

**SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE FAKULTA ELEKTROTECHNIKY
A INFORMATIKY**

**Príprava cvičení a simulácie na predmet
Riadenia mobilných robotov**

TÍMOVÝ PROJEKT

Študijný program:	Robotika a kybernetika
Študijný odbor:	9.2.7 Kybernetika
Školiace pracovisko:	Ústav robotiky a kybernetiky
Vedúci záverečnej práce/školiťel:	Ing. Miroslav Kohút

Bratislava 2021

**Andrej Šutý
Samuel Kostúr
Róbert Málík
Samuel Kacej**

TÍMOVÝ PROJEKT ZADANIE

Študijný program: Robotika a kybernetika
Študijný odbor: kybernetika
Vedúci projektu: Ing. Miroslav Kohút
Miesto vypracovania projektu: Ústav robotiky a kybernetiky
Riešitelia: Bc. Andrej Šutý, Bc. Samuel Kostúr,
 Bc. Róbert Málík, Bc. Samuel Kacej

Názov projektu:
 Príprava cvičení a simulácie na predmet Riadenia mobilných robotov

Špecifikácia zadania:

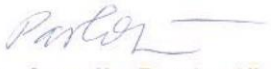
Úlohou riešiteľov tímového projektu bude príprava simulovaného zadania pre možnosť rozšírenia predmetu Riadenie mobilných robotov (pre prípady online výučby). Zadanie bude rozširovať predmet o integráciu mobilného robota do vonkajšieho prostredia a simulovať správanie sa globálneho a lokálneho plánovania pre diferenciálny podvozok.

Úlohy:

1. Naštudovanie si existujúcich riadiacich programov robota MRVK.
2. Naštudovanie si Robotického operačného systému a možnosti simulácie v prostredí Gazebo.
3. Integrácia riadenia robota do prostredia Gazebo.
4. Integrácia senzorov robota do prostredia Gazebo.
5. Analýza hardvéru robota pre prípadné aplikovanie simulácie na reálne zariadenie.
6. Vyhodenie dokumentácie k tímovému projektu a riadeniu robota.

Termín odovzdania projektu: 14.5.2021

V Bratislave dňa 15.2.2021


 prof. Ing. Jarmila Pavlovičová PhD.
 garantka študijného programu

Obsah

Úvod	5
1 Robotický operačný systém (ROS)	6
1.1 Medziprocesová komunikácia	6
1.1.1 Hlavný uzol (Master)	6
1.1.2 Uzol (Node)	6
1.1.3 Téma (Topic)	7
1.1.4 Správa (Message)	7
1.1.5 Publikovanie (Publishing)	7
1.1.6 Odoberanie (Subscribing)	7
1.1.7 Ukážka medziprocesovej komunikácie cez témy	7
2 Integrácia simulačného modelu robota do prostredia Gazebo	9
2.1 Simulačné prostredie Gazebo	9
2.2 Model robota	10
2.3 Laserový diaľkomer	11
2.4 Inerciálna meracia jednotka (IMU)	13
2.5 Navigácia	15
2.5.1 Globálna navigácia	15
2.5.2 Reaktívna navigácia	15
2.5.3 Balík move_base	15
2.5.4 Metóda dynamického okna	17
Množiny obmedzení rýchlostí	17
Účelová funkcia	19
2.6 Lokalizácia a mapovanie	19
2.6.1 Balík gmapping	20
2.6.2 Príklad algoritmu SLAM	21
2.7 Vizuálny systém na sledovanie cesty	21
2.7.1 Kamera	21
2.7.2 Detekcia chodníka	21

2.7.3	Pridanie novej vrstvy do účelových máp	25
	Účelová mapa (Costmap).....	25
	Perspektívna transformácia	26
	Kalibrácia kamery	28
	Implementácia novej vrstvy	29
3	Analýza robota MRVK	31
3.1	Hardvér	31
3.2	Softvér	32
4	Balík tímového projektu	34
4.1	Architektúra projektu.....	34
4.2	Inštalácia	35
4.2.1	ROS Melodic	35
4.2.2	Balík tímového projektu	36
	Záver	37
	Literatúra	39

Úvod

Niet pochýb o tom, že v súčasnosti pozorujeme obrovský rozmach v oblasti robotiky. Tento trend sa však netýka len implementácie robotiky v priemysle za účelom výrobných operácií. Práve naopak, aktuálna situácia koronakrízy ukázala, že kľúčovú úlohu zohrávala práve servisná robotika. Servisné roboty boli aplikované najmä vo vysoko infekčných prostrediach ako napríklad hromadná doprava a nemocnice za účelom ich dezinfekcie.

Ďalším aktuálnym príkladom je aj donáška jedla prostredníctvom mobilných robotov, ktorá zabezpečila nižšiu mieru medziľudskej interakcie. V oblasti mobilnej robotiky sa nepretržite otvára mnoho nových príležitostí čo len potvrdzuje, že má veľký význam sa touto problematikou zaoberať.

Dvere do sveta mobilnej robotiky otvára mnoho súťaží, pričom jednou z nich je aj Robotour, ktorej sa zúčastňuje aj tím MRVK sídliači na FEI. Robotour je pretekom mobilných autonómnych robotov s cieľom preniesť užitočný náklad.

Cieľom tohto zadania je príprava simulovaného zadania pre možnosť rozšírenia predmetu Riadenie mobilných robotov vychádzajúc práve z robota MRVK. Problematika tímového projektu je spracovaná v štyroch sekciách. Prvá sekcia je venovaná základným informáciám týkajúcich sa Robotického operačného systému. Druhá sekcia je zameraná na integráciu simulačného modelu robota do prostredia Gazebo vrátane popisu jednotlivých modulov použitých pri riadení robota. Tretia sekcia opisuje Hardvérové a Softvérové vybavenie samotného robota MRVK. V štvrtej sekcií je popísaný postup potrebný pre inštaláciu Robotického operačného systému a samotného balíku tímového projektu.

1 Robotický operačný systém (ROS)

ROS je voľne dostupný softvér, obsahujúci knižnice a nástroje pre vývoj robotických aplikácií s možnosťou samotného rozšírenia softvérového rámca. Idea vzniku ROS-u spočívala vo vytvorení nástroja na kolaboratívny vývoj softvéru v oblasti robotiky.

1.1 Medziprocesová komunikácia

V tejto práci boli na medziprocesovú komunikáciu preferované témy (topics). Ďalej je preto práca rozpracovaná hlavne na danú medziprocesovú komunikáciu. Celkovo ROS poskytuje 3 základné spôsoby medziprocesovej komunikácie:

- **Témy (Topics)** – určené na spojitý príjem a vysielanie dát.
- **Služby (Services)** – určené pre vzdialené volania procedúr, procedúry musia byť rýchle, nie sú vhodné pre dlho vykonávajúce sa procesy.
- **Akcie (Actions)** – určené pre akékoľvek diskrétné udalosti, alebo aj udalosti, ktoré trvajú po dlhšiu dobu ale poskytujú spätnú väzbu počas vykonávania.[5]

1.1.1 Hlavný uzol (Master)

Master je hlavný proces v ROS, synchronizuje komunikáciu medzi ostatnými uzlami. Spravuje údaje o jednotlivých uzloch a ich publikovaniach a odoberaniach z danej témy.[5]

1.1.2 Uzol (Node)

Uzol predstavuje samostatný proces, na ktorom sa vykonáva nejaký program. ROS je zameraný práve na prácu s paralelne existujúcimi uzlami, takáto konfigurácia prináša mnoho výhod. Medzi hlavné výhody patrí zvýšenie odolnosti voči chybám, pričom chyby sú izolované na jednotlivých uzloch, celkovo dochádza k zjednodušeniu štruktúry kódu. Ďalšou významnou vlastnosťou je kolaboratívny prístup, ROS umožňuje jednoduché spájanie od viacerých kolaborantov (aj z viacerých programovacích jazykov) tým, že jednotlivé balíčky sa vykonávajú na nezávislých uzloch.[5]

1.1.3 Téma (Topic)

Téma predstavuje akúsi virtuálnu zbernicu, na ktorú sú publikované správy a zároveň môžu byť z nej tieto správy aj odoberané. Publikovanie aj odoberanie z témy je anonymné, jednotlivé uzly nevedia, ktorý publikuje ani, ktorý odoberá, podstatné je len vedieť tému a dátový typ, na ktorú chce daný uzol publikovať popřípade z nej odoberať. Na jednu tému môže byť súčasne neobmedzený počet publikovaní a odoberaní. [5]

1.1.4 Správa (Message)

Správa predstavuje dátovú štruktúru, ktorá môže byť publikovaná alebo odoberaná z danej témy. [5]

1.1.5 Publikovanie (Publishing)

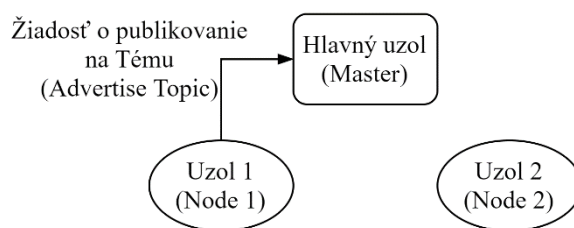
Publikovanie znamená generovanie správ na príslušnú tému. Uzol, ktorý publikuje, sa nazýva vydavateľ (publisher).[5]

1.1.6 Odoberanie (Subscribing)

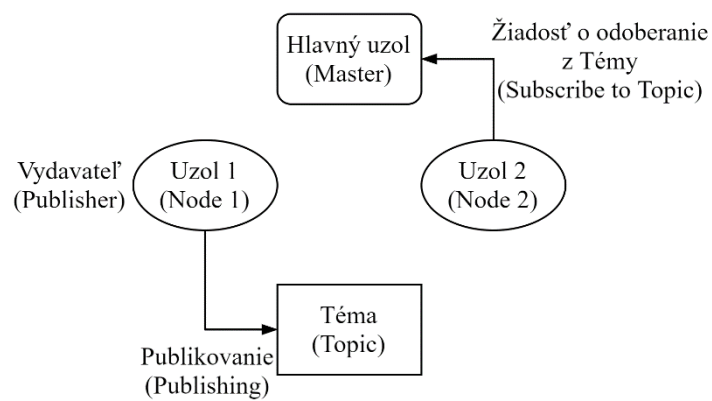
Odoberanie znamená sťahovanie správ z príslušnej témy. Uzol, ktorý odoberá, sa nazýva odoberateľ (subscriber).[5]

1.1.7 Ukážka medziprocesovej komunikácie cez témy

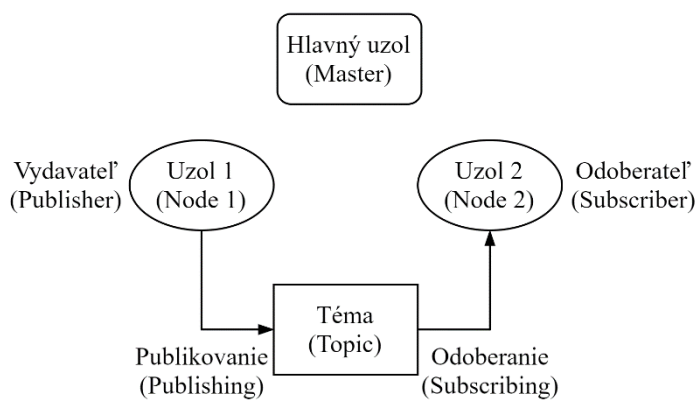
1. Uzol 1 si vyžiada možnosť publikovať na Tému od Hlavného uzla.



