



FAKULTA APLIKOVANÝCH VĚD  
ZÁPADOČESKÉ UNIVERZITY  
V PLZNI



ZÁPADOČESKÁ  
UNIVERZITA  
V PLZNI

Západočeská univerzita v Plzni  
Fakulta aplikovaných věd

**Vývoj autonomní mobilní platformy pro přepravu  
nákladů**

---

**Diplomová práce**

Jakub Tvrz

Ing. Milan Mráz  
Leuze Engineering Czech s.r.o.  
KKY/Ing. Miroslav Flídr, Ph.D.

Datum zpracování: 16. května 2025



ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2024/2025

# ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Bc. Jakub TVRZ**

Osobní číslo: **A23N0046P**

Studijní program: **N0714A150011 Kybernetika a řídící technika**

Specializace: **Automatické řízení a robotika**

Téma práce: **Vývoj autonomní mobilní platformy pro přepravu nákladů**

Zadávající katedra: **Katedra kybernetiky**

## Zásady pro vypracování

- Seznamte se s SW rámcem ROS2 a jeho moduly pro autonomní navigaci.
- Navrhněte a otestujte vhodnou skladbu sensoru pro detekci vzájemné polohy platformy a převáženého nákladu.
- Navrhněte řídicí systém autonomní platformy využívající ROS2, který zohledňuje uložení nákladu na platformě.
- Otestujte navrhnuté řešení v reálném prostředí.

Rozsah diplomové práce:

**40-50 stránek A4**

Rozsah grafických prací:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

Klancar, G., Zdesar, A., Blazic, S., & Skrjanc, I. (2017). *Wheeled Mobile Robotics: From Fundamentals Towards Autonomous Systems*. Butterworth-Heinemann.

Vedoucí diplomové práce:

**Ing. Miroslav Flídr, Ph.D.**

Katedra kybernetiky

Datum zadání diplomové práce:

**1. října 2024**

Termín odevzdání diplomové práce: **19. května 2025**



**Doc. Ing. Miloš Železný, Ph.D.**  
děkan



**Doc. Dr. Ing. Vlasta Radová**  
vedoucí katedry

V Plzni dne 1. září 2024

## **Prohlášení**

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 16. května 2025:

---

## **Poděkování**

Rád bych poděkoval svému vedoucímu diplomové práce Ing. Miroslavu Flídovi, Ph.D. za cenné rady, vstřícnost a čas, který mi během zpracování dokumentace věnoval. Rovněž bych chtěl poděkovat firmě Leuze Engineering Czech s.r.o za možnost zpracovat toto téma; především Ing. Milanu Mrázovi, za kvalitní technické podklady a zdroje, a Ing Václavu Aubrechtovi, který celý projekt vedl a směroval. Poděkování patří i mému kolegovi Bc. Janu Holubovi, zejména za vytvoření ovládací aplikace, interaktivního API a PLC programů.

# **Abstrakt**

Tato práce se zabývá návrhem a implementací systému autonomní navigace pro robotické platformy s důrazem na integraci teoretických metod, softwarových nástrojů a hardwarových řešení v průmyslovém prostředí. Cílem je vyvinout efektivní a ekonomicky dostupná navigační řešení, která překonávají omezení stávajících přístupů, jako je tradiční sledování čáry. Práce zahrnuje podrobné odvození matematických principů algoritmů, jako jsou například prediktivní řízení, Pure Pursuit, spojování LIDARových skenů či shlukování dat, které řeší dílčí úlohy regulace trajektorie a detekce objektů. Dále je analyzována architektura frameworku ROS2 a jeho navigační knihovny Nav2, konkrétně algoritmy lokalizace a generování trajektorií, jež tvoří základ softwarové implementace. Hardwarová část popisuje senzory Leuze a strukturu řešení čtyř prototypů s různými podvozky, včetně jejich bezpečnostního zajištění pomocí LIDARů. Navržené metody byly ověřeny simulací v prostředí Gazebo, kde byly testovány scénáře autonomní navigace, pohybu po virtuálních drahách a pohyb v realistickém modelu skládové haly. Praktická aplikace teoretických poznatků je demonstrována funkčním modulárním systémem schopným řešit navigační úlohy. Výsledky nasazení na čtyřech různých platformách ukazují, že navržený systém nabízí robustní a flexibilní řešení, které lze dále rozšiřovat. Práce tak slouží jako metodický návod, přehled nezbytných komponent a zdroj inspirace pro návrh vlastního průmyslového robota postaveného na frameworku ROS2, čímž přispívá k rozvoji autonomní robotiky jak uceleným a shrnutým teoretickým základem, tak praktickými podněty pro další aplikace.

---

# **Klíčová slova**

autonomní navigace, robotika, ROS2, Nav2, virtuální dráhy, sledování čáry, LIDAR, bezpečnostní zóny, prediktivní řízení, Pure Pursuit, shlukování dat, kinematické modely, lokalizace, generování trajektorií, sledování trajektorií, Gazebo, simulace, průmyslové roboty, senzory Leuze, AGV, AMR navigační algoritmy, zpracování LIDARových dat, detekce palety, SLAM

# **Abstract**

This thesis deals with the design and implementation of an autonomous navigation system for robotic platforms with an emphasis on the integration of theoretical methods, software tools and hardware solutions in an industrial environment. The goal is to develop efficient and cost-effective navigation solutions that overcome the limitations of existing approaches such as traditional line tracking. The work includes a detailed derivation of the mathematical principles of algorithms such as predictive steering, Pure Pursuit, LIDAR scan fusion and data clustering that solve the sub-problems of trajectory control and object detection. Furthermore, the architecture of the ROS2 framework and its navigation library Nav2 is analyzed, specifically the localization and trajectory generation algorithms that form the basis of the software implementation. The hardware part describes the Leuze sensors and the solution structure of four prototypes with different chassis, including their safety provisioning using LIDARs. The proposed methods were validated by simulations in the Gazebo environment, where scenarios of autonomous navigation, movement on virtual tracks and movement in a realistic warehouse model were tested. The practical application of the theoretical knowledge is demonstrated by a functional modular system capable of solving navigation tasks. Deployment results on four different platforms show that the proposed system offers a robust and flexible solution that can be further extended. Thus, the work serves as a methodological guide, an overview of the necessary components and a source of inspiration for the design of a custom industrial robot based on the ROS2 framework, contributing to the development of autonomous robotics both with a comprehensive and summarized theoretical foundation and with practical suggestions for further applications.

