri mapovaní. Po prejdení celého bludiska sme v RVIZ-e skontrolovali výsledok pomocou odoberania témy /map a pomocou map_servera sme uložili mapu prostredia. Výslednú mapu bludiska môžeme vidieť na obrázku nižšie.

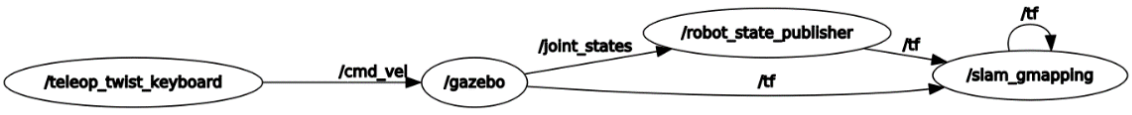


Obr. 23 Mapa bludiska vytvorená pomocou balíku GMapping

Ešte pred začatím mapovania sme skontrolovali správne prepojenie uzlov a ich vzájomnú komunikáciu pomocou príkazu:

```
$ rqt_graph
```

Vďaka vyššie spomenutému príkazu sa nám vyobrazí štruktúra všetkých aktívnych uzlov a ich prepojenia ako môžeme vidieť na Obr. 24.



Obr. 24 ROS rqt graf pri mapovaní pomocou GMapping

5.4 Navigačný balíček pre model youBota

Pre navigovanie youBota sme sa rozhodli zostrojiť navigačný balíček, ktorý bude obsahovať všetky potrebné uzly, nevyhnutné na zabezpečovanie lokalizácie, mapovania

a ovládania. Pri tvorbe balíku sme postupovali podľa návodu na oficiálnej stránke ROSu zameranej na tvorbu balíčkov.

Ako prvé bolo potrebné do konzoly zadať príkaz, v ktorom je definovaný názov balíčku a potrebné knižnice ktoré bude balíček podporovať. Tieto knižnice vieme aj dodatočne dopísať. Zadaný príkaz :

```
$ catkin_create_pkg nav_stack roscpp tf move_base_msg rospy
```

Catkin_create_pkg má na starosti vygenerovanie balíčku, *nav_stack* je názov balíčku a *roscpp* dáva podporu pre programovací jazyk C++, *rospy* zase pre Python a *tf* je knižnica potrebná pre balíček tf.

Po vytvorení balíčka bol rozdelený do troch pod priečinkov. Prvý sa nazýva configG a obsahuje konfiguračné súbory(parametre),ktoré sa načítavajú do potrebných uzlov. Sú tu konfigurácie pre globálny plánovač, lokálny plánovač a pre globálnu a lokálnu mapu obsadenosti.

Do druhého pod priečinku s názvom Map bola uložená vytvorená mapa prostredia za pomoci balíčku GMapping, ktorú budeme načítavať balíčkom map_server, aby bolo nami vytvorené prostredie pre youBota známe.

V treťom priečinku s názvom launch sa nachádzajú spúšťacie súbory pre jednotlivé uzly. Môžeme tu nájsť súbory s názvami amcl, slamGmapping, move_base. Taktiež sa tu nachádza spúšťací súbor, ktorý zabezpečuje teleovládanie youBota, ktoré je nutné spustiť samostatne pomocou príkazu:

```
$ rosrun nav_stack youbot_keyboard_teleop.py
```

Kde nav_stack je názov balíčku, v ktorom sa súbor nachádza a youbot_keyboard_teleop.py je samotný názov uzla.

5.4.1 Spúštací súbor Nav.launch

Spúšťací súbor s názvom nav.launch, ktorý sme vytvorili má za úlohu spúšťať uzly, ktoré zabezpečujú lokalizáciu, a riadenie robota.

Ako prvý uzol spúšťame /map_server, kvôli načítaniu už vytvorenej mapy, ktorú sme mohli vidieť na Obr. 23. Jej umiestnenie sme vložili do argumentu, a docielili sme

tým úspešné načítanie statickej mapy, pomocou ktorej sa bude vypočítavať výsledná trajektória robota.

Druhý uzol je uzol s názvom amcl, vďaka ktorému sa robot vie lokalizovať na 2D mape. Lokalizácia prebieha pomocou odosielania transformácii z mapy do odometrie, a tým docieľuje opravenie chýb ktoré môžu nastať v odometrii. Do uzla amcl sme museli pridať inicializačné pozície pre robota, aby sa zhodovali pri zapnutí simulácie. Ak by sme tieto hodnoty nemenili, uzol amcl by si myslel, že sa robot na začiatku simulácie nachádza v súradnicovom systéme v bodoch 0 0 0 (ak nie je určené inak je to východzí bod pre začiatok lokalizácie) a preto sme zmenili nasledovné :

```
<param name="initial_pose_x" value="-9.0"/>
<param name="initial_pose_y" value="-9.2"/>
```

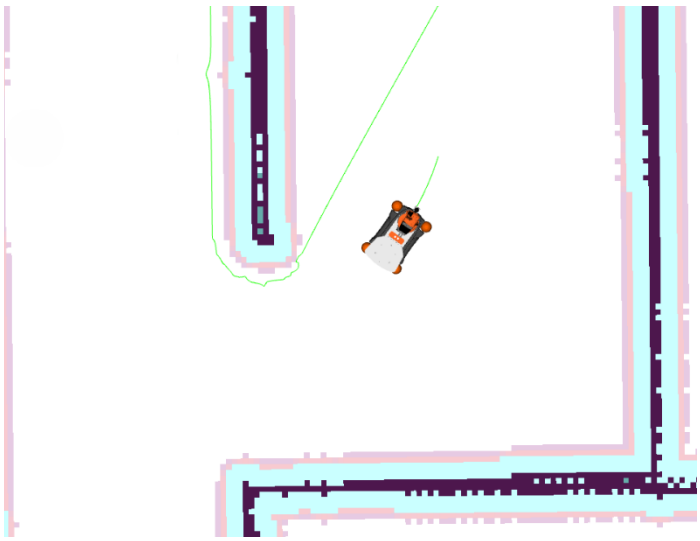
Tretí uzol je move_base, teda uzol, ktorý je základným uzlom balíka pre navigovanie robota. Ako sme spomínali už v kapitole 4.3.4 Move base, tak vieme, že daný uzol nám spája rôzne časti navigačného balíka.