2. Hlavný uzol priradí možnosť Uzlu 1 publikovať na Tému. Uzol 2 požiada Hlavný uzol o odoberanie z Témy (Téma musí byť najskôr publikovaná). [5]



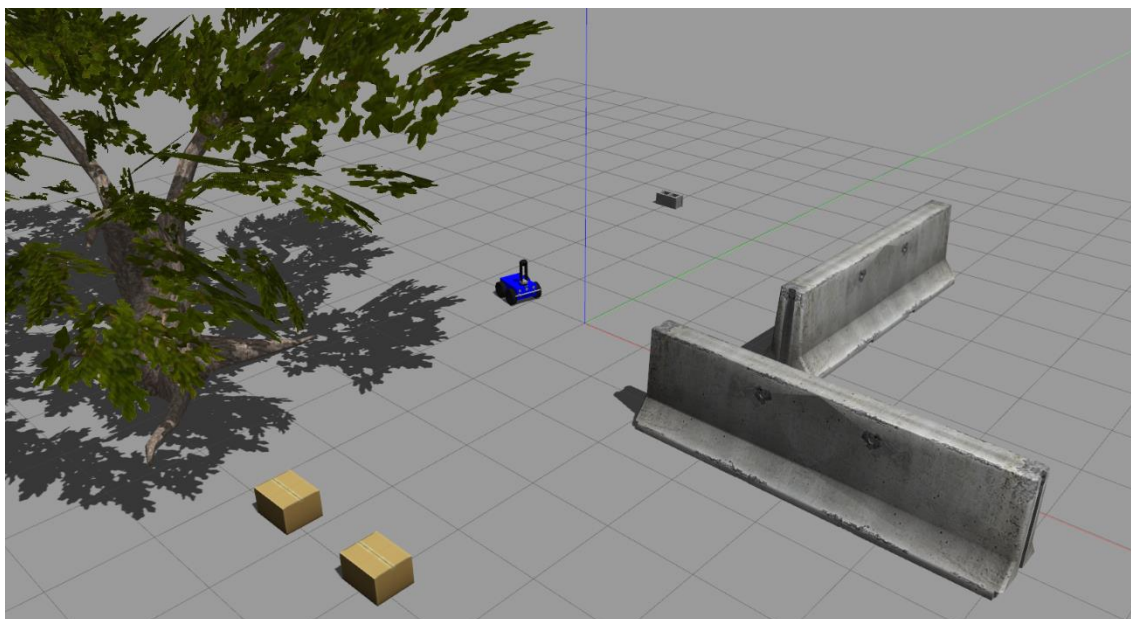
3. Hlavný uzol priradí možnosť odoberať z Témy pre Uzol 2. Uzol 1 a Uzol 2 vytvorili sieť so vzájomným sprístupňovaním (peer-to-peer sieť). [5]



2 Integrácia simulačného modelu robota do prostredia Gazebo

2.1 Simulačné prostredie Gazebo

Gazebo je voľne dostupné simulačné prostredie vyvinuté pre 3D simulovanie robotických aplikácií. Gazebo v sebe integruje fyzikálne engine-y (ODE, Bullet, Simbody, DART), OpenGL renderovanie a podporu pre simulovanie činnosti senzorov a riadenia aktuátorov. Od roku 2012 je Gazebo zastrešované organizáciou Open Robotics (organizácia poskytuje aj podporu pre ROS). Gazebo ponúka možnosť realistickej a efektívnej simulácie aj väčšieho počtu robotov v rozličných komplexných vonkajších, poprípade vnútorných prostrediach.[3]



Obr. 1: Ukážka modelu robota MRVK v prostredí Gazebo



Obr. 2: Ukážka modelu robota MRVK v simulovanej mape parku Baylands[11]

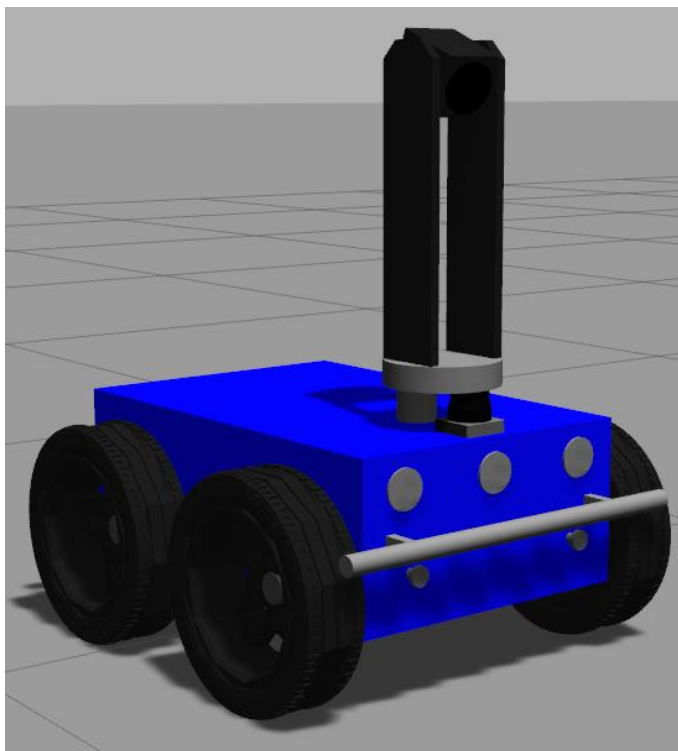


Obr. 3: Ukážka snímania cez kameru implementovanú do modelu robota MRVK v simulovanej mape parku Baylands[11]

2.2 Model robota

Simulačný model robota bol modelovaný podľa reálneho robota MRVK. Simulačný model robota je napísaný v jazyku XML, konkrétne bol použitý formát: Unified Robot Description Format (URDF). URDF je štandardizovaný formát jazyka

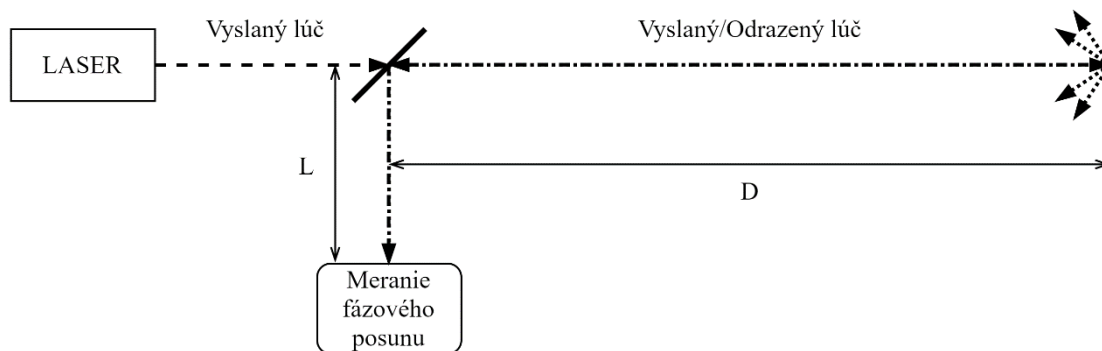
XML, ktorý sa v ROS používa na opis všetkých prvkov robota, z geometrického, vizuálneho, kinematického a dynamického hľadiska, v súčasnosti sa ale postupne stáva zastaraným a je postupne nahrádzaný formátom Simulation Description Format (SDF).[4]



Obr. 4: Vytvorený simulačný model robota MRVK

2.3 Laserový diaľkomer

Do modelu robota bol ako doplnok (plugin) implementovaný model skutočného LIDAR-u Hokuyo URG-04LX-UG01 Scanning Laser Range Finder. Laserový diaľkomer používa polovodičovú laserovú diódu, ktorá generuje infračervené žiarenie o vlnovej dĺžke $\lambda = 785 \text{ nm}$. Laserový diaľkomer pracuje na princípe merania fázového posunu. Spoľahlivý rozsah detekcie vzdialenosti garantovaný výrobcom je $D \in (0,06; 4) \text{ m}$. Laserový diaľkomer deteguje 2D priestor v rozsahu 240° s krokom $0,36^\circ$. [2]



Obr. 5: Princíp fungovania laserového diaľkomera merajúceho fázový posun signálu[1]

Vzdialenosť prekážky D je možné vypočítať:

$$D = \frac{ct}{2}$$

Kde t je doba letu lúča a c je rýchlosť svetla, $\frac{1}{2}$ kvôli prejdenu vzdialenosti D lúčom 2-krát. Dobu letu lúča je možné vypočítať za pomoci fázového posunu $\Delta\varphi$:

$$t = \frac{\Delta\varphi}{\omega}, \quad \omega = 2\pi f, \quad f = \frac{c}{\lambda}$$

Kde ω je uhlová frekvencia, f je frekvencia a λ je vlnová dĺžka vysielaného lúča. Vzdialenosť od prekážky je potom možné vypočítať v závislosti od fázového posunu:

$$D = \frac{c}{4\pi f} \Delta\varphi = \frac{\lambda}{4\pi} \Delta\varphi$$

Touto metódou je merací rozsah vzdialenosti obmedzený na $\frac{\lambda}{2}$, kvôli periodicky sa opakujúcemu fázovému posunu.[1]

Základná štruktúra:

Uzol:

- gazebo

Proces, na ktorom sa vykonáva snímanie laserovým diaľkomerom.

Publikované témy:

- scan (sensor_msgs/LaserScan)

Výstup z laserového diaľkomera, obsahuje základné informácie o laserovom diaľkomery a 2 výstupné 1D polia reprezentujúce meranie intenzity svetelného lúča a odmeranej vzdialenosti metódou fázového posunu.