---

# **Keywords**

autonomous navigation, robotics, ROS2, Nav2, virtual trajectories, line tracking, LIDAR, safety zones, predictive control, Pure Pursuit, data clustering, kinematic models, localization, trajectory generation, trajectory tracking, Gazebo, simulation, industrial robots, Leuze sensors, AGV, AMR navigation algorithms, LIDAR data processing, pallet detection, SLAM

# Slovníček pojmu

Pojem	Definice
AGV	Automaticky řízené vozidlo (Automated Guided Vehicle), mobilní robot používaný v průmyslu pro automatizovanou dopravu materiálu, obvykle navigující podle předem definovaných tras nebo senzorů.
Autonomní navigace	Schopnost robota pohybovat se v prostředí bez přímého lidského zásahu na základě senzorových dat a algoritmů plánování.
Gazebo	Open-source simulační prostředí pro testování robotických systémů v realistických virtuálních scénářích.
Kinematický model	Matematický popis pohybu robota založený na jeho geometrii a omezeních bez zohlednění silových vlivů.
LIDAR	Optický senzor využívající laserové paprsky k měření vzdáleností a mapování prostředí, často používaný pro navigaci a detekci překážek.
Nav2	Navigační knihovna v ROS 2 poskytující algoritmy pro plánování tras, lokalizaci a řízení autonomních robotů.
SLAM	Simultánní lokalizace a mapování (Simultaneous Localization and Mapping), technika umožňující robotovi vytvářet mapu neznámého prostředí a současně určovat svou polohu v ní na základě senzorových dat.
Odometrie	Odhad polohy a orientace robota na základě měření pohybu jeho kol nebo jiných mechanických částí.
PLC	Programovatelný logický automat pro řízení průmyslových procesů, včetně bezpečnostních funkcí při integraci se senzory.
PointCloud	Datová struktura reprezentující soubor bodů v prostoru získaný např. z LIDARu, používaný pro mapování a analýzu prostředí.
ProfiSafe	Bezpečnostní komunikační protokol pro přenos dat mezi senzory a řídicími jednotkami v průmyslových aplikacích.
Pure Pursuit	Algoritmus sledování trajektorie, který určuje řídicí příkazy na základě geometrického pronásledování cílového bodu.

<b>Pojem</b>	<b>Definice</b>
ROS 2	Robotický operační systém druhé generace, framework pro vývoj distribuovaných robotických aplikací s důrazem na komunikaci a modularitu.
RViz	Vizualizační nástroj v ROS pro zobrazení senzorových dat, map a trajektorií robota v reálném čase.
Navigační stack	Soubor softwarových nástrojů a algoritmů, který zajišťuje autonomní navigaci mobilního robota. Zahrnuje globální a lokální plánování trajektorií, zpracování senzorových dat (např. LiDAR, odometrie), lokalizaci v mapě, vyhýbání se překážkám a generování řídicích příkazů pro pohyb robota k cíli.

# Obsah

<b>1 Specifikace práce</b>	<b>1</b>
Úvod . . . . .	2
Poznámka autora . . . . .	3
1.1 Rešerše . . . . .	4
1.1.1 Požadavky kladené na funkce AGV . . . . .	4
1.1.2 Existující řešení . . . . .	6
1.1.2.1 SSI SCHÄFER WEASEL . . . . .	6
1.1.2.2 Linde Material Handling C-Matic . . . . .	7
1.1.2.3 KUKA KMP 600-S diffDrive . . . . .	8
1.1.2.4 ServisControl . . . . .	11
1.1.2.5 ABB AMR Solutions . . . . .	12
1.1.2.6 Toyota Autopilot . . . . .	13
1.1.2.7 myFABER . . . . .	14
1.1.3 Existující SW řešení pro autonomní navigaci . . . . .	16
1.1.3.1 SIMOVE . . . . .	16
1.1.3.2 ROS2 . . . . .	17
1.1.4 Konkrétní řešení vybraná na základě průzkumu . . . . .	18
1.1.4.1 Navigace . . . . .	18
1.1.4.2 Detekce palety . . . . .	18
1.1.4.3 Simulace . . . . .	19
1.1.4.4 Vývojové prostředí . . . . .	19
1.1.4.5 Použití jako testovací platforma . . . . .	19
<b>2 Hardware</b>	<b>21</b>
2.1 Použité senzory . . . . .	22
2.1.1 RSL400 . . . . .	22
2.1.2 RSL 200 . . . . .	25
2.1.2.1 Stavba senzoru a optický princip . . . . .	25
2.1.2.2 Nastavení zón . . . . .	27
2.1.2.3 Využití v praxi . . . . .	28
2.1.2.4 Provozní podmínky . . . . .	28
2.1.3 OGS . . . . .	30
2.1.4 DCR . . . . .	33
2.2 Použité podvozky . . . . .	35
2.2.0.1 Poznámka k všesměrovým kolům - mecanum wheels . . . . .	35
2.2.1 DJI Robomaster EP . . . . .	37
2.2.2 Diferenciální podvozek . . . . .	40
2.2.3 Velký všesměrový podvozek . . . . .	42