Balíky ktoré nám tento uzol spája sú :

- Globálny plánovač (preklad /global_planner), zabezpečuje optimálny presun zo súradníc, na ktorých sa robot nachádza na cieľové súradnice za pomoci znalosti mapy, pričom my sme využili algoritmus Dijkstra, ak by sme chceli využiť A* algoritmus museli by sme upraviť parameter use_dijkstra na false.
- Lokálny plánovač (preklad /base_local_planner), rieši pohyb na lokálnej mape, čiže mape ktorá sa práve nachádza v okolí modelu robota, pričom robot je vždy v jej strede. Lokálnu mapu môžeme vidieť na Obr. 26, kde je vyobrazená aj lokálna a aj globálna mapa prostredia. Na obrázku Obr. 25 si môžeme všimnúť zobrazenie globálneho a lokálneho plánovača. Globálny udáva celú trajektóriu robota a lokálny iba aktuálnu, ktorá sa snaží kopírovať globálnu trajektóriu.
- Rotate_recovery, rieši udalosť zaseknutia robota, pričom jeho riešením je, že robot sa začne otáčať okolo svojej osy (yaw axis) o 360° pokiaľ sa neobnoví navigácia.
- Clear_costmap_recovery, ak nastane stav, že robot uviazne, napríklad narazí na prekážku, ktorá nie je na globálnej mape zaznačená, tak clear_costmap_recovery obnoví a nanovo začne vytvárať mapy obsadenosti, pričom využije už nové dáta

zo senzorov.

- Slam_gmapping, vie lokalizovať robota na tvorenej mape pomocou odometrie a dát zo senzorov (napríklad Lidaru), pričom vďaka týmto dátam dokáže zostrojiť zodpovedajúcu mapu prostredia. Taktiež vieme slam_GMapping využiť na lokalizáciu v už vytvorenej mape.
- Map_server
- Amcl



Obr. 25 Globálny a lokálny plánovač

Z uzlov AMCL a GMapping je potrebné si vybrať jeden, ktorý nám bude lokalizovať robota v priestore. Taktiež medzi voliteľné uzly patrí aj map_server, ktoré nie je nutné použiť.

5.5 Spúšťací súbor Youbot_bc

Youbot_bc.launch spája spustenie všetkých potrebných uzlov, ktoré sa využívajú pri navigovaní robota. Na kompletnú automatizáciu bolo nutné zahrnúť do prostredia nasledujúce spúšťače :
/youbot.launch, ktorého spustením (opísaný v kapitole Pre správne fungovanie simulačného prostredia je potrebné aj jeho správne nastavenie, ktoré sme docielili pomocou využitia nadobudnutých poznatkov z teoretického využitia balíčkov potrebných na lokalizovanie a riadenie robota.
5.1 Youbot launch súbor), sa zapne nástroj Gazebo, ktorý obsahuje svet s modelom nami vybudovaného bludiska a model youBota. Taktiež sa nám súbežne rozbehne uzol, vďaka ktorému vieme čítať parametre z modelu robota.

Na výber máme zo štyroch rôznych modelov youBota, ktoré chceme aby sa vo svete nachádzali. V prípade potreby je možné nahrať do sveta aj viac youBotov, no je potrebné, aby tomu boli prispôsobené parametre simulácie. V našej situácií však jeden model stačil.

Celý proces zabezpečuje nasledujúci riadok nachádzajúci sa v Youbot_bc.launch, ktorý udáva umiestnenie spúšťacieho súboru youbot.launch:

```
<include file="$(find youbot_gazebo_robot)/launch/youbot.launch" />
```

Gazebo s naším svetom a modelom youBota je implementované do globálneho spúšťacieho súboru, preto je potrebné sem zahrnúť aj spustenie riadiacich a lokalizačných balíčkov. Tieto balíčky sme si vopred pripravili v predchádzajúcej kapitole 5.4 , kde sme ich zahrnuli do jedného .launch súboru, ktorého cestu sem pridáme nasledujúcim riadkom:

```
<include file="$(find youbot_navigation_global)/launch/nav.launch" />
```

A posledným balíkom, je RVIZ nástroj s našimi už uloženými konfiguráciami, ktoré nájdeme v 5.2.2 RVIZ konfigurácia**.** RVIZ spúšťame pomocou nasledovného riadku :

```
<node type="rviz" name="rviz" pkg="rviz" args="-d $(find
youbot_gazebo_robot) /rvizCNFG/rvizGmappingCNFG.rviz" />
```

Pričom node nám hovorí o type a názve uzlu, pkg nám prezrádza balíček, ktorý voláme. V args sa nachádza cieľové uloženie našej prednastavenej konfigurácie nástroja RVIZ.

5.6 Publikovanie súradníc pre cieľ navigácie

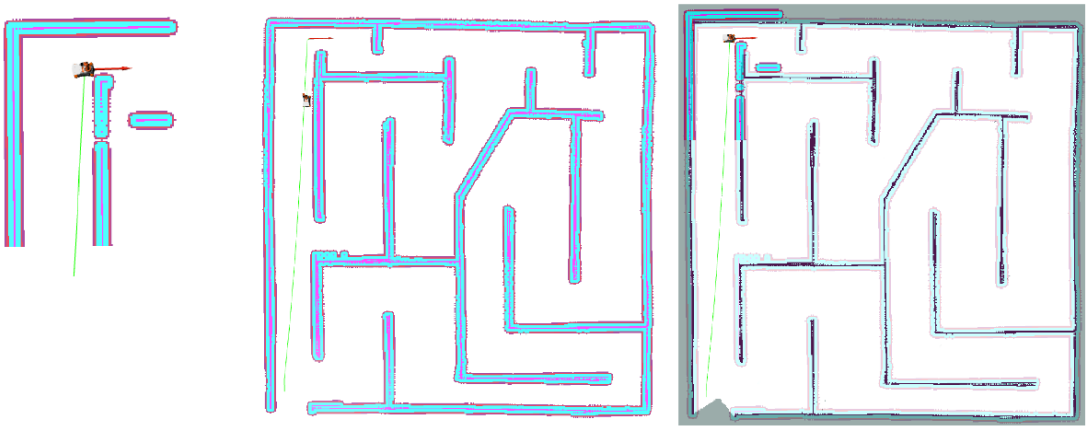
Aby robot vedel, na aké súradnice sa má dostať, musíme mu túto informáciu predať. Použijeme na to možnosť nástroja RVIZ, ktorá nám umožní voľbu cieľa navigácie pomocou tlačidla s názvom *2D Nav Goal*.