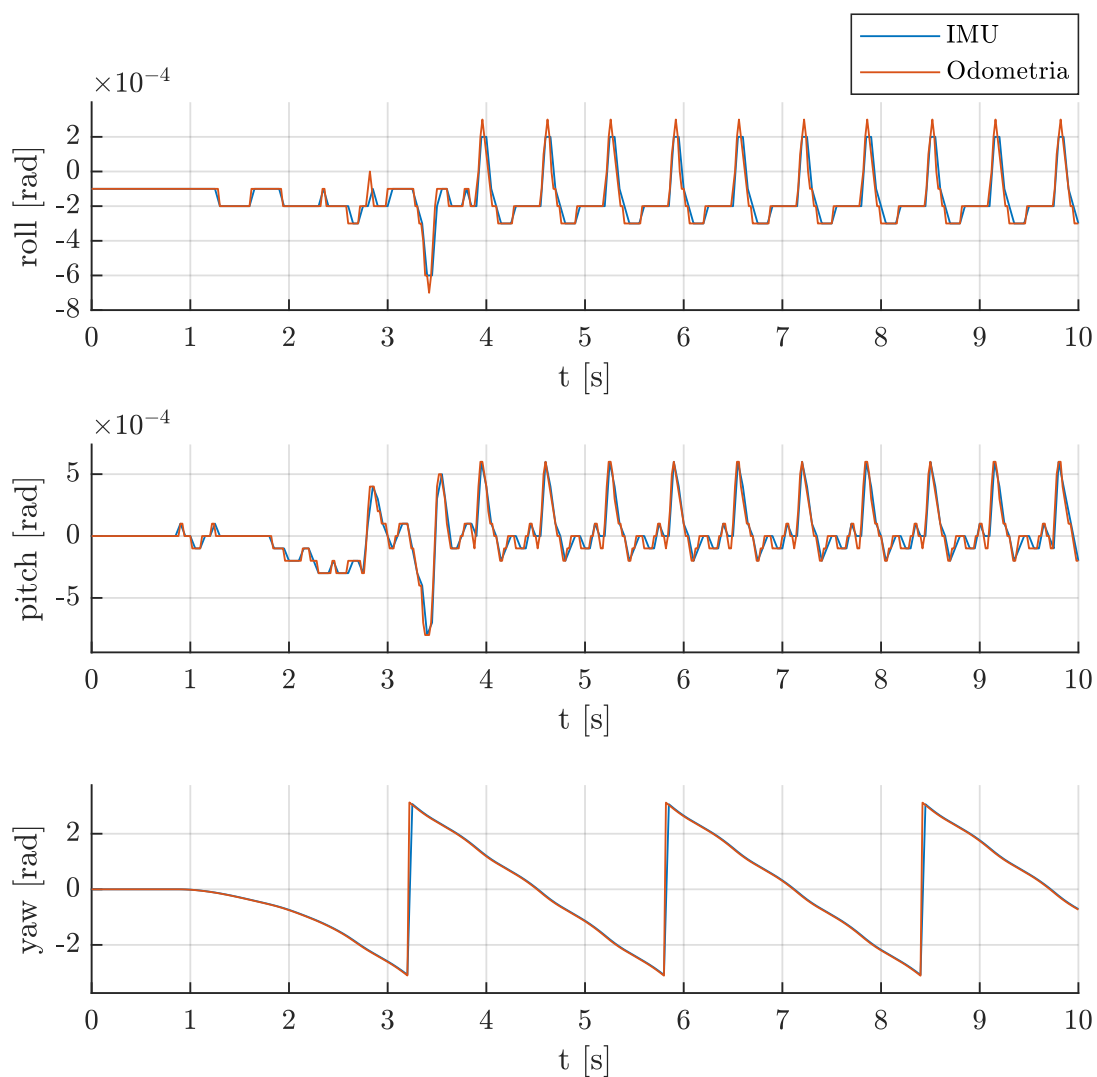
2.4 Inerciálna meracia jednotka (IMU)

Inerciálna meracia jednotka je zariadenie, ktoré sa skladá z gyroskopu, akcelerometra a prípadne magnetometra. Toto zariadenie dokáže merať rotačnú rýchlosť a lineárne zrýchlenie v osi x , y , z .

Určovať polohu na základe akcelerometra nie je vhodné pretože integrácia, by vnášala do odhadu polohy veľkú chybu. Preto IMU bolo používané hlavne na odhad uhlovej rýchlosti robota z ktorého sa počíta natočenie robota. Dáta z IMU jednotky je možné spájať s dátami z odometrie, pomocou aritmetického priemeru sa potom filtruje odhad polohy robota:

$$\mathbf{p} = \frac{\mathbf{p}_o + \mathbf{p}_{IMU}}{2}$$

Kde \mathbf{p} je výsledný vektor odhadu polohy, \mathbf{p}_o predstavuje odhad polohy z dát odometrie a \mathbf{p}_{IMU} je vektor dát z IMU.[1]



Obr. 6: Porovnanie priebehov veličín z IMU a odometrie pri rovnomernom otáčaní modelu robota MRVK okolo osi z (yaw) v prostredí Gazebo. Pre zrealnenie systému bol zavedený šum merania, ktorý je generovaný normálnym rozdelením s nulovou strednou hodnotou, pričom jeho rozptyl je nastavený na hodnotu $\sigma^2 = 2,89 \cdot 10^{-8} \text{ rad}^2 \cdot \text{s}^{-2}$. Parametre šumu je možné nastaviť v súbore `urdf/robot.plugins.xacro`, ktorý sa nachádza v balíčku `mrvk_description`.

2.5 Navigácia

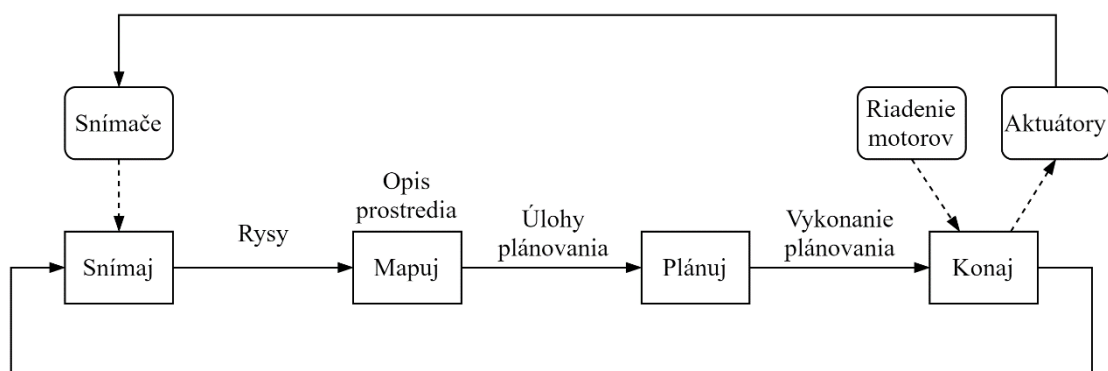
2.5.1 Globálna navigácia

Súbor úloh, ktorých účelom je nájsť dráhu medzi štartom a cieľom na základe stanovených kritérií alebo podmienok.[1]

2.5.2 Reaktívna navigácia

Navigácia, ktorá predpisuje robotu bezkolízne správanie sa na základe aktuálnych vnemov a uvažujúca s rozmermi a dynamickými obmedzeniami robota. Reaktívna navigácia je inšpirovaná biologickými princípmi:

- Snímaj (Sense)
- Mapuj (Map)
- Plánuj (Plan)
- Konaj (Act)



Obr. 7: Schematické zobrazenie SMPA princípu[1]

2.5.3 Balík move_base

Balík move_base poskytuje bezkolíznú navigáciu na používateľom určený cieľ v prostredí, v ktorom sa robot nachádza. Balík na dosiahnutie cieľa bezkolíznej globálnej navigácie používa súčinnosť lokálneho a globálneho plánovanie pohybu. Na nájdenie trajektórie sa používajú lokálne a globálne metrické účelové mapy (local costmap/global costmap), ktoré obsahujú ohodnotené bunky. Ohodnotené bunky označujú miesta, ktorým sa robot má vyhnúť, čím vyššia hodnota tým viac sa danej bunke robot snaží vyhýbať. Následne sa zvoleným prehľadávajúcim algoritmom, podľa hodnoty jednotlivých buniek, vygeneruje trajektória. V tejto práci bol na hľadanie trajektórie

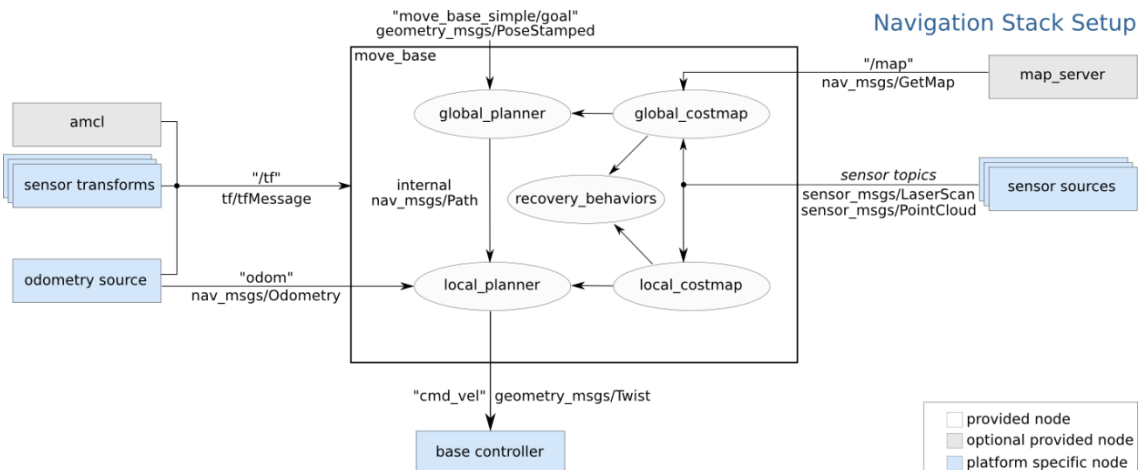
zvolený Dijkstrov algoritmus, dosiahnuteľnosť nájdených bodov trajektórie je potom riešená metódou dynamického okna.[6]

Základná štruktúra:

Uzol:

- move_base

Proces zabezpečujúci bezkolíznú navigáciu robota v prostredí.



Obr. 8: Schematické zobrazenie uzla move_base z navigačného balíka[6]

Akcie:

Akcie odoberajúce témy:

- move_base/goal (move_base_msgs/MoveBaseActionGoal)
Cieľ pre move_base, ktorý sa snaží dosiahnuť.
- move_base/cancel (actionlib_msgs/GoalID)
Žiadosť o zrušenie daného cieľa.

Akcie publikujúce témy:

- move_base/feedback (move_base_msgs/MoveBaseActionFeedback)
Spätná väzba obsahujúca aktuálnu pozíciu základne robota.
- move_base/status (actionlib_msgs/GoalStatusArray)
Poskytuje informácie o stave vykonávania cieľov.
- move_base/result (move_base_msgs/MoveBaseActionResult)
Prázdna správa pre move_base akciu.

Odoberané témy:

- `move_base_simple/goal` (`geometry_msgs/PoseStamped`)
Používateľské rozhranie na nastavenie cieľa, nasleduje sa stav vykonávania plánu.

Publikované témy:

- `cmd_vel` (`geometry_msgs/Twist`)
Príkazy zmeny rýchlosti pre robota.

Služby:

- `~make_plan` (`nav_msgs/GetPlan`)
Vyžiadanie plánu trajektórie, bez vykonania daného plánu.
- `~clear_unknown_space` (`std_srvs/Empty`)
Vymazanie priestoru v blízkom okolí robota.
- `~clear_costmaps` (`std_srvs/Empty`)
Vymazanie prekážok z účelovej mapy.