2.2.4	TinyAGV diferenciální podvozek	44
<b>3</b>	<b>ROS2 - Robot Operating system</b>	<b>46</b>
3.1	ROS2 vs ROS1	47
3.2	Architektura ROS2	48
3.3	ROS2 graf a komunikační mechanismy	50
3.3.1	Uzly a jejich role	50
3.3.2	Komunikační mechanismy	51
3.3.3	Quality of Service (QoS)	52
3.4	Transformace v ROS2	54
3.4.1	Transformační strom	54
3.4.2	Statické a dynamické transformace	54
3.4.2.1	Statické transformace	55
3.4.2.2	Dynamické transformace	55
3.5	Lokalizace – odometrie a AMCL	57
3.5.1	Částicový filtr	58
3.5.2	Adaptivní částicový filtr	59
3.5.3	Použití adaptivního částicového filtru pro lokalizaci	60
3.5.4	Koncepce balíčku AMCL	61
3.6	Plánování trajektorie	63
3.6.1	Dijkstrův algoritmus jako teoretický základ	63
3.6.1.1	Příklad	64
3.7	Prohledávání grafů a hledání nejkratší cesty	66
3.7.1	Základní principy prohledávání grafů	66
3.7.2	Hledání nejkratší cesty pomocí A* algoritmu	66
3.7.2.1	Základní princip A*	66
3.7.2.2	Postup algoritmu A*	67
3.7.2.3	Příklad heuristiky	67
3.7.3	Výhody a omezení A*	68
3.7.4	NavFnPlanner	69
3.7.4.1	Generování nákladové mapy	69
3.7.4.2	Výpočet optimální cesty	69
3.7.4.3	Extrakce výsledné trajektorie	70
3.7.5	Lokální plánování s MPPI	70
3.8	Simulační prostředí Gazebo	71
3.8.1	Fyzikální engine	72
3.8.2	Návrh simulace	73
3.8.2.1	Model prostředí	73
3.8.2.2	Model robota	73
3.8.2.3	Použití	76
3.9	RViz	77
3.9.1	Princip fungování	78
3.9.2	Základní funkce a aplikace	78
3.9.2.1	Vizualizace senzorových dat	78
3.9.2.2	Vizualizace stavu robota	79
3.9.2.3	Interaktivní ovládání a navigace	79
3.9.2.4	Vizualizace map a modelů	79
3.9.3	Praktické aplikace	80

<b>4 Teoretické podklady</b>	<b>81</b>
4.1 Kinematický model . . . . .	82
4.1.1 Diferenciální podvozek . . . . .	82
4.1.1.1 Dopředná kinematika . . . . .	82
4.1.1.2 Inverzní kinematika . . . . .	84
4.1.1.3 Pohyb v globálním souřadném systému . . . . .	85
4.1.2 Podvozek s mecanum koly . . . . .	86
4.1.2.1 Inverzní kinematika . . . . .	86
4.1.2.2 Dopředná kinematika . . . . .	91
4.1.2.3 Pohyb v globálním souřadném systému . . . . .	92
4.1.2.4 Simulační aplikace . . . . .	93
4.2 Spojení dvou LIDARových skenů do virtuálního LIDARu . . . . .	94
4.2.1 Problém a konfigurace LIDARů . . . . .	94
4.2.2 Transformace dat do virtuálního LIDARu . . . . .	95
4.2.3 Zpracování překrývajících se dat . . . . .	96
4.2.4 Vizualizace a výsledky . . . . .	96
4.3 Prediktivní řízení . . . . .	97
4.3.1 Matematické základy MPC . . . . .	97
4.3.1.1 Formulace stavového modelu . . . . .	98
4.3.1.2 Predikce stavů a výstupů . . . . .	98
4.3.1.3 Metoda move blocking . . . . .	99
4.3.2 Formulace a řešení optimalizačního problému . . . . .	100
4.3.2.1 Regulace do nuly . . . . .	100
4.3.2.2 Obecné řízení a sledování trajektorie . . . . .	101
4.3.3 Odhad stavů pomocí klouzavého horizontu (MHE) . . . . .	103
4.3.3.1 Formulace modelu a měření . . . . .	103
4.3.3.2 Optimalizační problém MHE . . . . .	104
4.3.4 Lokální plánovač pomocí Model Predictive Path Integral Control . . . . .	105
4.3.4.1 Formulace modelu a vstupů . . . . .	105
4.3.4.2 Optimalizační problém MPPI . . . . .	106
4.3.4.3 Použití pro sledování trajektorie . . . . .	107
4.3.4.4 Simulace a vizualizace . . . . .	107
4.4 Pure Pursuit algoritmus . . . . .	109
4.4.1 Geometrický princip algoritmu . . . . .	109
4.4.2 Odvození poloměru otáčení . . . . .	109
4.4.3 Výpočet řídicích vstupů . . . . .	110
4.4.4 Implementace algoritmu . . . . .	111
4.4.5 Příklad aplikace . . . . .	111
4.5 Detekce malých objektů z 2D LIDARových skenů . . . . .	113
4.5.1 Detekce objektů ze ztrátové mapy pomocí OpenCV . . . . .	113
4.5.2 Shlukování ve strojovém učení a rozpoznávání objektů . . . . .	115
4.5.2.1 Algoritmus K-means . . . . .	115
4.5.2.2 Algoritmus DBSCAN . . . . .	115
4.5.2.3 Vyhodnocení a rozdíly mezi K-means a DBSCAN . . . . .	117