Pri zvolení tejto možnosti pomocou kurzora navolíme, kam sa má robot dostať, do ktorej strany má byť otočený. Ako môžeme vidieť na Obr. 26, na konci navigačnej

cesty je vyobrazená červená šípka, ktorá reprezentuje cieľ navigácie a výsledné natočenie robota.

Po zadaní navigačného cieľa sa vygeneruje trajektória, ktorú sa robot bude snažiť dodržať, avšak môžu nastať odchýlky v lokálnych prekážkach alebo odchýlkach medzi statickou a lokálnou mapou, ktoré sa na globálnej mape nemuseli zobraziť.

Na obrázku nižšie je zobrazenie zľava, lokálna mapa, globálna mapa a spojenie lokálnej globálnej a statickej mapy načítanej z map_servera v jednom obrázku.



Obr. 26 Zobrazenie lokálnej a globálnej mapy bludiska

Vo výslednom porovnaní si môžeme všimnúť drobné nepresnosti lokálnej mapy voči statickej mape z map_servera. Tieto nepresnosti mohli vzniknúť pri mapovaní prostredia, avšak v konečnom dôsledku sú zanedbateľné, pretože robot si pri každom pohybe kontroluje aj priestor okolo seba pomocou lokálnej mapy.

Ak by sme potrebovali zadať presné súradnice kam sa má robot dostať vieme napísať skript ktorý by publikoval na tému navigačného cieľa alebo pomocou nasledujúceho príkazu, ktorý zadáme do terminálu :

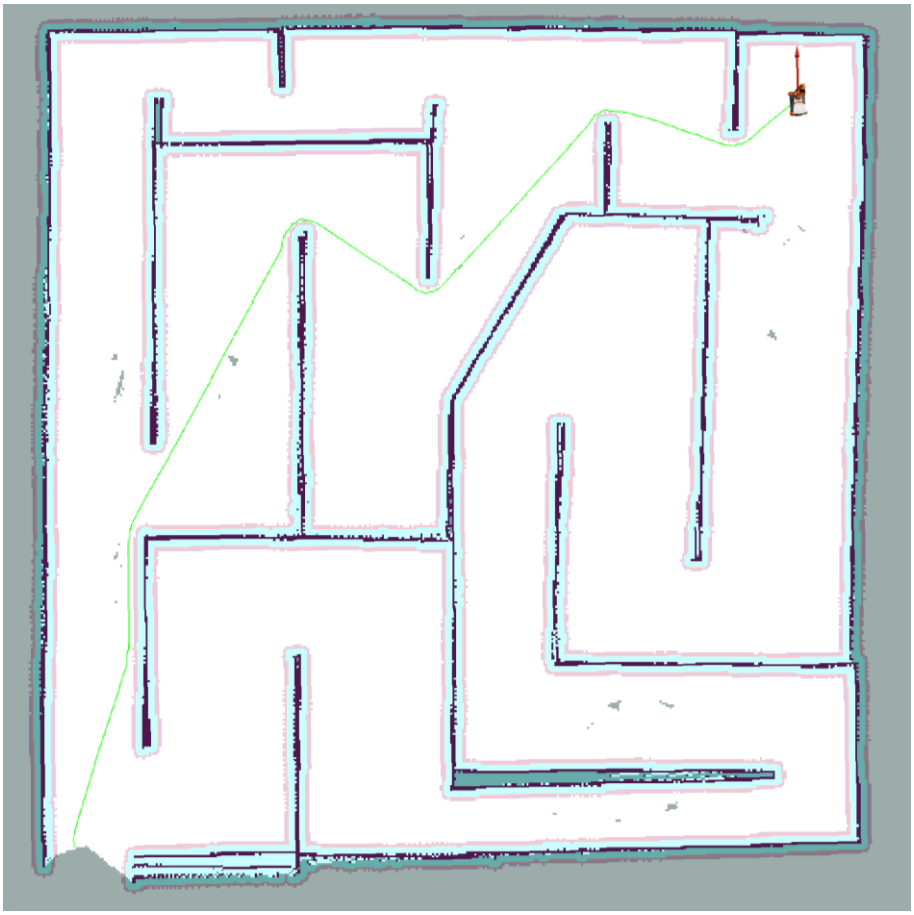
```
$ rostopic pub /move_base_simple/goal geometry_msgs/PoseStamped '{head
er: {stamp: now, frame_id: "map"}, pose: {position: {x: 1.0, y: 0.0, z
: 0.0}, orientation: {w: 1.0}}}'
```

Kde publikujeme na tému */move_base_simple/goal geometry_msgs/PoseStamped*, pričom zadávame súradnice na mape. Potom požadované súradnice a natočenie robota zadáme do zátvorky *position*.

6. Dosiahnuté výsledky práce

S nadobudnutými znalosťami z teoretickej práce, ktoré boli následne implementovali do praktickej časti práce sme schopný vykonať niekoľko testov na overenie správneho lokalizovania a riadenia robota v simulačnom prostredí.