2.5.4 Metóda dynamického okna

Balík `move_base` poskytuje dve možnosti lokálnej bezkolíznej navigácie: Trajectory rollout a Metódu dynamického okna (Dynamic Window Approach – DWA). Obidve metódy pracujú na princípe prediktívneho riadenia. Cieľom je nájsť najvhodnejšiu kombináciu rýchlosti (ω , v), simulované sú rôzne kombinácie rýchlostí a následne účelovou funkciou sú jednotlivé simulácie vyhodnotené a je vybraná najlepšia kombinácia (ω , v). Rozdiel v týchto dvoch metódach spočíva v dobe prediktívnej simulácie. Trajectory rollout simuluje viacero simulačných krokov dopredu, DWA simuluje len jeden simulačný krok. Obidve metódy zvyčajne dosahujú obdobné výsledky, DWA je však menej výpočtovo náročná, v práci ako reaktívna navigácia bola zvolená práve DWA.[1],[6],[7]

Množiny obmedzení rýchlostí

DWA potrebuje generovať kombinácie rýchlostí z nejakého pracovného priestoru, aby metóda pracovala správne je nutné zmenšiť tento pracovný priestor na prijateľné dosiahnuteľné rýchlosti. Redukcia pracovného priestoru prebieha vyhodnotením niekoľkých množín:

- **Množina V_s** – obmedzenie od maximálnych dosiahnuteľných rýchlostí robota (ω_{max}, v_{max}) , dané fyzikálnymi parametrami robota (výkon motorov, rozmery, hmotnosť, moment zotrvačnosti).[1]
- **Množina V_a** – obmedzenie od prijateľných rýchlostí robota, vzhľadom na prekážky, také dvojice (ω, v) , pri ktorých robot dokáže bezpečne zastaviť pred prekážkami pri pohybe po dráhovom oblúku daným rýchlosťami (ω, v) .

$$V_a = \{(\omega, v) | \omega \leq \sqrt{2r(\omega, v)\varphi(\omega, v)\dot{\omega}} \wedge v \leq \sqrt{2r(\omega, v)\varphi(\omega, v)\dot{v}}\}$$

Kde $r = \frac{v}{\omega}$ je polomer dráhy, φ je opísaný uhol a $(\dot{\omega}, \dot{v})$ predstavujú zrýchlenia pri brzdení.[1]

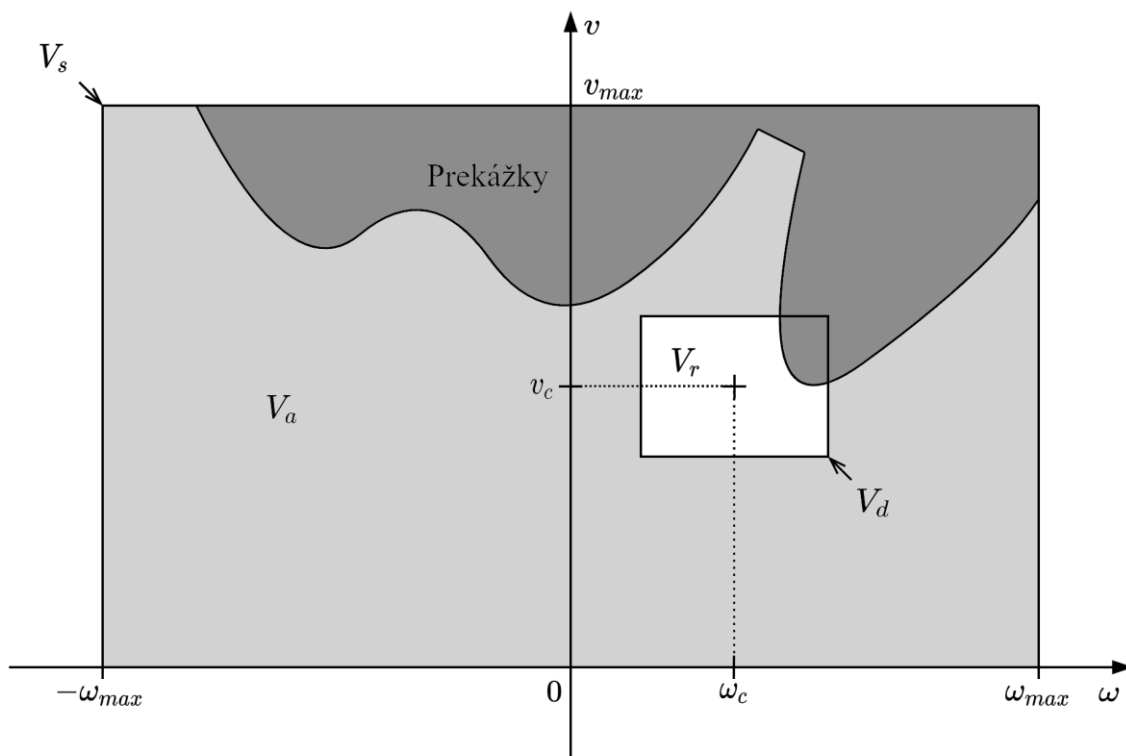
- **Množina V_d** – obmedzenie od dosiahnuteľných rýchlostí v zvolenom časovom úseku Δt , dané dynamikou robota.

$$V_d = \{(\omega, v) | \omega \in \langle \omega_c - \dot{\omega}\Delta t; \omega_c + \dot{\omega}\Delta t \rangle \wedge v \in \langle v_c - \dot{v}\Delta t; v_c + \dot{v}\Delta t \rangle\}$$

Kde Δt je zvolený krok numerickej simulácie, (ω_c, v_c) predstavuje aktuálne rýchlosti a $(\dot{\omega}, \dot{v})$ sú dosiahnuteľné zrýchlenia.[1]

- **Množina V_r** – vznikne prienikom všetkých vymenovaných množín a predstavuje prijateľné dosiahnuteľné rýchlosti (ω, v) . [1]

$$V_r = V_s \cap V_a \cap V_d$$



Obr. 9: Ukážka priestoru rýchlostí (ω, v) v metóde dynamického okna, s vyznačenými množinami obmedzení rýchlostí. Kde (ω_c, v_c) predstavujú aktuálne rýchlosti robota.[1].

Účelová funkcia

Cieľom je nájsť najlepšiu kombináciu (ω, v) z množiny rýchlostí V_r maximalizáciou účelovej funkcie, napríklad v tvare:

$$J(\omega, v) = c_1 e_\varphi(\omega, v) + c_2 v_T(\omega, v) + c_3 r(\omega, v) \varphi(\omega, v)$$

Kde e_φ vyjadruje odchýlku od smeru k cieľu, v_T translačnú rýchlosť pohybu, r je polomer dráhy a φ je opísaný uhol, koeficienty c sú voliteľné váhy jednotlivých členov, pomocou nich sa dá nastaviť správanie robota.[1]

2.6 Lokalizácia a mapovanie

Simultaneous Localization And Mapping (SLAM) – Súčasná lokalizácia a mapovanie predstavuje koncept, v ktorom sa predpokladá, že robot nie je schopný určiť svoju polohu a polohu značiek prostredia presne, môže ale spresniť tieto odhady polohy. Merania sú zaťažené chybou, preto sa predpokladá poloha robota a značiek prostredia na základe pravdepodobnosti určenej neistotou snímačov. Robot súčasnou lokalizáciou

a mapovaním dokáže zmenšovať neistoty merania a tým upresňovať svoju polohu a polohu značiek prostredia.[1]

2.6.1 Balík gmapping

V simulácii bol SLAM implementovaný do modelu robota cez balík gmapping. Daný balík umožňuje vytváranie 2D máp prostredia, na základe údajov z laserového diaľkomera.[9]

Základná štruktúra:

Uzol:

- `slam_gmapping`
Proces, na ktorom sa vykonáva SLAM.

Odoberané témy:

- `tf (tf/tfMessages)`
Transformácia zo súradných systémov laserového diaľkomera, základne robota a odometrie do spoločného súradného systému.
- `scan (sensor_msgs/LaserScan)`
Údaje z laserového diaľkomera na vytvorenie mapy.

Publikované témy:

- `map_metadata (nav_msgs/MapMetaData)`
Charakteristické dáta o výstupnej mape, periodicky aktualizované.
- `map (nav_msgs/OccupancyGrid)`
Výstupná mapa, reprezentovaná ako mriežka s okupovanými bunkami, periodicky aktualizované.
- `~entropy (std_msgs/Float64)`
Odhad informačnej entropie polohy robota (vyššia hodnota predstavuje vyššiu neistotu určenia polohy robota).

Služby:

- `dynamic_map (nav_msgs/GetMap)`
Služba na získanie dát mapy.

2.6.2 Príklad algoritmu SLAM

1. **Inicializácia** – počiatočný odhad polohy robota.
2. **Meranie** – robot skenuje prostredie snímačom prostredia (napr. LIDAR).
3. **Aktualizácia** – robot aktualizuje svoju polohu a mapu prostredia, poloha značiek prostredia je určená s neistotou snímača prostredia a neistotou hrubého odhadu polohy robota. Pokiaľ robot znova deteguje známe značky prostredia dochádza k zmenšeniu neistoty určenia polohy (snímače prostredia bývajú presnejšie ako hrubý odhad polohy robota).
4. **Predikcia** – hrubý odhad polohy robota (napr. odometria), s pohybom robota rastie neistota v určení polohy. Pokračuje sa bodom 2.[1]

2.7 Vizuálny systém na sledovanie cesty

2.7.1 Kamera

Vizuálny systém v projekte bol tvorený doplnkom (plugin) kamery do simulačného prostredia Gazebo.

Základná štruktúra:

Uzol:

- `image_view`
Proces, na ktorom sa vykonáva snímanie kamerou.

Publikované témy:

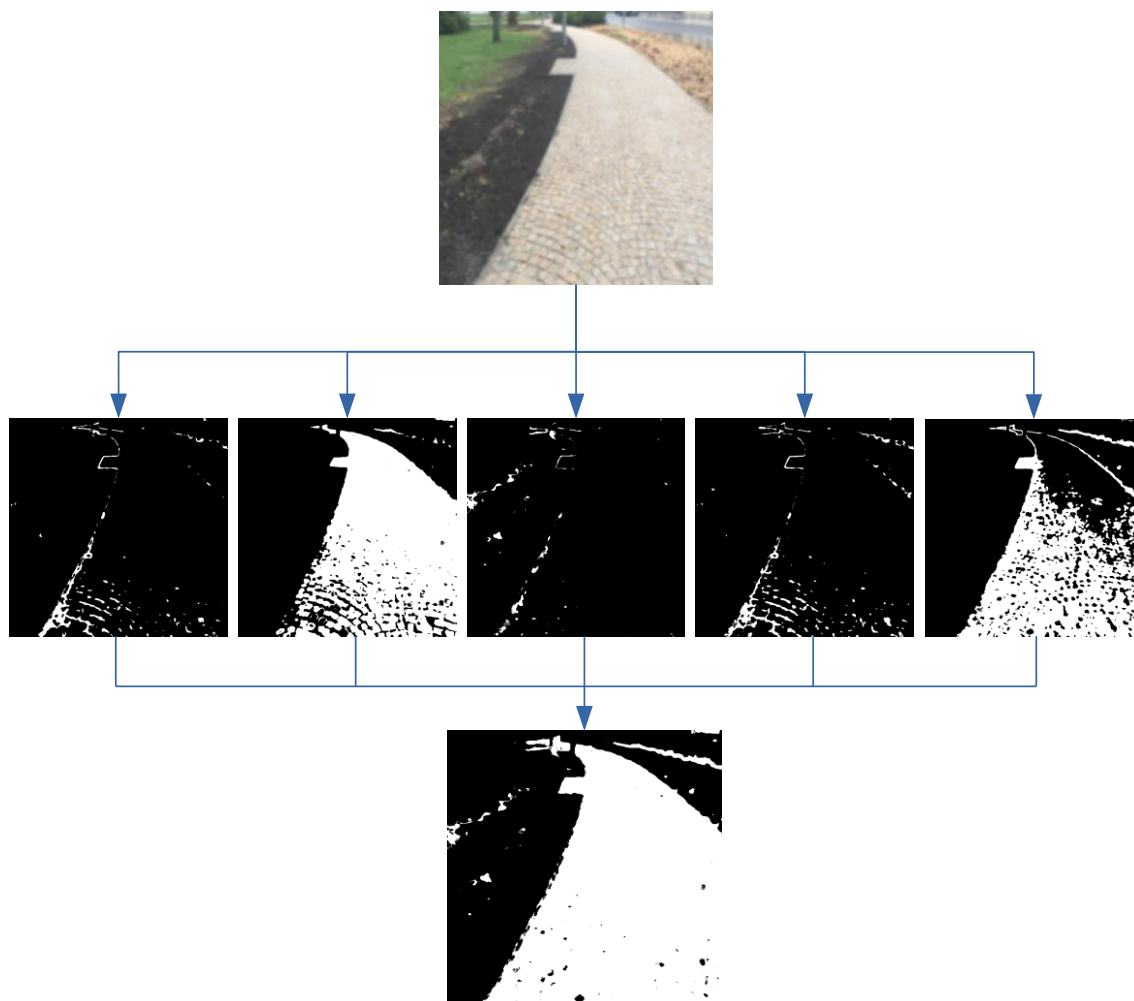
- `image_raw (sensor_msgs/Image)`
Výstup z kamery, obraz vo forme 1D poľa, reprezentujúci RGB farebnú sústavu.
- `camera_info (sensor_msgs/CameraInfo)`
Údaje o kamere.