<b>5 Aplikace</b>	<b>118</b>
5.1 Bezpečnostní zóny autonomních vozíků . . . . .	119
5.1.1 Hardwareové řešení s PLC . . . . .	120
5.1.2 Softwarové řešení v ROS2 . . . . .	121
5.1.3 Kombinované řešení s OSSD a softwarovým monitoringem . . . . .	121
5.1.4 Shrnutí . . . . .	122
5.2 Autonomní navigace . . . . .	123
5.2.1 Základní koncepty a mechanismy navigace v Nav2 . . . . .	123
5.2.1.1 Akční servery . . . . .	123
5.2.1.2 Uzly s řízeným životním cyklem . . . . .	124
5.2.1.3 Stromy chování . . . . .	125
5.2.1.4 Navigační servery . . . . .	126
5.2.1.5 Sledování bodů tras . . . . .	128
5.2.1.6 Odhad stavu a transformace v navigačním stacku . . . . .	128
5.2.2 Struktura propojení subsystémů navigace . . . . .	131
5.2.2.1 Vytvoření mapy prostředí pomocí SLAM Toolbox . . . . .	131
5.2.2.2 Lokalizace v mapě . . . . .	133
5.2.2.3 Navigace a plánování trajektorie . . . . .	135
5.2.2.4 Bezpečnostní mechanismy . . . . .	139
5.2.2.5 Integrovaný proces navigace . . . . .	140
5.2.3 Aplikace na různých platformách . . . . .	142
5.2.3.1 Podvozek DJI . . . . .	142
5.2.3.2 Diferenciální podvozek . . . . .	143
5.2.3.3 AGV s velkým všeobecným podvozkem . . . . .	144
5.3 Jízda po čáře . . . . .	146
5.3.1 Motivace a výchozí požadavky . . . . .	146
5.3.2 Technická implementace a konfigurace . . . . .	147
5.3.3 Aplikace na různých platformách . . . . .	148
5.3.3.1 Podvozek DJI . . . . .	148
5.3.3.2 Diferenciální podvozek . . . . .	149
5.3.3.3 Shrnutí . . . . .	150
5.4 Automatická navigace . . . . .	151
5.4.1 Jízda po virtuální čáře . . . . .	151
5.4.1.1 Jednoduché grafické uživatelské rozhraní (GUI) . . . . .	152
5.4.1.2 Shrnutí . . . . .	153
5.4.2 Fleet management . . . . .	154
5.4.2.1 Definice a podstata správy flotily . . . . .	154
5.4.2.2 Výhody správy flotily nad autonomní navigací . . . . .	154
5.4.3 Virtuální dráhy . . . . .	156
5.4.3.1 Základní princip a využití . . . . .	156
5.4.3.2 Shrnutí . . . . .	158
5.4.4 Aplikace na různých platformách . . . . .	159
5.4.4.1 Podvozek DJI . . . . .	159
5.4.4.2 TinyAGV . . . . .	160
5.5 Automatická detekce palety . . . . .	162
5.5.1 Shlukování pro detekci objektů z dat LIDARu . . . . .	162
5.5.2 Využití pro detekci palety . . . . .	163
5.5.3 Potenciální rozšíření: Odhad pohybu objektů pomocí Kalmanova filtru . . . . .	165

5.5.4	Aplikace na různých platformách . . . . .	167
5.5.4.1	DJI podvozek . . . . .	167
5.5.4.2	AGV s velkým všesměrovým podvozkem . . . . .	170
5.6	Simulace . . . . .	174
5.6.1	Model robota a simulace senzorů . . . . .	174
5.6.2	Simulační prostředí . . . . .	175
5.6.3	Testované scénáře . . . . .	175
5.6.3.1	Autonomní navigace . . . . .	176
5.6.3.2	Jízda po virtuálních čarách . . . . .	177
5.6.3.3	Kombinace autonomní jízdy a virtuálních čar . . . . .	177
5.6.3.4	Virtuální dráhy . . . . .	178
5.6.4	Shrnutí . . . . .	179
<b>6</b>	<b>Závěr</b>	<b>180</b>
<b>Appendices</b>		<b>183</b>
<b>A</b>	<b>Porovnání s komerčním řešením SIMOVE</b>	<b>184</b>
Reference		192

# Seznam obrázků

1.1	Ilustrační obrázek SSI SCHÄFER WEASEL [1] . . . . .	6
1.2	Ilustrační obrázek Linde C-Matic [2] . . . . .	7
1.3	Ilustrační obrázek KMP 600-S diffDrive [3] . . . . .	9
1.4	Ilustrační obrázek AMR U1500XN [4] . . . . .	10
1.5	Ilustrační obrázek RoboMec [5] . . . . .	11
1.6	Ilustrační obrázek Flexley Tug [6] . . . . .	12
1.7	Ilustrační obrázek CDI120 [7] . . . . .	13
1.8	Ilustrační obrázek myFABER Q3-600 [8] . . . . .	14
1.9	Ilustrační obrázek SIMOVE [9] . . . . .	16
2.1	Schéma sestavy senzoru RSL 400 s měřicí hlavou a komunikační jednotkou [10].	22
2.2	Konfigurace bezpečnostních zón senzoru RSL 400 s možností monitorování až čtyř polí [10].	23
2.3	Znázornění bezpečnostních zón senzoru RSL 400 v AGV aplikaci [11].	24
2.4	Schéma sestavy senzoru RSL 200 s měřicí a vyhodnocovací jednotkou.	25
2.5	Princip rozmítání paprsku senzoru RSL 200 s rotujícím polygonem.	26
2.6	Skenovací rovina senzoru RSL 200.	26
2.7	Bezpečnostní zóny senzoru RSL 200 s vyznačením ZLD (modrá) a bezpečnostního rozsahu (červená).	27
2.8	Triplet zón u RSL 200.	28
2.9	Senzor OGS 600 pro detekci optických vodicích stop [12].	30
2.10	Fungování filtru šířky čáry pro potlačení rušivých prvků [12].	31
2.11	Ilustrační fotografie senzoru DCR 248i [13].	33
2.12	Detail mecanum kola	35
2.13	Možnosti pohybu vozíku v závislosti na rychlosti jednotlivých kol	36
2.14	DJI RoboMaster EP Core [14]	37
2.15	Schéma komunikace a propojení funkčních celků prototypu postaveného na DJI podvozku	38
2.16	DJI přestavěný pro účely autonomní navigace	39
2.17	DJI přestavěný pro účely sledování čáry	39
2.18	Schéma komunikace a propojení funkčních celků prototypu postaveného na vlastní diferenciálním podvozku	40
2.19	Diferenciální podvozek přestavěný pro účely autonomní navigace	41
2.20	Diferenciální podvozek přestavěný pro účely sledování čáry	41
2.21	Schéma komunikace a propojení funkčních celků prototypu postaveného na mecanum podvozku	42
2.22	Fotografie AGV z veletrhu v Norimberku	43
2.23	Schéma komunikace a propojení funkčních celků prototypu postaveného na malém diferenciálním podvozku	44