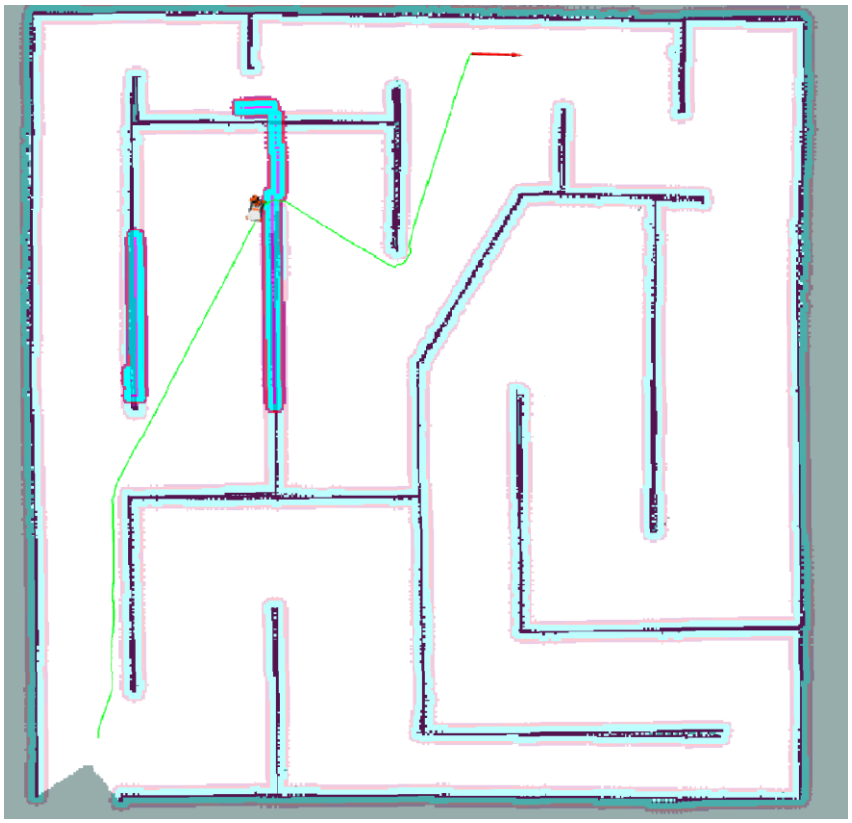
Vo vykonaných testoch lokalizáciu robota zabezpečuje Adaptívna Monte Carlo lokalizácia v skratke AMCL. Vytváranie globálnej cesty z bodu A do bodu B zabezpečuje global_planner s implementovaným algoritmom Djikstra a lokálne riadenie robota vykonáva základný vnorený uzol base_local_planner.



Obr. 27 Navigačný test č.1

V prvom teste sa snažíme dokázať správny výber optimálnej trasy, t. j. global_planner vypočíta najkratšiu trasu medzi začiatočným bodom a cieľom navigácie. Aby bol tento test ľahko rozpoznateľný aj voľným okom zvolili sme nasledovnú trasu a to aby sa robot dostal z ľavého spodného rohu do pravého vrchného bodu bludiska čo v súradnicovom systéme x y predstavuje presun zo súradníc -9 -9.2 na súradnice

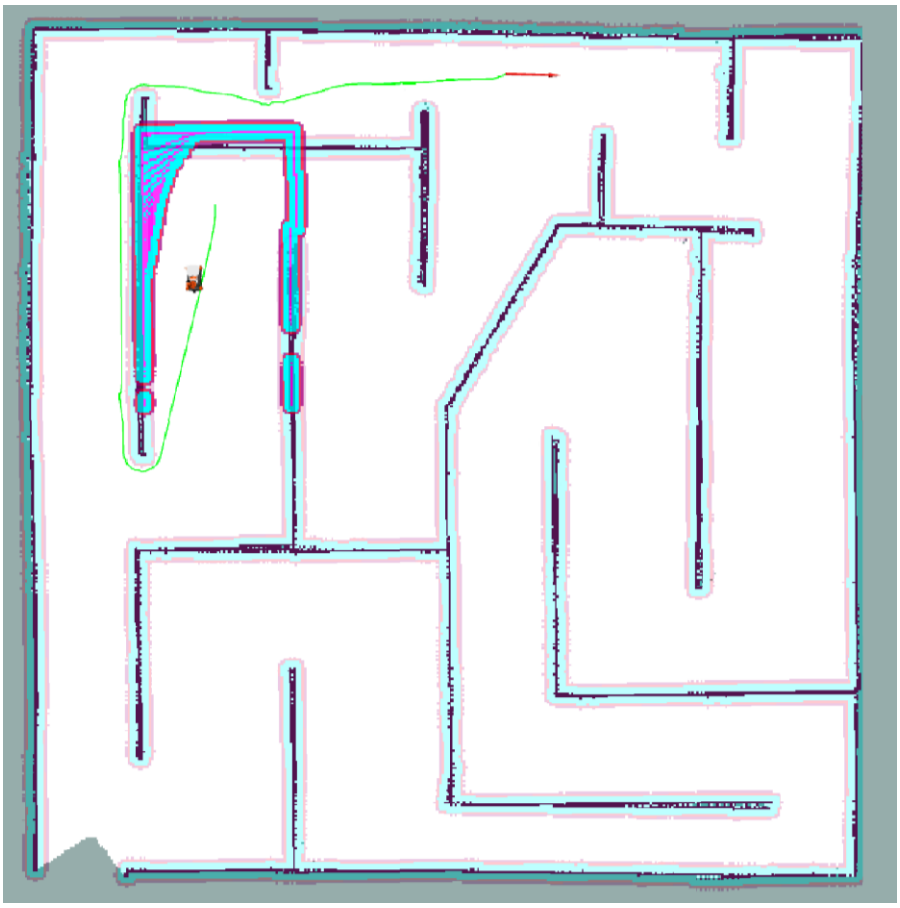
súradnice 8.8 8.2 (boli odčítané po dosiahnutí cieľa). Celá trajektória presunu robota je vyznačená zelenou čiarou, ktorá nám spája východzí a cieľový bod.



Obr. 28 Trajektória pred zresetovaním cesty do cieľu

V poradí druhý navigačný test bol zameraný na overenie správania navigácie ak pozmeníme prostredie ale statická globálna mapa nahratá do map servera zostane pôvodná. Prostredie sme pozmenili takým spôsobom že sme uzavreli strednú uličku, ktorá bola priechodná.