2.7.2 Detekcia chodníka

Pred samotným spracovaním obrazovej informácie sa vykonáva transformácia z 1D poľa na 3D maticu reprezentujúcu BGR obraz spracovateľný prostredníctvom OpenCV. Následne prebehne filtrácia obrazu využitím Gaussovho filtra s veľkosťou jadra 3x3.

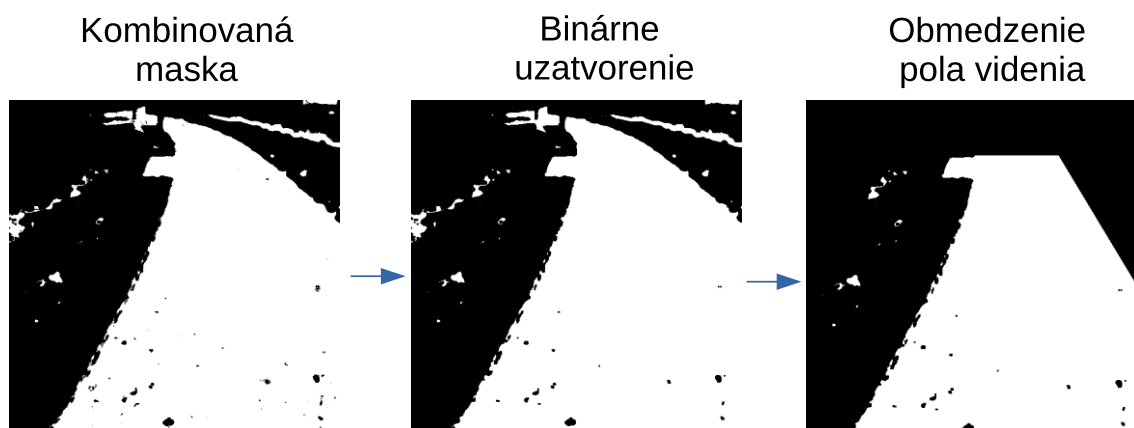
Pred ďalším spracovaním obrazovej informácie je potrebné vykonať inicializáciu. Inicializácia spočíva v identifikovaní stanoveného počtu (N) dominantných farieb chodníka po ktorom sa má robot pohybovať. Nakoľko je našim cieľom detegovať chodník, ktorého možnú polohu v obraze vieme očakávať. Je v inicializácii taktiež vytvorená aj lichobežníková maska predstavujúca obmedzené pole videnia robota. Inicializácia sa vykonáva za predpokladu, že robot sa nachádza na chodníku pričom jeho natočenie umožňuje, že dolná spodná časť obrazu z kamery zachytáva len chodník, po ktorom sa má pohybovať. Uvedená časť obrazu sa pri inicializácii extrahuje a identifikuje sa v nej stanovený počet (N) dominantných farieb chodníka využitím metódy zhlukovej analýzy K-means. Nájdene dominantné farby sú ďalej prevedené z BGR farebnej sústavy do sústavy HSV. Nakoniec sa na základe stanovených prípustných odchýlok jednotlivých zložiek HSV modelu pre každú dominantnú farbu definujú dolné a horné prahové hodnoty predstavujúce prípustné pásmo farieb chodníka.

Po ukončení inicializácie nasleduje priebežná detekcia chodníka vykonávaná po príchode vizuálnych dát na odoberanú tému. Ďalšie spracovanie filtrovaného obrazu spočíva v jeho prevedení z BGR farebného modelu do HSV. Na základe v inicializácii stanovených prahových hodnôt prípustných farieb chodníka sa ďalej vytvorí N masiek, pričom každá odpovedá jednej dominantnej farbe chodníka. Logickým súčtom uvedených masiek je vytvorená kombinovaná maska.



Obr. 10: Získanie kombinovanej masky z filtrovaného obrazu

Kombinovaná maska je ďalej spracovaná binárnym uzavretím. Následne sa vykoná logický súčin morfológicky transformovanej kombinovanej masky a masky predstavujúcej obmedzené pole videnia robota.



Obr. 11: Spracovanie kombinovanej masky binárnym uzavretím a obmedzením poľa videnia

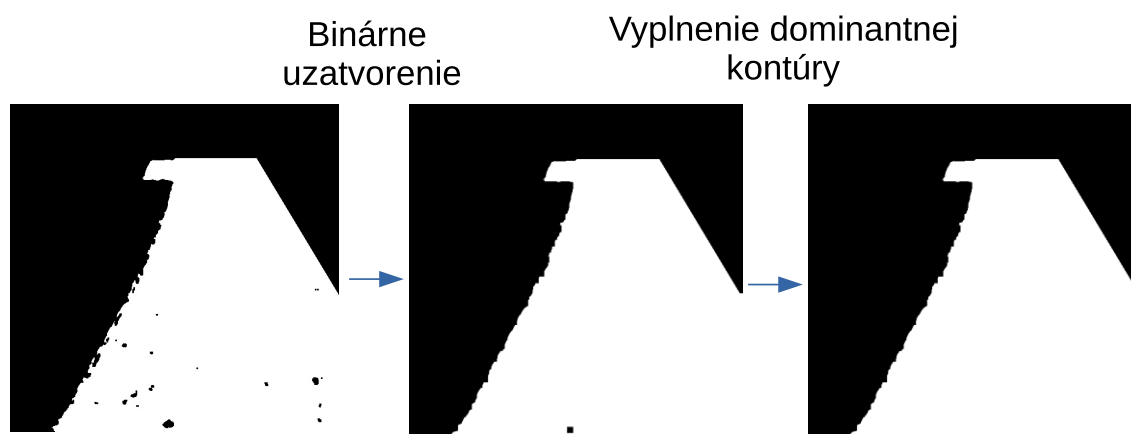
V maske sa ďalej odstránia nesprávne detegované zóny v okolí chodníka a taktiež aj detegované plochy chodníka, ktoré nie sú z aktuálnej pozície dosiahnuteľné. Uvedené sa dosiahne nájdením kontúry s najväčšou plochou a odstránením ostatných kontúr.

Ponechanie dominantnej kontúry



Obr. 12: Ponechanie kontúry s najväčšou plochou a odstránenie ostatných kontúr

Pokračuje sa vykonaním ďalšieho binárneho uzatvorenia (s väčšou veľkosťou jadra). Posledné spracovanie predstavuje vyplnenie kontúry s najväčšou plochou, čím je získaná výsledná maska reprezentujúca detegovaný chodník.



Obr. 13: Spracovanie masky binárnym uzatvorením a vyplnenie kontúry s najväčšou plochou

Porovnanie masky reprezentujúcej detegovaný chodník a obrazu chodníka je na Obr. 14. Z výsledkov vyplýva, že v tomto prípade bol chodník naozaj úspešne detegovaný, a to aj napriek tomu, že spracovanie obrazu prebiehalo v inom mieste ako inicializácia. Aby bolo

možné dobre ukázať jednotlivé kroky spracovania obrazu, bola pre znázornenie vybraná snímka, kedy robot nie je natočený priamo na chodník.



Obr. 14: Overenie detekcie chodníka

Základná štruktúra:

Uzol:

- `path_detection`

Proces zabezpečujúci spracovanie obrazu a detekciu chodníka.

Odoberané témy:

- `image_raw` (`sensor_msgs/Image`)

Výstup z kamery, obraz vo forme 1D poľa, reprezentujúci RGB farebnú sústavu.

Publikované témy:

- `cmd_vel` (`geometry_msgs/Twist`)

Príkazy zmeny rýchlosti pre robota.

- `detected_path` (`path_detection_msgs/DetectedPath`)

Detegovaný chodník vo forme 1D binárneho poľa.

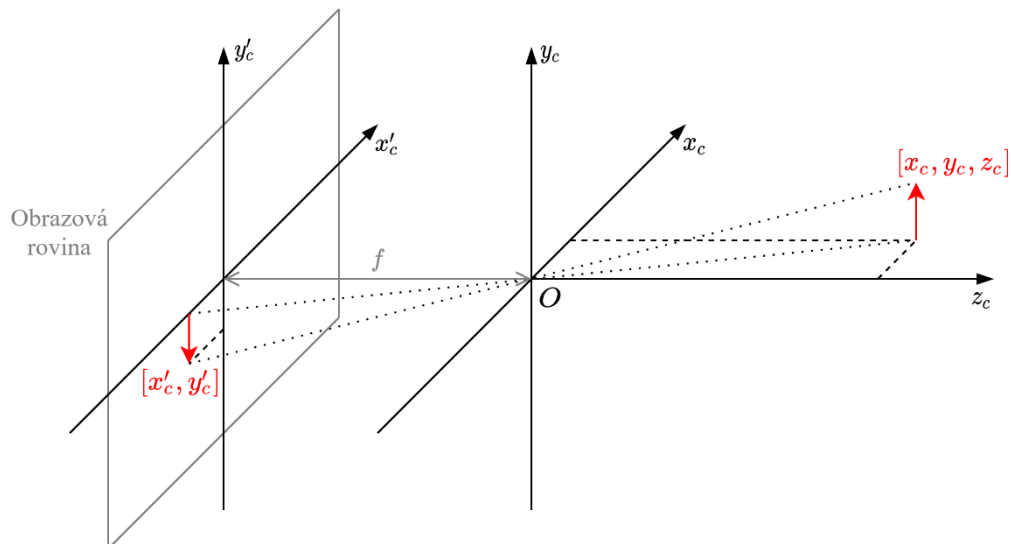
2.7.3 Pridanie novej vrstvy do účelových máp