2.24 Fotografie AGV z veletrhu v Lionu . . . . .	45
2.25 Fotografie TinyAGV AGV . . . . .	45
3.1 Ilustrační obrázek ROS2 [15]. . . . .	47
3.2 Schéma komunikace v DDS vrstvě ROS2. . . . .	48
3.3 Layerová architektura systému ROS2 s RMW a DDS. . . . .	49
3.4 Princip komunikace mezi komponentami ROS2 grafu. . . . .	50
3.5 Vizualizace ROS2 grafu pomocí nástroje <i>rqt</i> . . . . .	51
3.6 Mechanismus publikování transformací v <i>tf2</i> . . . . .	54
3.7 Paralela mezi transformačním stromem a URDF strukturou. . . . .	55
3.8 Zpracování URDF a publikování dynamických transformací. . . . .	56
3.9 Znázornění transformačního stromu pomocí nástroje <i>rqt</i> . . . . .	56
3.10 Princip integrace kinematických modelů a odometrie pro navigaci a lokalizaci .	57
3.11 Princip lokalizace pomocí AMCL v ROS2: Odometrie publikuje transformaci <code>base_frame → odom_frame</code> , AMCL koriguje drift a publikuje <code>odom_frame</code> <code>→ map_frame</code> . . . . .	61
3.12 Příklad Dijkstrova algoritmu . . . . .	65
3.13 Gazebo simulace AGV vybaveného LIDARy v definovaném prostředí. . . . .	71
3.14 Gazebo a RViz – testování autonomní navigace AGV v simulačním prostředí. .	76
3.15 Ukázka nástroje RViz v kombinaci se simulací v Gazebo: vizualizace mapy a trajektorie robota. . . . .	77
4.1 Geometrie pohybu diferenciálního podvozku.. . . . .	82
4.2 Přenášení sil na mecanum kolu 2 (pravé přední) . . . . .	86
4.3 Přenášení sil na mecanum kolu 1 (levé přední) . . . . .	87
4.4 Zohlednění vlivu otáčení vozíku kolem osy <i>z</i> . . . . .	88
4.5 Otáčení v kladném směru úhlové rychlosti $\omega$ . . . . .	89
4.6 Vztah globálního a lokálního souřadného systému . . . . .	90
4.7 Možnosti pohybu vozíku v závislosti na rychlosti jednotlivých kol . . . . .	91
4.8 Ukázka aplikace pro test funkčnosti kinematického modelu . . . . .	93
4.9 Ukázka aplikace pro test funkčnosti kinematického modelu 2 . . . . .	93
4.10 Rozmístění dvou LIDARů na obdélníkovém vozíku. . . . .	94
4.11 Sjednocení dvou LIDARových skenů do virtuálního LIDARu prostřednictvím transformace souřadnic. . . . .	95
4.12 Vizualizace spojení LIDARových skenů v RViz: červená a modrá data předsta- vují původní skeny, bílá data výsledný virtuální <i>LaserScan</i> . . . . .	96
4.13 Zobrazení principu horizontů . . . . .	99
4.14 Znázornění simulace MPPI . . . . .	108
4.15 Znázornění příkladu funkce Pure Pursuit algoritmu . . . . .	112
4.16 Výsledek detekce z OpenCV - RViz vizualizace . . . . .	114
4.17 Schéma principu algoritmu DBSCAN s rozlišením středových, hraničních a od- lehlych bodů. . . . .	116
5.1 Schéma vícevrstvých bezpečnostních zón kolem autonomního vozíku. . . . .	119
5.2 Konfigurace zón RSL400 a detekce narušení v Sensor Studiu [16]. . . . .	120
5.3 Zóny kolem robota v RVizu vytvořené Collision Monitorem. . . . .	121
5.4 Ukázká průběhu mapování pomocí SLAMToolbox. . . . .	131
5.5 Schéma propojení jednotlivých komponent SLAM Toolboxu. . . . .	132
5.6 Ukázka vizualizace lokalizace v mapě pomocí nástroje RViz. . . . .	134

5.7	Schéma propojení jednotlivých komponent lokalizace. . . . .	134
5.8	Příklad přímé trajektorie generované algoritmem NavFn v úzké chodbě. . . . .	136
5.9	Rozšíření překážek o bezpečnostní zónu odpovídající poloměru robota, zajišťující bezkolizní navigaci. . . . .	137
5.10	Vyhlazení zatáček s ohledem na minimální poloměr otáčení a bezpečnostní zóny. . . . .	138
5.11	Schéma propojení jednotlivých komponent pro plánování a sledování trajektorie. . . . .	138
5.12	Příklad prediktivního řízení s MPPI v prostředí s překážkami, ukazující výběr optimální cesty k bodu na trajektorii. . . . .	139
5.13	Schéma propojení jednotlivých SW komponent bezpečnostních prvků. . . . .	140
5.14	Kompletní schéma propojení všech komponent autonomní navigace. . . . .	141
5.15	Ukázka fúze LIDARu a kamery v ROS2. . . . .	143
5.16	Ukázka funkce světlené signalizace. . . . .	144
5.17	Prostředí definující požadavky na AGV. . . . .	146
5.18	Schéma propojení komponent při navigaci po čáře. . . . .	147
5.19	Fotografie podvozku DJI sledujícího čáru a čtoucího QR kódy. . . . .	148
5.20	Fotografie finálního prototypu pro testy jízdy po čáře. . . . .	150
5.21	Ukázka kreslení drah do mapy pomocí <i>rmf-traffic-editor</i> . . . . .	151
5.22	Propojení jednotlivých komponent pro jízdu po virtuálních silnicích. . . . .	152
5.23	Ukázka jednoduchého ovládacího GUI. . . . .	153
5.24	Znázornění principu a funkce virtuálních drah. . . . .	156
5.25	Propojení jednotlivých komponent pro jízdu po virtuální dráze. . . . .	157
5.26	Ukázka virtuální čáry definované na mapě pro podvozek DJI. . . . .	159
5.27	Vizualizace mapy prostředí, lokalizace a následování trajektorie na displeji robota. . . . .	161
5.28	Schematické znázornění klíčových bodů palety, jejich geometrické konfigurace a vypočteného středu pro detekci pomocí DBSCAN a geometrické analýzy. . . . .	164
5.29	Schéma propojení komponent detekce palety se zbylým systémem. . . . .	164
5.30	Vizualizace detekce malých objektů a palet v RViz, s červenými body označujícími odhadované polohy a rozměry objektů. . . . .	165
5.31	Detekce palety z nákladové mapy. . . . .	167
5.32	Grafy detekce palety při průběžném pohybu robota. . . . .	168
5.33	Graf detekce palety při průběžném pohybu robota. . . . .	169
5.34	Fotografie podvozku DJI při testování zajetí pod paletu. . . . .	169
5.35	Ukázka držáku na paletu a jeho detekce. . . . .	170
5.36	Graf detekce palety při průběžném pohybu robota. . . . .	171
5.37	Ilustrační fotografie velkého AGV podjíždějícího paletu . . . . .	172
5.38	Testovací scénář . . . . .	173
5.39	Model robota . . . . .	174
5.40	Model simulačního prostředí . . . . .	175
5.41	Zachycení simulace v Gazebo a ovládání a vizualizace v RViz při aplikaci autonomní navigace. . . . .	176
5.42	Zachycení simulace v Gazebo a ovládání a vizualizace v RViz při aplikaci jízdy po virtuálních čarách. . . . .	177
5.43	Zachycení simulace v Gazebo a ovládání a vizualizace v RViz při kombinování autonomní navigace a jízdy po virtuálních čarách. . . . .	178
5.44	Zachycení simulace při aplikaci jízdy po virtuální dráze. . . . .	178
A.1	Schéma docker kontejnerů a jejich propojení. . . . .	185
A.2	Porovnání aplikací pro ROS2 a k dodanému řešení SIMOVE. . . . .	187