Následne z prvého testu sme nadobudli poznatku, že global_planner vypočíta optimálnu trasu cez stred bludiska, tak sme mu zadali cieľ navigácie podobný ako v prvom teste. Avšak ako reaktívna navigácia zaznamenala zmenu prostredia, zapísala túto informáciu a premapovala globálnu mapu z ktorej globálny plánovač nanovo vypočítal trasu do zadaného cieľa. Moment testu, kde lokálna navigácia zaznamenala prekážku môžeme vidieť na obrázku nižšie, kde je viditeľná zmena cesty tvorená globálnym plánovačom. Výslednú globálnu mapu obsadenosti z testu 2 vieme nájsť v prílohách. (Príloha B. 6 Globálna mapa obsadenosti z testu č.2 s neznámou prekážkou.)



Obr. 29 Trajektória po zachytení neuvedenej prekážky

Na lepšie preukázanie funkčnosti sme priložili aj dve videá. V prvom videu sa nachádza navigovania robota v prostredí RVIZ a Gazebo, pričom sme zadali dva rôzne navigačné ciele a sledovali pri tom autonómne riadenie robota. YouBot prešiel ku všetkým cieľom plynulo a bez kolízie pričom išiel najkratšou cestou.

V druhom videu s názvom videu sa nachádza zaznamenaný test č.2. Pričom je zobrazená uzavretá priechodná ulička, ktorá sa nenachádza na globálnej mape obsadenosti. Vo videu je preukázaná funkčnosť preplánovania trajektórie a taktiež dosiahnutie zadaného cieľa.

Ako si však môžeme všimnúť na videách je menší nesúlad medzi globálnou statickou mapou a lokálnou mapou. Tieto chyby merania nastali pri mapovaní prostredia za pomoci balíčku GMapping, buď kvôli lokalizačnej chybe alebo chybe pri pomalom obnovovaní mapovaného prostredia pretože GMapping ma obnovovaciu slučku nastavenú na 5s. Obe videá sú okrem príloh uložené aj na vebovej stránke youtube.com na nasledujúcich odkazoch Video č. 1 a video č.2 [35][36].

Záver

Cieľom našej práce bolo implementovať balíčky zabezpečujúce lokalizáciu a riadenie s využitím modelu KUKA robota a overiť ich funkčnosť.

V prvom bode sme sa podrobnejšie zoznámili so systémom ROS, kde sme nadobudli novým poznatkom o funkčnosti a práce s ním. Oboznámili sme sa s dostupnosťou balíčkov a pomocou ROS tutoriálu vyskúšali základné funkcie ROS-u.

Ďalej sme nadobudli znalosťou technických parametrov dostupných robotov od spoločnosti KUKA, ktorých modely by sme vedeli následne použiť v systéme ROS.

Od spoločnosti KUKA sme vybrali model youBota, ktorý sme implementovali do nástroja Gazebo, kde sme mu vytvorili prostredie v ktorom sa bude pohybovať. Následne bola preukázaná možnosť použitia SLAM konkrétne balíku GMapping pri mapovaní prostredia pričom samotný URDF model sme ovládali pomocou teleovládania. Ďalej sme preskúmali balíčky, ktoré by nám zabezpečovali lokalizáciu a riadenie mobilného robota pričom sme nadobudli znalostiam ako prebieha lokalizovanie a riadenie mobilných robotov.

V piatej kapitole sa nám podarilo naplniť náš hlavný cieľ práce a to úspešná autonómna navigácia mobilného robota v nami vytvorenom prostredí, čo dokazuje že sa nám podarilo vytvoriť funkčný navigačný balíček s názvom *nav_stack* v prostredí ROS. Pričom sme do tohto balíčku úspešne implementovali uzly zabezpečujúce lokalizáciu (AMCL) a uzly ktoré zabezpečujú riadenia (move_base) s využitím algoritmu Dijkstra. Výslednú funkčnosť preukazujú aj vytvorené videá opísané v poslednej kapitole. Taktiež sme sa venovali preukázaniu funkčnosti navigácie v poslednej kapitole, kde sme vykonali zopár testov a snažili sa preukázať ako navigácia mobilného robota prebieha s využitím nášho navigačného balíku.

Avšak v správnom navigovaní mobilného robota v známom prostredí je potrebná čo najlepšia a najpresnejšia znalosť prostredia čo sa v preukázaných videách javí ako najväčší nedostatok, ktorý by bolo potrebný vylepšiť do budúcich meraní, chyba pravdepodobne nastala pri mapovaní prostredia z dôvodu že aktualizácia mapy pri balíčku GMapping trvá dlhšie, preto by bolo potrebné spomaliť pohyby robota pri mapovaní prostredia.

Ďalším pokračovaním mojej práce by bolo adekvátne v budúcnosti otestovať navigačný balíček aj na hardvérovom modeli youBota a preukázať jeho funkčnosť aj

v reálnom prostredí, a vykonať testy rovnakého charakteru ako sme vykonali v simuláciách.

Úspešne sme dokončili cieľ našej práce, podarilo sa nám model youBota autonómne navigovať naprieč celým bludiskom, preto Robot Operating System môžem odporučiť v rámci simulácie na riešenie akéhokoľvek typu roboticky zameraných úloh. Okrem autonómneho navigovania modelu youBota sme dokázali, že ROS vie byť aj plnohodnotný robotický simulátor.

Zdroje

[1]Duchoň, František. Lokalizácia a navigácia mobilných robotov do vnútorného prostredia. 1. vyd. Bratislava: Nakladateľstvo STU v Bratislave, 2012. ISBN 978-80-227-3646-6. [Citované 30.4.2021]

[2]Autor neuvedený. Bezpečnosť laseru Triedy 1 až 4 [online]. Dostupné na internete: http://www.mylt.sk/e-learning/laser/bezpecnost-laseru-tridy-1-az-4.