Účelová mapa (Costmap)

Účelová mapa je zložená z viacerých prekrývajúcich sa vrstiev. Na každej vrstve sú jednotlivé bunky mapy ohodnotené pokutami v závislosti od výskytu prekážok v pohybe robota. Jednotlivé vrstvy, z ktorých sa skladajú účelové mapy je možné rozdeliť do 4 základných skupín:

- **Statická vrstva (Static Map Layer)** – obsahuje nemeniace sa prostredie (napr.: predpripravená mapa so statickými prekážkami).
- **Vrstva prekážok (Obstacle Map layer)** – vrstva obsahuje zmapované prekážky na základe senzorov robota, bunky sú ohodnotené mŕtvymi pokutami.
- **Inflačná vrstva (Inflation Map Layer)** – nafúknutie prekážok na základe zvolených parametrov, hodnoty buniek klesajú exponenciálne so vzdialenosťou od prekážky.
- **Doplnkové vrstvy (Other Layers)** – rozličné vrstvy, patria sem aj nové vrstvy definované používateľom.[8]

Perspektívna transformácia



Obr. 15: Model ideálnej dierkovej kamery (O predstavuje stred projekcie)

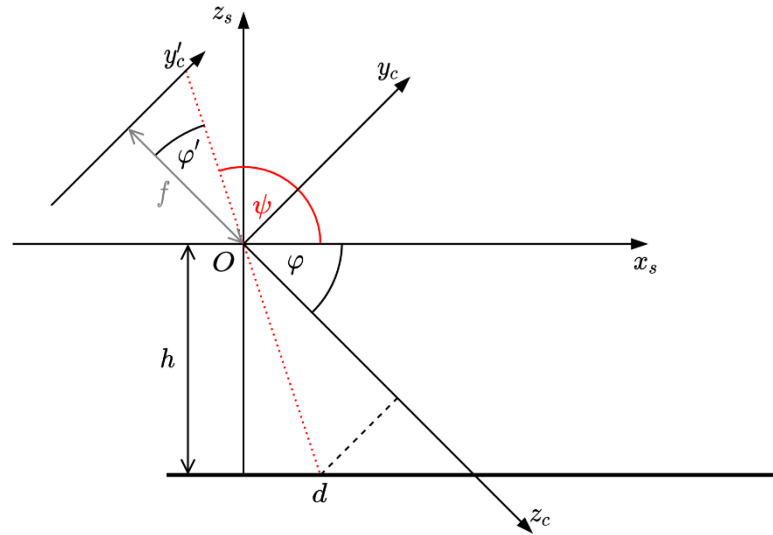
Z uvedeného obrázka, na základe podobnosti trojuholníkov, pre ideálnu dierkovú kameru, je možné odvodiť transformačné rovnice:

$$x'_c = -f \frac{x_c}{z_c}$$

$$y'_c = -f \frac{y_c}{z_c}$$

Kde f je ohnisková vzdialenosť, $[x_c, y_c, z_c]$ sú súradnice objektu v súradnicovom systéme kamery a $[x'_c, y'_c]$ sú súradnice objektu v obrazovej rovine. Transformácia mapuje 3D priestor do 2D roviny, stráca sa tak informácia o jednej súradnici.

Spätná transformácia z 2D roviny do 3D priestoru, tak ostáva neúplná, uvažovaním niekoľkých predpokladov ale je možné premietnuť detegovanú cestu kamerou do svetového súradnicového systému $[x, y, z]$. Zvolia sa 3 doplnkové súradné systémy: súradný systém obrazovej roviny $[x'_c, y'_c]$, súradný systém kamery $[x_c, y_c, z_c]$ a súradný systém naň natočený $[x_s, y_s, z_s]$. Predpokladá sa, že kamera je umiestnená v známej výške h so známym sklonom smerom na cestu φ . Ďalej sa predpokladá, že cesta je rovná, teda sa predpokladá, že všetky body na ceste majú rovnakú z_s súradnicu, to znamená, že súradnica y'_c je potom závislá len od súradnice x_s .



Obr. 16: Bokorys na zvolené súradné systémy (O predstavuje stred projekcie)

Bod d je tvorený priesečníkom červenej priamky a cesty, v súradnom systéme $[x_s, z_s]$ pre červenú priamku a cestu platí:

$$z_s = -\tan(\psi)x_s$$

$$z_s = -h$$

Z toho vyplýva:

$$x_s = -\frac{h}{\tan(\psi)}$$

Pre uhol ψ platí:

$$\psi = \pi - \varphi - \varphi'$$

$$\varphi' = \arctan\left(\frac{y'_c}{f}\right)$$

Súradnica y_s sa určí na základe perspektívnej transformácie:

$$y_s = -\frac{z_c}{f} x'_c$$

Pričom systémy $[x_s, z_s]$ a $[z_c, y_c]$ sú na seba natočené o uhol φ :

$$z_c = x_s \cos(\varphi) + h \sin(\varphi)$$

$$y_c = x_s \sin(\varphi) - h \cos(\varphi)$$

Nakoniec treba určiť transformáciu do svetového súradnicového systému $[x, y, z]$:

$$x = x_r + (x_{cT} + x_s) \cos(\varphi_r) - y_s \sin(\varphi_r)$$

$$y = y_r + (x_{cT} + x_s) \sin(\varphi_r) + y_s \cos(\varphi_r)$$

Kde $[x_r, y_r, \varphi_r]$ predstavujú pozíciu a natočenie robota, x_{cT} je pozícia kamery v osi x vzhľadom na stred robota.

Kalibrácia kamery

V reálnom systéme by bolo potreba určiť celú transformačnú maticu kamery, pre potreby simulácie je ale postačujúce určiť len ohniskovú vzdialenosť. Bude sa predpokladať, že bod d má známu polohu vzhľadom na kameru, potom je možné f vypočítať z perspektívnej transformácie, pričom bod d , so zložkami $[d, -h]$ v sústave $[x_s, z_s]$: sa premietne rotáciou o uhol φ do sústavy $[z_c, y_c]$:

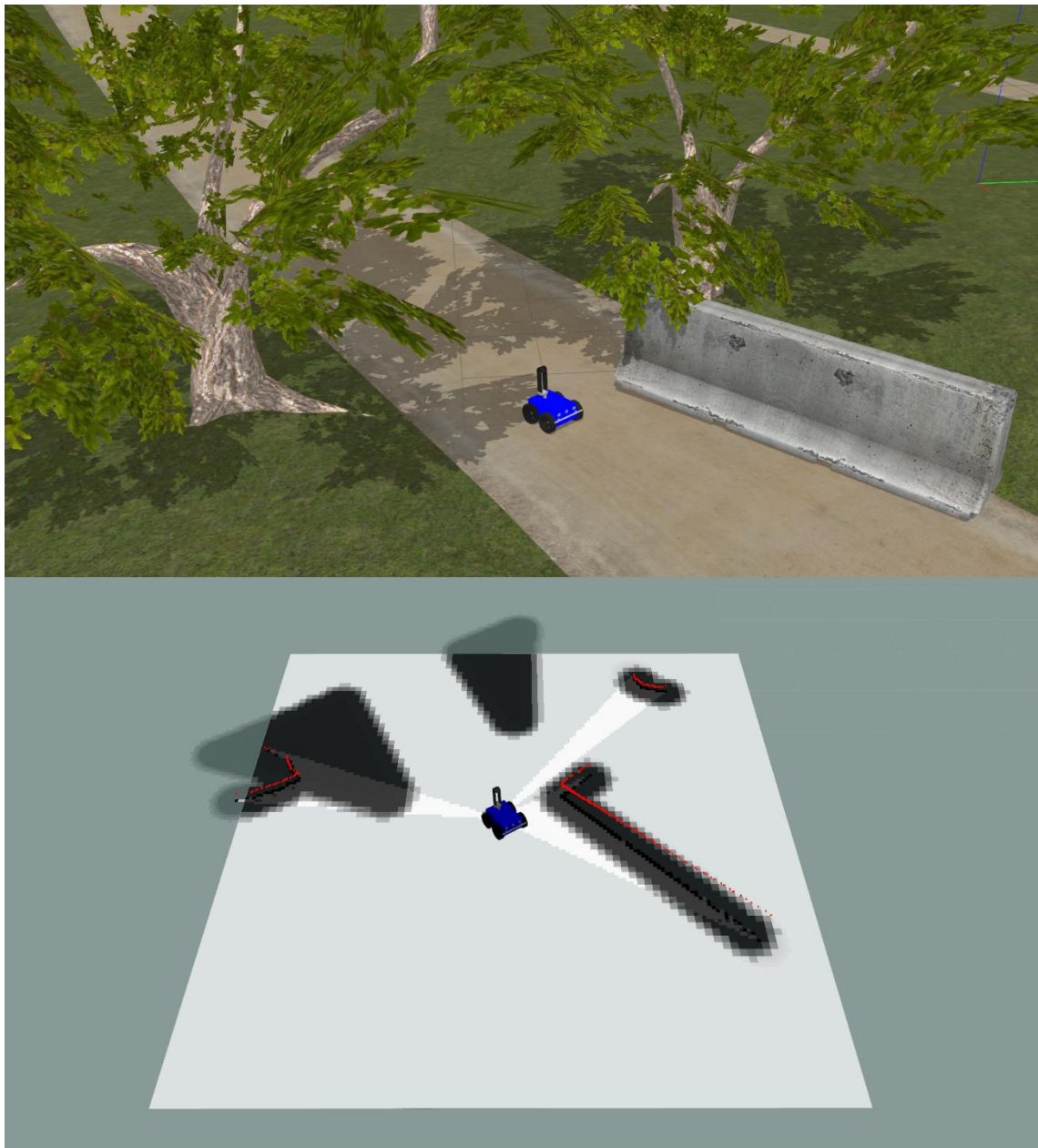
$$z_c = d \cos(\varphi) + h \sin(\varphi)$$

$$y_c = d \sin(\varphi) - h \cos(\varphi)$$

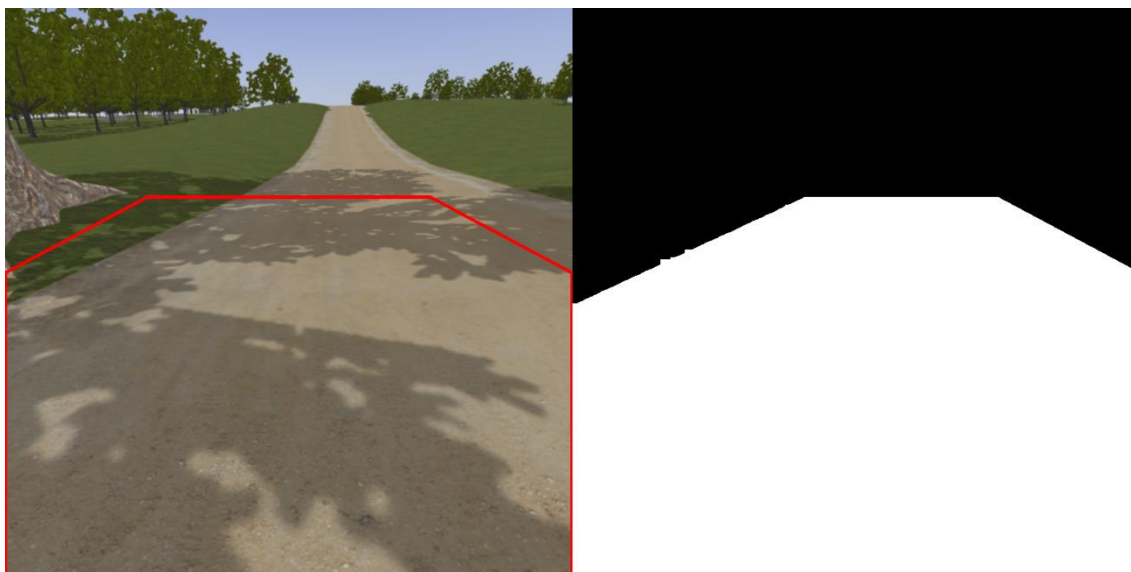
$$f = -\frac{z_c}{y_c} y'_c$$

Bod d je kvôli presnosti lepšie voliť blízko od kamery vzhľadom na os x_s , pretože so zvyšujúcou sa súradnicou x_s klesá rozlišovacia schopnosť kamery, samozrejme pre spresnenie výsledkov je vhodné vykonať viacero meraní.