A.3 Porovnání podobných přístupů v ROS2 a SIMOVE. . . . .	188
A.4 Porovnání vizualizace v SIMOVE a ROS2. . . . .	188

# Seznam matematických vztahů

4.1	Lineární rychlosť kol z obrázku . . . . .	83
4.2	Vyjádření $\omega$ a $l$ . . . . .	83
4.3	Čisté vyjádření $\omega$ . . . . .	83
4.4	Vyjádření rychlosti $v$ . . . . .	83
4.5	Vyjádření rychlosti $v$ 2 . . . . .	84
4.6	Výsledné rovnice dopředné kinematiky . . . . .	84
4.7	Vyjádření úhlových rychlostí 1 . . . . .	84
4.8	Vyjádření úhlových rychlostí 2 . . . . .	84
4.9	Výsledné rovnice inverzní kinematiky . . . . .	84
4.10	Globální kinematické rovnice . . . . .	85
4.11	Kompletní globální kinematika . . . . .	85
4.12	Rychlosť působící na výsledný pohyb mecanum wheel . . . . .	86
4.13	Rychlosť působící na výsledný pohyb mecanum wheel . . . . .	86
4.14	Složky rychlosť působící na výsledný pohyb mecanum wheel . . . . .	87
4.15	Složení složek sil . . . . .	87
4.16	Druhý typ kola – odvození . . . . .	88
4.17	Výsledné rovnice inverzní kinematiky pro mecanum wheels . . . . .	89
4.18	Rovnice dopředné kinematiky . . . . .	91
4.19	Finální rovnice dopředné kinematiky . . . . .	92
4.20	Maticový tvar dopředné kinematiky . . . . .	92
4.21	Globální kinematické rovnice . . . . .	92
4.22	Kompletní globální kinematika . . . . .	92
4.23	Diskrétní stavový model . . . . .	98
4.24	Budoucí stavy a výstupy . . . . .	98
4.24	Kompletní globální kinematika . . . . .	98
4.25	Predikce stavů . . . . .	98
4.26	Predikce výstupů . . . . .	98
4.27	Metoda move blocking . . . . .	99
4.28	Upravené predikční matice . . . . .	99
4.29	Upravené vztahy pro predikci . . . . .	99
4.30	Obecný tvar ztrátové funkce . . . . .	100
4.31	Maticový tvar ztrátové funkce . . . . .	100
4.32	Kvadratický tvar ztrátové funkce . . . . .	100
4.33	Optimalizační problém s vazebními podmínkami . . . . .	100
4.34	Optimalizační problém s vazebními podmínkami . . . . .	101
4.35	Ztrátová funkce pro sledování trajektorie . . . . .	101
4.36	Kvadratický tvar pro sledování trajektorie . . . . .	101
4.37	Optimalizační problém pro sledování trajektorie s vazebními podmínkami . . . . .	102
4.38	Dynamický model systému . . . . .	103

4.39	Lineární dynamický model . . . . .	103
4.40	Nákladová funkce MHE . . . . .	104
4.41	Predikce stavů v horizontu . . . . .	104
4.42	Nelineární dynamický model . . . . .	105
4.43	Vzorkované vstupy . . . . .	105
4.44	Nákladová funkce MPPI . . . . .	106
4.45	Aproximace očekávané hodnoty . . . . .	106
4.46	Váhy trajektorií . . . . .	106
4.47	Aktualizace vstupů . . . . .	107
4.48	Nákladová funkce pro sledování trajektorie . . . . .	107
4.49	Vzdálenost mezi robotem a cílovým bodem . . . . .	109
4.50	Směrový vektor od robota k cílovému bodu . . . . .	109
4.51	Cosinus úhlu alpha . . . . .	109
4.52	Sinus úhlu alpha . . . . .	109
4.53	Zakřivení dráhy . . . . .	110
4.54	Dynamická změna lineární rychlosti . . . . .	110
4.55	Úhlová rychlosť omega . . . . .	110
4.56	Saturace úhlové rychlosti . . . . .	110
4.57	Přepočet Pure Pursuit přes kinematiku . . . . .	110
5.1	Experimentální identifikace v uzavřené smyčce . . . . .	149

# Seznam tabulek

2.1	Technické parametry senzoru RSL 400.	23
2.2	Provozní podmínky senzoru RSL 200.	29
2.3	Doporučené kombinace barev podkladu a vodicích stop pro senzor OGS 600.	32