[3]Autor neuvedený. Dokumentácia Hokuyo URG-04LX-UG01.Dostupné na internete: https://www.hokuyo-aut.jp/search/single.php?serial=166 [Citované 20.8.2020]

[4]Autor neuvedený. História ROS-u. Dostupné na internete: https://www.ros.org/history/ [Citované 5.10.2020]

[5]Gerkey B.. Gmapping . [Citované 5.4.2021] Dostupné na internete: http://wiki.ros.org/gmapping

[6]KUKA. Špecifikácie robotického manipulátora KUKA KR6. Dostupné na internete: https://www.kuka.com/-/media/kuka-downloads/imported/6b77eecacfe542d3b736af377562ecaa/0000205456_en.pdf

[7]KUKA. Špecifikácie robotického manipulátora KUKA IIWA. Dostupné na internete: https://www.kuka.com/-/media/kuka-downloads/imported/9cb8e311 bfd744b4b0eab25ca883f6d3/kuka_lbr_iiwa_broschuere_de.pdf

[8]Autor neuvedený. Obrázky robotického manipulátora KUKA IIWA. Dostupné na internete: https://www.making.unsw.edu.au/learn/kuka-lbr-iiwa-learn-module/#Safety_in_the_Collab_Robotics_Lab

[9]Shaun Edwards. Podrobnosti o KUKA Experimental. [Citované 5.1.2021]Dostupné na internete: http://wiki.ros.org/kuka_experimental

[10] KUKA. Špecifikácie v podobe obrázkov mobilného robota KUKA youBot. Dostupné na internete: https://www.generationrobots.com/img/Kuka-YouBot-Technical-Specs.pdf

[11] Autor neuvedený. ROS Master a Uzly komunikácia, obrázok. Dostupné na internete : https://www.dailyautomation.sk/04-ros-robot-operating-system-zakladne-principy/

[12] Jan Paulus. Dokumentácia k youbot_driver [Citované 9.2.2021] Dostupné na internete: http://wiki.ros.org/youbot_driver

[13] Sebastian Blumentha. Dokumentácia k youbot_oodl. [Citované 9.2.2021]

Dostupné na internete: http://wiki.ros.org/youbot_oodl

[14] Autor neuvedený. Youbot API . Dostupné na internete: http://janpaulus.github.io

[15] Margaret E. Jefferies, Wai-Kiang Yeap. Robotics and Cognitive Approaches to Spatial Mapping

[16] Autor neuvedený. Model Hokuyo Lidar. Dostupné na internete : https://cyberbotics.com/doc/guide/lidar-sensors

[17] GRISETTI, G. – STACHNISS, C. – BURGARD, Gmapping [Citované 27.4.2021] Dostupné na internete : https://openslam-org.github.io/gmapping.html

[18] Autor neuvedený. Distribúcie ROS-u a základné informácie. [Citované 9.11.2020] Dostupné na internete : http://wiki.ros.org/Distributions

[19] Brian Gerkey, Tony Pratkanis, Špecifikácie k uzlu map_server. [Citované 27.3.2021] Dostupné na internete : http://wiki.ros.org/map_server

[20] Autor neuvedený. ROS Industrial.[Citované 18.2.2021] Dostupné na internete : https://rosindustrial.org

[21] WilliwGaragevideo. Dosiahnutý a vysvetlený milník 2. Dostunpé na internete : https://www.youtube.com/watch?v=I1emTXIzhZw

[22] Brian Gerkey. Robonaut 2. [Citované 9.2.2021] Dostupné na internete : https://www.ros.org/news/2014/09/ros-running-on-iss.html

[23] Stefan Kohlbrecher, Johannes Meyer. Hector Slam. Dostupné na internete : http://wiki.ros.org/hector_slam

[24] Dave Hershberger, David Gossow, Josh Faust Rviz dokumentácia. Dostupné na internete : http://wiki.ros.org/rviz

[25] Eitan Marder-Eppstein. Balíček move_base .[Citované 19.3.2021] Dostupné na internete : http://wiki.ros.org/move_base

[26] Autor neuvedený. Robonaut 2. [Citované 19.11.2020] Dostupné na internete : https://robonaut.jsc.nasa.gov/R2/

[27] Brian P. Gerkey, Špecifikácia balíku AMCL. [Citované 19.11.2020] Dostupné na internete : http://wiki.ros.org/amcl

[28] Autor neuvedený, Rozdelenie systému ROS. [Citované 19.10.2020] Dostupné na internete:
http://matlab.fei.tuke.sk/wiki/index.php?title=ROS(robotic_operating_system)

[29] Tully Foote, Eitan Marder-Eppstein, Wim Meeussen. Špecifikácia balíčku tf.[Citované 15.5.2021] Dostupné na internete : http://wiki.ros.org/tf

[30] Locomotec. Kuka youBot User Manual.[Citované 18.5.2021] Dostupné na internete : http://ftp.youbot-store.com/manuals/KUKA-youBot_UserManual.pdf

[31] Autor neuvedený. Informácie o systéme ROS [Citované 5.1.2021] Dostupné na internete : https://www.ros.org/core-components/

[32] Autor neuvedený. Špecifikácia k lidaru HOKUYO. Dostupné na internete : https://www.hokuyo-aut.jp/search/single.php?serial=165

[33] Graylin Trevor Jay, Popis uzlu teleop_twist_keyboard [Citované 16.1.2021] Dostupné na internete : http://wiki.ros.org/teleop_twist_keyboard

[34] Nate Koenig, Andrew Howard, Opis nástroja Gazebo [Citované 5.3.2021] Dostupné na internete : http://wiki.ros.org/gazebo

[35] Tomáš Nyiri. Video test zadávanie navigačných cieľov. Dostupné na internete : https://www.youtube.com/watch?v=zF5CSrNXiuI

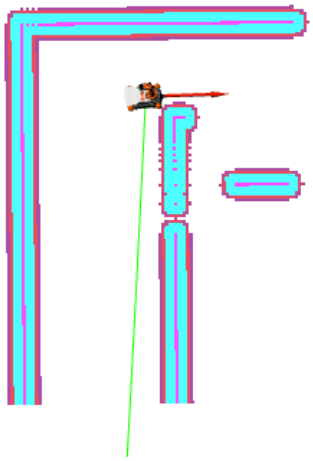
[36] Tomáš Nyiri. Video test s reakcia navigácie na neuvedenú prekážku. Dostupné na internete : https://www.youtube.com/watch?v=tSKZUSGOZWs