Implementácia novej vrstvy



Obr. 17: Horný obrázok vyobrazuje robota v prostredí Gazebo, dolný obrázok je z prostredia RViz a predstavuje svet z pohľadu robota. Biely štvorec predstavuje lokálnu účelovú mapu, v ktorej je možné vidieť spojenie vrstvy prekážok, doplnkovej vrstvy (transformácia zo snímok kamery) a inflačnej vrstvy (prekážky sú nafúknuté). Červené body predstavujú prekážky zachytené LIDAR-om, tmavé trojuholníky, nachádzajúce sa pred robotom, predstavujú okraje cesty transformované z kamery do lokálnej účelovej mapy.



Obr. 18: Ilustrácia snímania kamerou. Cesta sa deteguje len v červenom mnohoúhľníku, cieľom bolo aby sa obmedzilo detegovanie falošných ciest, skosenie mnohoúhľníka v hornej časti, má prispôbiť vnímanie na základe perspektívnej transformácie, ďalej sa ním má obmedziť možné oscilovanie pri riadení.

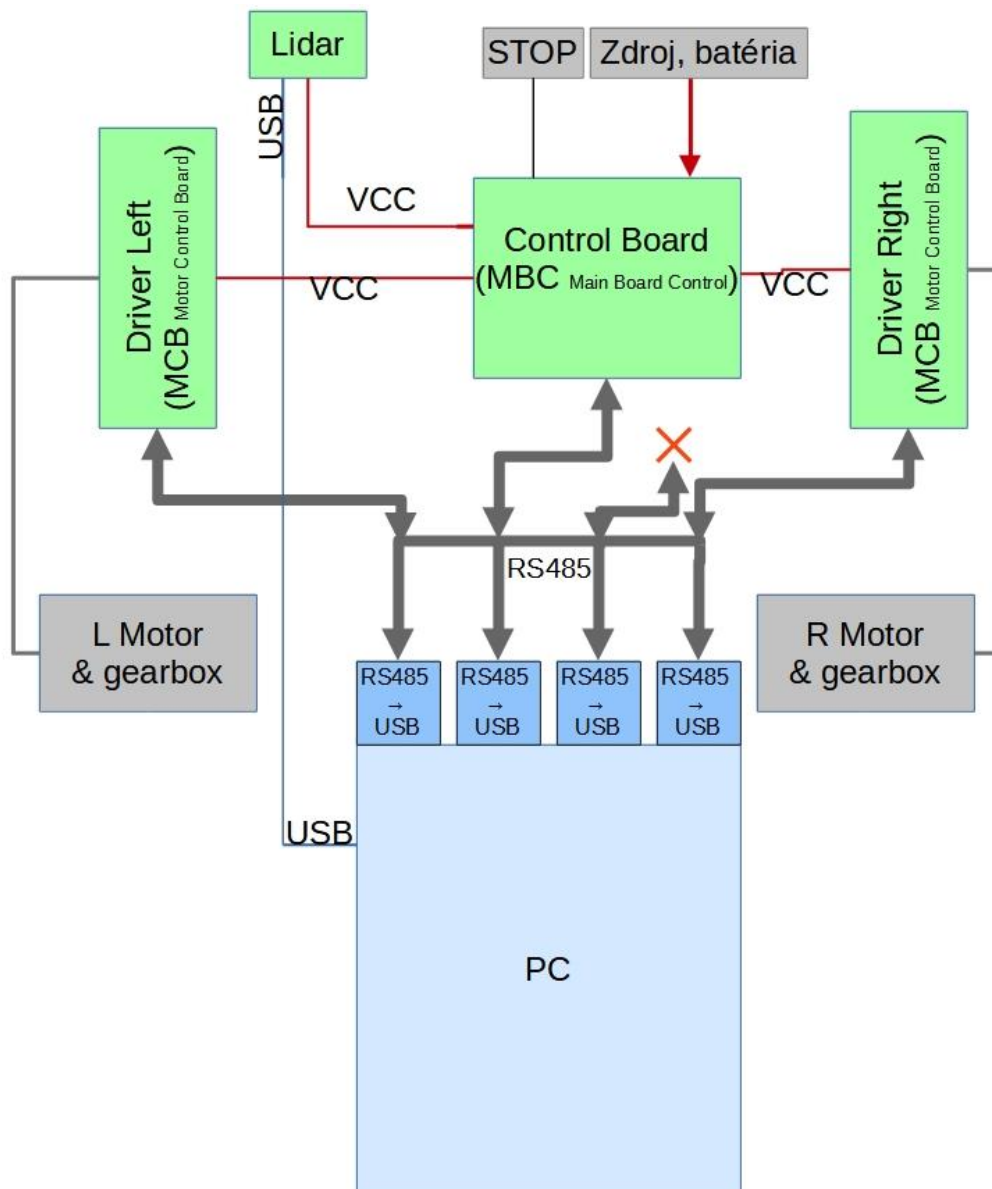
3 Analýza robota MRVK



Obr. 19: Robot MRVK

3.1 Hardvér

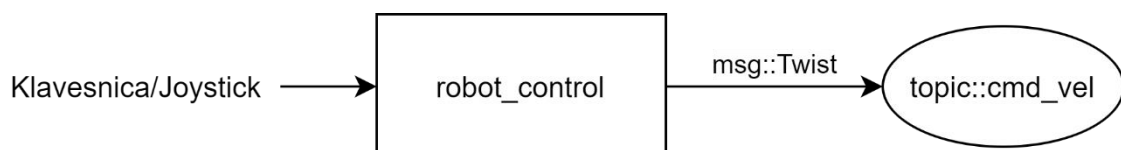
Robot MRVK je štvorkolesový mobilný robot s diferenciálnym spriahnutým podvozkom, ktorý je poháňaný dvojicou DC motorov cez prevodovku. O riadenie motorov sa starajú budiče označené ako MCB (pozri Obr. 20). Budiče spolu s PC a hlavnou riadiacou doskou (MBC), sú vzájomne prepojené cez sériovú zbernicu RS485. MBC sa okrem iného stará aj o distribúciu napájania pre jednotlivé komponenty robota.



Obr. 20: Schematické znázornenie štruktúry hardvéru robota MRVK

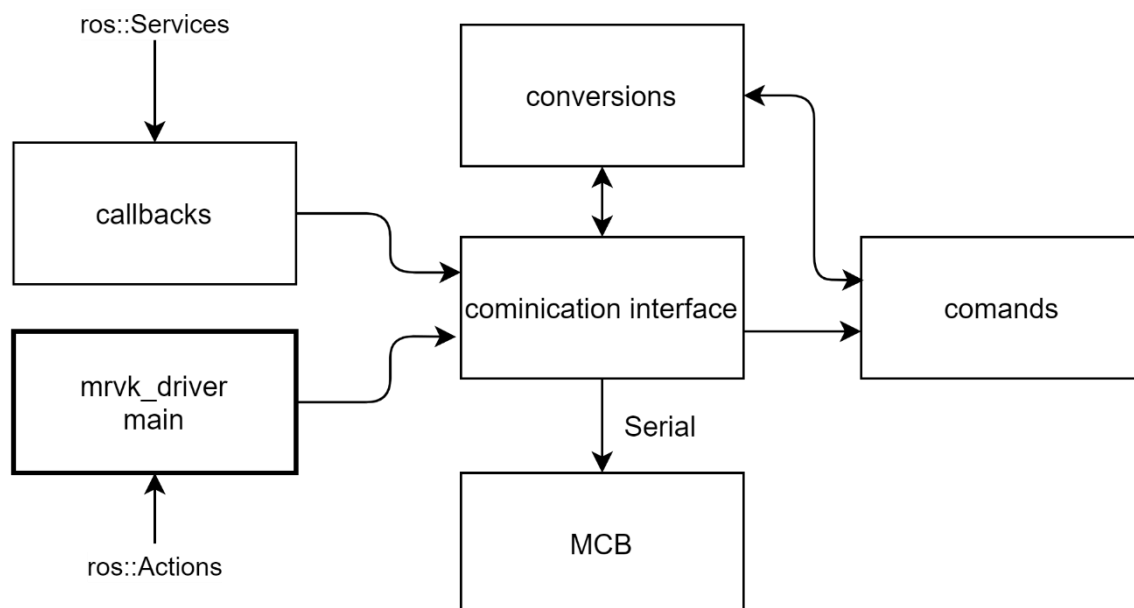
3.2 Softvér

Riadenie robota MRVK je možné dvoma spôsobmi. V prvom je robot ovládaný pomocou konzolového rozhrania. V tomto rozhraní môže posielať príkazy pre robota, ovládať ho pomocou klávesnice/joysticku.



Obr. 21: Schematické znázornenie štruktúry riadenia robota cez konzolu

V druhom prípade robot reaguje na akcie z ROS-u ktoré spracováva a následne posiela príkazy cez sériovú linku pre Hlavnú riadiacu dosku.