# Seznam videoukázek

1	<a href="https://youtu.be/S_b6215s_JA">https://youtu.be/S_b6215s_JA</a>	93
2	<a href="https://youtu.be/m8dPczKWj1Y">https://youtu.be/m8dPczKWj1Y</a>	108
3	<a href="https://youtu.be/TdZPEyCgbRs">https://youtu.be/TdZPEyCgbRs</a>	114
4	<a href="https://youtu.be/In4ohmL6BnM">https://youtu.be/In4ohmL6BnM</a>	133
5	<a href="https://youtu.be/Ek70PZHnybw">https://youtu.be/Ek70PZHnybw</a>	134
6	<a href="https://youtu.be/FsqOBiu9Me4">https://youtu.be/FsqOBiu9Me4</a>	142
7	<a href="https://youtu.be/N6Oc99miTk4">https://youtu.be/N6Oc99miTk4</a>	144
8	<a href="https://youtu.be/BV9gnHkLGfw">https://youtu.be/BV9gnHkLGfw</a>	145
9	<a href="https://youtu.be/Bhdc4WxN1-g">https://youtu.be/Bhdc4WxN1-g</a>	145
10	<a href="https://youtu.be/5y_YiSTZgvs">https://youtu.be/5y_YiSTZgvs</a>	145
11	<a href="https://youtu.be/Ent6ZXx1k0U">https://youtu.be/Ent6ZXx1k0U</a>	147
12	<a href="https://youtu.be/e3kIIHGrRsQ">https://youtu.be/e3kIIHGrRsQ</a>	149
13	<a href="https://youtu.be/Ent6ZXx1k0U">https://youtu.be/Ent6ZXx1k0U</a>	149
14	<a href="https://youtu.be/1445HYQfIKg">https://youtu.be/1445HYQfIKg</a>	150
15	<a href="https://youtu.be/8kM1mF827_4">https://youtu.be/8kM1mF827_4</a>	150
16	<a href="https://youtu.be/gN5MGQDTLaU">https://youtu.be/gN5MGQDTLaU</a>	160
17	<a href="https://youtu.be/RJeJiCzLGpE">https://youtu.be/RJeJiCzLGpE</a>	160
18	<a href="https://youtu.be/hkSZ4L7MABU">https://youtu.be/hkSZ4L7MABU</a>	160
19	<a href="https://youtu.be/A5xVp2POyXs">https://youtu.be/A5xVp2POyXs</a>	161
20	<a href="https://youtu.be/aiPO5do6CC8">https://youtu.be/aiPO5do6CC8</a>	161
21	<a href="https://youtu.be/HEZgW6SYUNU">https://youtu.be/HEZgW6SYUNU</a>	161
22	<a href="https://youtu.be/SkuWGRKmKqk">https://youtu.be/SkuWGRKmKqk</a>	161
23	<a href="https://youtu.be/Tj8zgDCbYXM">https://youtu.be/Tj8zgDCbYXM</a>	161
24	<a href="https://youtu.be/ukYFQAh4GeA">https://youtu.be/ukYFQAh4GeA</a>	161
25	<a href="https://youtu.be/Ek70PZHnybw">https://youtu.be/Ek70PZHnybw</a>	161
26	<a href="https://youtu.be/UupdoPTlnb8">https://youtu.be/UupdoPTlnb8</a>	165
27	<a href="https://youtu.be/95Joo2YPU7k">https://youtu.be/95Joo2YPU7k</a>	165
28	<a href="https://youtu.be/TdZPEyCgbRs">https://youtu.be/TdZPEyCgbRs</a>	167
29	<a href="https://youtu.be/XpXaTvXdGPs">https://youtu.be/XpXaTvXdGPs</a>	169
30	<a href="https://youtu.be/diw-rZ_1rrY">https://youtu.be/diw-rZ_1rrY</a>	169
31	<a href="https://youtu.be/Kb0JmeryOV4">https://youtu.be/Kb0JmeryOV4</a>	176
32	<a href="https://youtu.be/u1_5eXqmrbQ">https://youtu.be/u1_5eXqmrbQ</a>	177

33	<a href="https://youtu.be/gN5MGQDTLaU">https://youtu.be/gN5MGQDTLaU</a>	177
34	<a href="https://youtu.be/hkSZ4L7MABU">https://youtu.be/hkSZ4L7MABU</a>	178

# **Kapitola 1**

## **Specifikace práce**

---

Tato úvodní část práce má seznámit čtenáře s problematikou zkoumaného tématu, definovat konkrétní cíle práce a objasnit směr, kterým se vývoj ubírá. Obsahuje rešerši, jež shrnuje stávající hardwarová a softwarová řešení v oblasti autonomní navigace robotů, přičemž identifikuje jejich přednosti i nedostatky. Práce čerpá inspiraci z běžně používaných přístupů, avšak zaměřuje se i na odhalení mezer v těchto řešeních a navrhuje inovativní a ekonomicky efektivní alternativy, které představují potenciál pro další rozvoj.

# Úvod

Automaticky řízená vozidla (Automated Guided Vehicle, AGV) představují jednu z nově se rozvíjejících technologií v oblasti průmyslové automatizace, která v posledních letech zásadně změnila způsob manipulace s materiélem v továrnách, skladech a logistických centrech. Tato vozidla umožňují autonomní přepravu nákladů bez potřeby lidského řidiče, čímž zvyšují efektivitu provozu, snižují riziko chyb a optimalizují časové i finanční náklady. S příchodem konceptů Průmyslu 4.0, kde hraje zásadní roli propojení zařízení přes internet věcí (IoT) a adaptabilita na dynamicky se měnící podmínky, se však požadavky na AGV značně rozšířily. Tradiční systémy, které spoléhají na pevně vyznačené trasy, jako jsou čáry na podlaze, již nejsou dostatečně flexibilní ani bezpečné pro prostředí, kde se pohybují lidé, překážky, či pro měnící se dispozice skladů. Zmíněná omezení inspirovala tuto práci k tomu, aby přinesla moderní pohled na navigaci AGV a ukázala, jak lze vytvořit systém, který je nejen autonomní a bezpečný, ale také snadno přizpůsobitelný konkrétním potřebám.