Prílohy

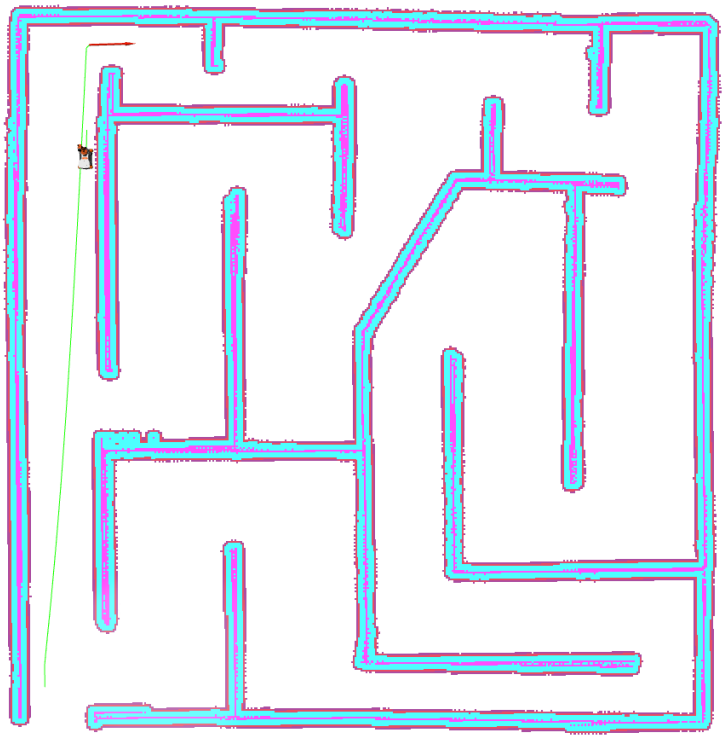
A. Elektronická príloha :