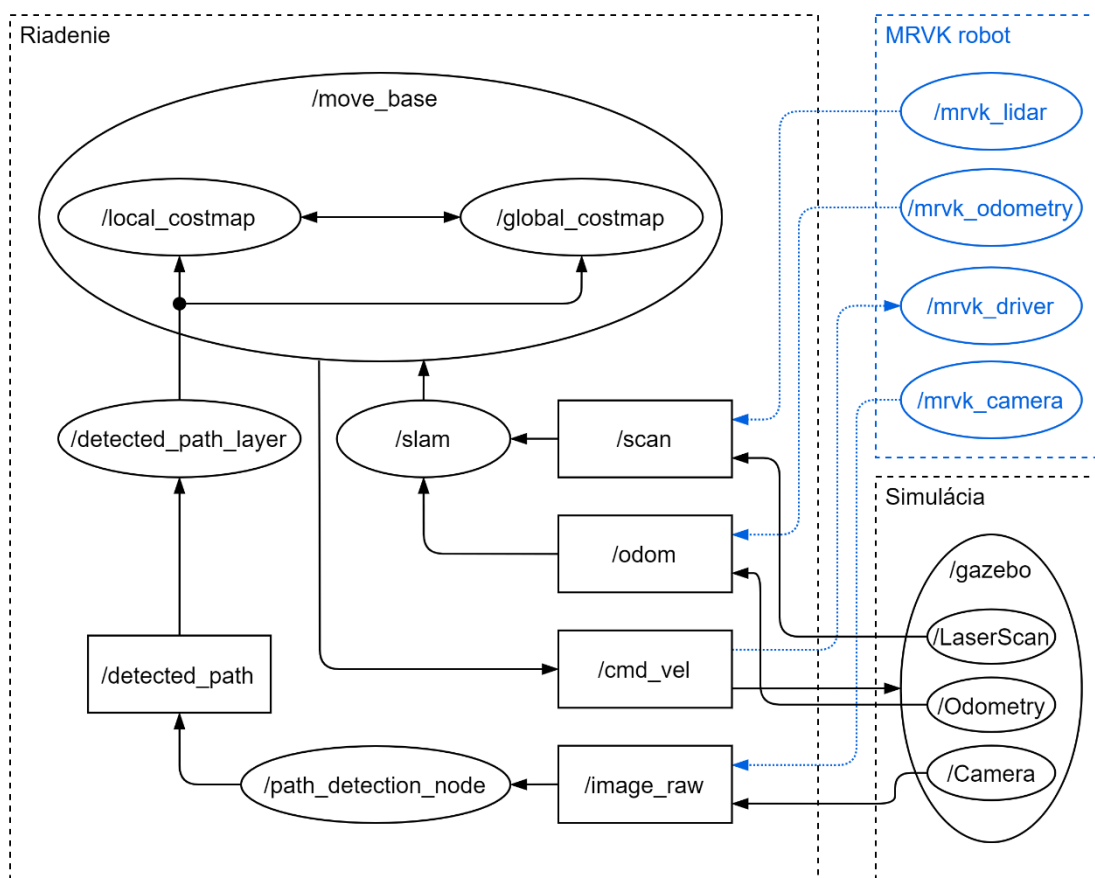


Obr. 22: Schematické znázornenie prepojenia štruktúry balíčka `mrvk_driver`

4 Balík tímového projektu

4.1 Architektúra projektu

Ako už bolo spomínané Gazebo je simulačné prostredie v ktorom sa môžu odladiť základné nedostatky softvéru a následne sa môže bez väčších ťažkostí implementovať na reálny hardvér a otestovať.



Obr. 23: Schematické znázornenie architektúry projektu

Veľkou výhodou ROS je, že riadenie je pre simulované prostredie a aj pre reálneho robota rovnaké. Mení sa len prepojenie niektorých vydavateľov a odoberateľov. Na Obr. 23 sú modrou farbou znázornené prepojenia ktoré je potrebné vytvoriť aby riadenie bolo implementované do reálneho robota. Pre testovanie riadenia simulačne sa ponechajú prepojenia, ktoré idú z/do Gazebo a prepojenia s MRVK robotom sa prerušia.

4.2 Inštalácia

4.2.1 ROS Melodic

Existuje viacero možností inštalácie ROS (verzia Melodic), oficiálnou možnosťou je inštalačný návod na materskej webovej stránke. Oveľa jednoduchšou alternatívou, oproti oficiálnej, je skript, ktorý sa nachádza na github repozitári *wsnewman/learning_ros_setup_scripts*. Pre jeho stiahnutie je potrebné v termináli použiť príkaz:

```
$ git clone  
https://github.com/wsnewman/learning\_ros\_setup\_scripts.git
```

Po stiahnutí sa musia nastaviť súbory ako spustiteľné skripty:

```
$ cd learning_ros_setup  
$ chmod +x *.sh
```

Pre spustenie inštalácia potom už len stačí zadať:

```
$ ./install_ros_and_tools_melodic.sh  
$ ./setup_workspace_learning_ros_melodic.sh GITHUB_USERNAME  
GITHUB_MAIL
```

Za premenné GITHUB_USERNAME GITHUB_MAIL je nutné zadať svoj nick a e-mail od github účtu. Prvý skript inštaluje všetky potrebné nástroje. Druhý inicializuje premenné a pripravuje pracovný priečinok ROS-u (workspace).

4.2.2 Balík tímového projektu

Nasledujúcimi príkazmi sa stiahne aktuálny repozitár tímového projektu a následne sa skompiluje.

```
$ git clone https://github.com/SamuelKacej/TP\_MRVK.git
$ roscd
$ catkin_make
```

Balík je potom možné spustiť z terminálu spustením súboru *robot.launch* (treba sa presvedčiť, že súbor má priradené práva na spustenie):

```
$ roslaunch mrvk_gazebo robot.launch
```

Uzol pre detekciu cesty, na základe vizuálneho systému, sa spúšťa zo súboru *path_detection_node.py* (tiež sa treba presvedčiť o spustiteľnosti):

```
$ rosrun path_detection path_detection_node.py
```

Záver

Cieľom tímového projektu bola príprava simulovaného zadania pre možnosť rozšírenia predmetu Riadenie mobilných robotov v prípade online výučby. Rozšírenie predmetu spočíva v integrácii mobilného robota do vonkajšieho prostredia a simulácii správania sa globálneho a lokálneho plánovania pre diferenciálny podvozok.

Dokumentácia tímového projektu je rozdelená do štyroch hlavných sekcií. Prvá sekcia je venovaná základným informáciám týkajúcich sa Robotického operačného systému, ktorý je základom existujúcich riadiacich programov robota MRVK a rovnako bol použitý aj pri vypracovaní tímového projektu.

Druhá sekcia je zameraná na integráciu simulačného modelu robota do prostredia Gazebo. Najskôr je v krátkosti predstavené samotné simulačné prostredie Gazebo a vytvorený model robota MRVK. Do modelu robota boli dodatočne integrované tri senzory, konkrétne laserový diaľkomer, inerciálna meracia jednotka a kamera. Na zabezpečenie bezkolíznej navigácie bol využitý balík `move_base`, ktorý využíva súčinnosť globálneho a lokálneho plánovania pre dosiahnutie stanovenej pozície. Balík `move_base` poskytuje možnosť výberu reaktívnej navigácie, pričom vybraná bola metóda dynamického okna. Mapa prostredia a poloha robota je získaná súčasnou lokalizáciou a mapovaním prostredníctvom balíka `slam_gmapping`. Do riadiaceho systému bol navyše pridaný aj systém zabezpečujúci detekciu chodníka vizuálnym systémom robota. Systém detekcie chodníka je schopný pracovať nielen samostatne, ale taktiež aj v spojení s navigačným systémom `move_base`. Uvedené bolo dosiahnuté vytvorením novej vrstvy v lokálnej costmape navigačného systému `move_base`, do ktorej sú transformované spracované dáta z kamery.

Tretia sekcia opisuje hardvérové a softvérové vybavenie reálneho robota MRVK. Nakoľko sa nám nepodarilo dostať ku žiadnej dokumentácii robota MRVK, jeho fungovanie a komunikáciu medzi jednotlivými modulmi sme analyzovali prostredníctvom reverzného inžinierstva.

V štvrtej sekcií je popísaný postup potrebný pre inštaláciu vytvoreného balíku. Nakoľko sme využili Robotický operačný systém, konkrétne jeho verziu Melodic, je v tejto sekcií uvedený aj postup k jeho nainštalovaniu.

Reálny robot MRVK je len jeden, čo významne obmedzilo jeho použitie na cvičeniach z predmetu Riadenie mobilných robotov. Dôvodom je, že počas normálnej formy vyučovania by nebolo umožnené na ňom pracovať všetkým skupinám súčasne. Dištančná výučba v čase koronakrízy však priniesla nielen negatíva, ale aj množstvo nových riešení a nápadov. Jedným z nich bolo aj použitie simulovaného zadania, pretože z dôvodov opatrení nebolo možné, aby študenti na cvičeniach s reálnymi robotmi pracovali. Simulácia samozrejme nedokáže úplne nahradiť prácu s reálnym zariadením, pri ktorej potrebné riešiť viacero ďalších problémov. Alternatívne riešenie využitím simulácie však prináša aj niekoľko výhod. Študenti mohli riadiace systémy navrhovať a testovať bez toho, aby bolo nutné prísť do laboratória, a teda neboli ani časovo obmedzení.

Okrem samotného navrhnutého riadenia robota MRVK je prínosom tohto projektu práve riešenie uvedeného problému. Aj napriek tomu, že reálny robot MRVK je len jeden, môžu študenti navrhovať a ladiť riadiace algoritmy využitím simulácii. Následne po základnom odladení môžu robiť pokusy na reálnom zariadení bez toho, aby výrazným spôsobom obmedzovali ostatných. Vytvorený balík je teda možné použiť nielen počas online výučby, ale taktiež aj počas normálnej formy vyučovania. Do pozornosti treba dať modulárnosť súčasného riešenia, a teda jednoduchosť pridania ďalších snímačov, prípadne náhrady existujúcich častí riadenia. Vzhľadom na uvedené môžeme považovať cieľ tímového projektu za splnený.

Literatúra

- [1] DUCHOŇ, F. 2021. *Riadenie mobilných robotov*. Skriptá z predmetu Riadenie mobilných robotov. [cit. 2021-05-22].
- [2] [s. n.]. Hokuyo URG-04LX-UG01 Scanning Laser Range Finder. 2009. *Specification*. [online]
URL: <<https://www.robotshop.com/media/files/pdf/hokuyo-urg-04lx-ug01-specifications.pdf>>. [cit. 2021-05-22].
- [3] [s. n.]. Gazebo. 2014. [online] URL: <<http://gazebo.org>>. [cit. 2021-05-22].
- [4] [s. n.]. Gazebo. 2014. *URDF in Gazebo*. [online]
URL: <http://gazebo.org/tutorials/?tut=ros_urdf>. [cit. 2021-05-22].
- [5] [s. n.]. ROS Wiki. 2014. *ROS/Wiki/Patterns/Communication*. [online]
URL: <<https://wiki.ros.org/ROS/Patterns/Communication>>. [cit. 2021-05-22].
- [6] [s. n.]. ROS Wiki. 2020. *move_base*. [online]
URL: <https://wiki.ros.org/move_base>. [cit. 2021-05-22].
- [7] [s. n.]. ROS Wiki. 2019. *base_local_planner*. [online]
URL: <https://wiki.ros.org/base_local_planner>. [cit. 2021-05-22].
- [8] [s. n.]. ROS Wiki. 2018. *costmap_2d*. [online]
URL: <https://wiki.ros.org/costmap_2d>. [cit. 2021-05-22].
- [9] [s. n.]. ROS Wiki. 2019. *gmapping*. [online]
URL: <<https://wiki.ros.org/gmapping>>. [cit. 2021-05-22].
- [10] [s. n.]. OpenCV. 2021. *OpenCV modules*. [online]
URL: <<https://docs.opencv.org/master/index.html>>. [cit. 2021-05-22].
- [11] [s. n.]. Gazebo worlds. 2020. [online] URL: <https://github.com/PX4/PX4-SITL_gazebo/blob/master/worlds/baylands.world> [cit. 2021-05-22].