1. Video dokumentácia
2. ROS balíčky a súčasti

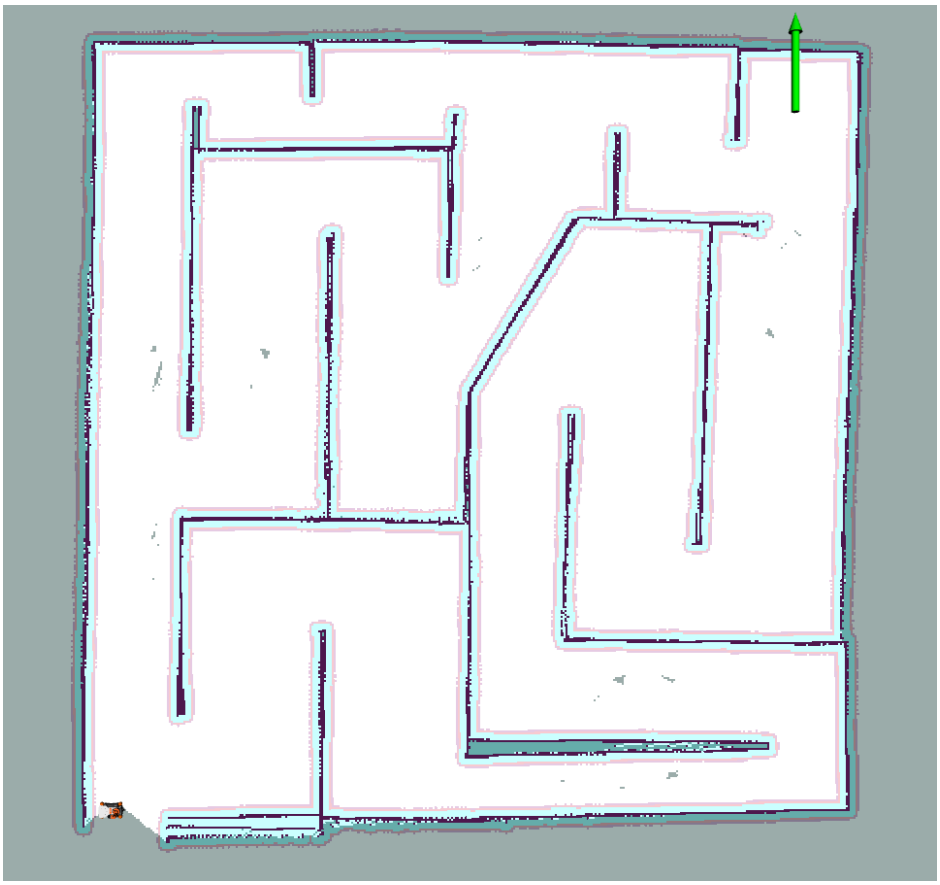
B. Fotografická dokumentácia



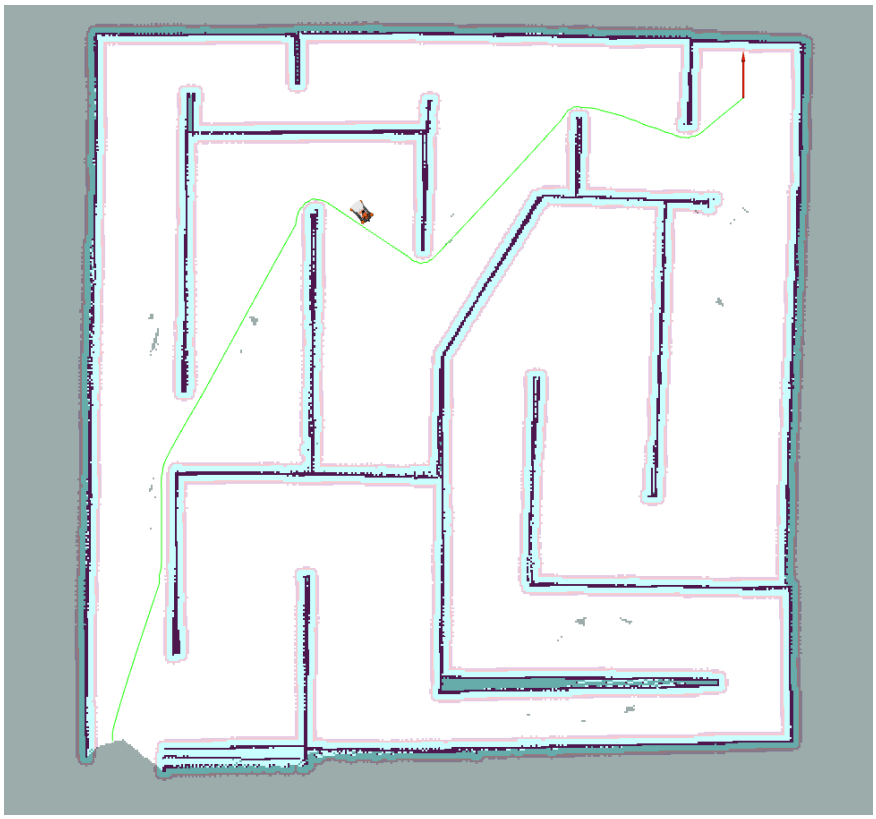
Príloha B. 1 Lokálna mapa obsadenosti



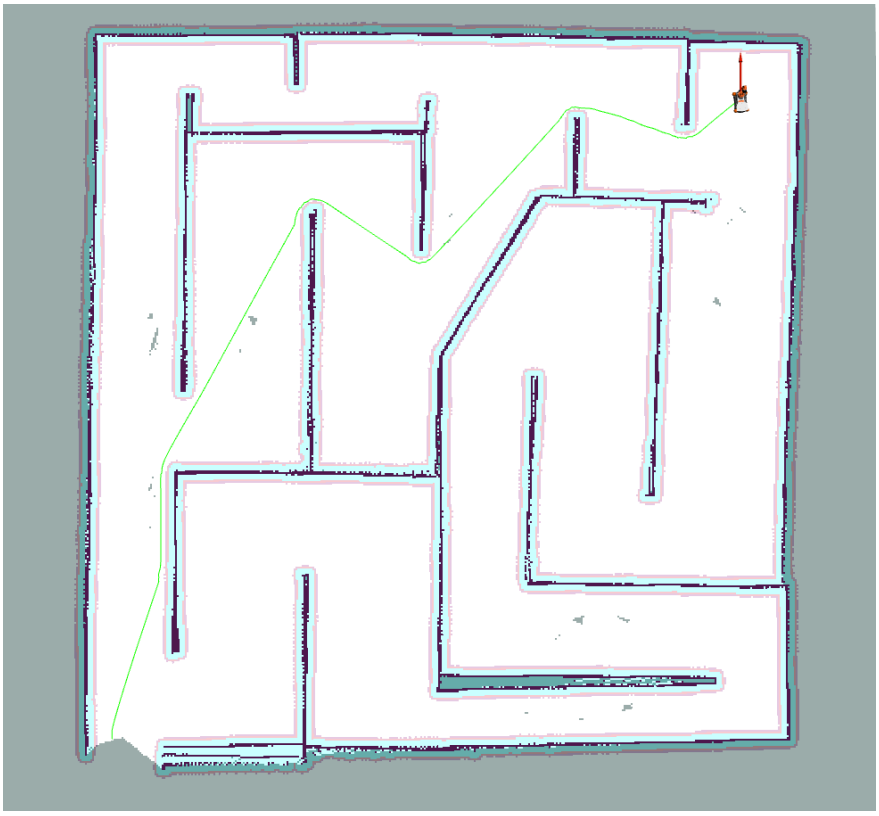
Príloha B. 2 Globálna mapa obsadenosti



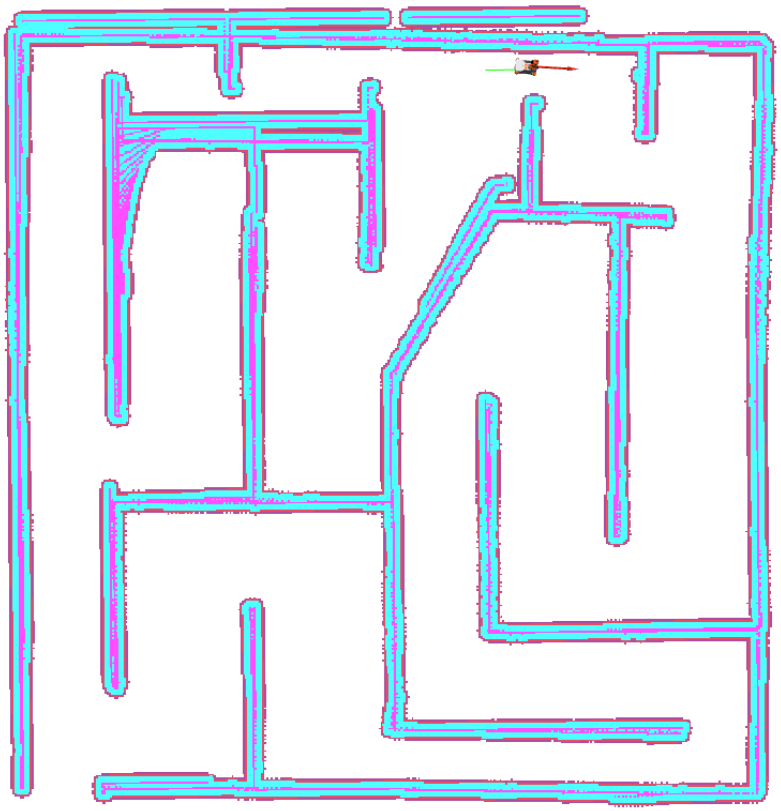
Príloha B. 3 Zadanie cieľu navigácie



Príloha B. 4 Trajektória navigácie a presun k cieľu



Príloha B. 5 Dosiahnutie Cieľu navigácie



Príloha B. 6 Globálna mapa obsadenosti z testu č.2 s neznámou prekážkou.