Hlavním účelem této práce je navrhnut a otestovat univerzální navigační systém pro AGV, který zvládne pohyb v různých prostředích, detekci palet a bezpečné reakce na překážky, a zároveň posloužit jako praktický manuál pro ty, kdo chtějí podobné řešení vybudovat. Práce reaguje na potřebu překlenout propast mezi tradičními a plně autonomními přístupy – například kombinací jednoduchosti virtuálních drah s pokročilou lokalizací a plánováním trajektorií. Zároveň zkoumá další navigační techniky, jako je jízda po čáře, využití virtuálních čar nebo detekce objektů čistě z 2D LIDARu. Výsledný systém je navržen tak, aby fungoval na dvou běžných typech podvozků – diferenciálním, vhodném pro standardní aplikace, a všeobecném s mecanum koly, který vyniká manévrovatelností ve stísněných prostorách. Celkem byly vytvořeny čtyři prototypy podvozků, které mechanicky i softwarově pokrývají širokou škálu AGV aplikací, odrážejí současné trendy v praxi a připravují půdu pro budoucí technologie. Jedná se tedy o poměrně rozsáhlou škálu AGV aplikací, které mohou pomoci v různých situacích. Výsledně čtyři podvozky tak pokrývají jak mechanicky tak softwarově velkou oblast využitelnosti a odražejí soudobé trendy v praxi i chystají půdu pro budoucí technologie. Kromě funkčního řešení pro průmyslové nasazení práce nabízí detailní dokumentaci hardwarových a softwarových komponent, včetně shrnutí principů algoritmů, norem a použitého open-source frameworku ROS2. Tím poskytuje ucelený úvod do problematiky, který eliminuje potřebu zdlouhavého prostudování rozsáhlých dokumentací, a slouží jako inspirace pro vývojáře vlastních AGV systémů, čímž přispívá nejen k technickému pokroku, ale i ke zpřístupnění know-how v oblasti autonomní robotiky.

Pro dosažení těchto cílů práce zkoumá teoretické základy navigačních algoritmů, jako je plánování a sledování trajektorií nebo lokalizace, a aplikuje je pomocí ROS2 a knihovny Nav2. Dále analyzuje vhodný hardware, včetně LIDARů a senzorů od firmy Leuze, a testuje systém jak v simulačním prostředí Gazebo, tak na reálných prototypech. Struktura práce začíná rešerší současných řešení, pokračuje popisem hardwaru, úvodem do ROS2, teoretickým odvozením potřebných metod a končí ověřením v simulacích a praktickou implementací. Výsledkem je robustní systém připravený pro praxi a dokument, který usnadňuje jeho hlubší pochopení a další rozvoj v oblasti autonomní robotiky.

## Poznámka autora

Automaticky řízená vozidla (AGV) a autonomní mobilní roboty (AMR) představují dvě odlišné kategorie technologií pro automatizaci interní logistiky, přičemž jejich hlavní rozdíly spočívají v úrovni flexibility, navigačních schopnostech a požadavcích na infrastrukturu. AGV jsou tradičním řešením, které se pohybuje po pevně definovaných trasách, vyznačených například indukčními smyčkami, magnetickými páskami nebo optickými senzory. Tyto trasy vyžadují náročnou instalaci, která může být finančně i časově nákladná a často omezuje provoz při změnách dispozice prostředí. AGV disponují minimální palubní inteligencí, což znamená, že se řídí pouze jednoduchými programovacími pokyny a jejich reakce na překážky jsou omezené – dokážou překážku detektovat, ale nejsou schopny ji autonomně překonat bez zásahu operátora. Změna trasy u AGV je navíc spojena s dalšími náklady a časovou náročností, což je činí méně vhodnými pro dynamická prostředí.

Na rozdíl od AGV jsou AMR vybaveny pokročilými navigačními technologiemi, které zahrnují kamery, senzory, laserové skenery a sofistikovaný software pro tvorbu map a autonomní navigaci. AMR se nespolehají na pevně vyznačené trasy, ale jsou schopny se v prostoru orientovat, detektovat překážky a bezpečně kolem nich manévrovat výběrem alternativní trasy. To je činí ideální volbou pro dynamická prostředí, jako jsou moderní sklady nebo výrobní haly s proměnlivými podmínkami. Kromě toho AMR nabízejí vyšší efektivitu z hlediska provozních nákladů a rychlejší návratnost investic, zejména ve vícesmenných provozech, kde mohou díky optimalizaci nabíjení fungovat až 24 hodin denně. Zatímco AGV jsou typicky omezeny na „vozíková“ řešení pro přepravu materiálu, kategorie AMR zahrnuje i ostatní pokročilé roboty, kam mohou spadat rovněž humanoidy.

V rámci této práce je obecně používán termín AGV pro označení všech uvažovaných robotů s libovolným typem podvozku, ačkoliv některé jejich vlastnosti, zejména v kontextu autonomní navigace, odpovídají spíše definici AMR. Tento přístup je zvolen z několika důvodů. Zaprvé, termín AGV byl původně zaveden a běžně používán jak v odborné literatuře, tak ve firemním prostředí, kde tato práce vznikala, a zůstává tedy zavedenou konvencí. Zadruhé, vyvinuté podvozky zahrnují širokou škálu aplikací, z nichž některé jednoznačně spadají do kategorie AGV – například navigace podél optických vodicích stop pomocí senzoru OGS nebo detekce předem definovaných kódů senzorem DCR, což jsou typické úkoly pro AGV. Současně však podvozky využívají pokročilé navigační techniky, jako je autonomní plánování trajektorií a detekce překážek pomocí LIDARů RSL 200 a RSL 400, které jsou charakteristické pro AMR. Použití obecného termínu AGV tak reflekтуje širší záběr práce, která kombinuje tradiční i moderní přístupy, a zároveň zohledňuje historický kontext a terminologii firmy. Jsem si vědom rozdílů mezi AGV a AMR, jak byly popsány výše, a volba termínu AGV slouží k zachování konzistence a srozumitelnosti v rámci celé práce.

## 1.1 Rešerše

Tato kapitola se věnuje průzkumu stávajících hardwarových (HW) a softwarových (SW) řešení v oblasti autonomních skladových vozidel (AGV), která slouží jako inspirační základ pro návrh univerzálního systému navigace v této práci. Cílem projektu je vyvinout flexibilní software pro podjezdová AGV schopná přepravy nákladu v průmyslových provozech s dynamickým prostředím, kde se pohybují lidé, při splnění bezpečnostních norem a optimalizaci nákladů. Rešerše analyzuje dostupná řešení s ohledem na požadavky uvedené v následující podkapitole, identifikuje jejich silné stránky, nedostatky a potenciál pro inovace v navrhovaném systému.

### 1.1.1 Požadavky kladené na funkce AGV

#### Typ podvozku:

- Konstrukce musí být nízká, plochá a vybavená prostorem pro zvedací plošinu, což odpovídá podjezdovým AGV určeným k podjíždění palet nebo regálů.
- Software musí být univerzální, podporující jak diferenciální podvozek, tak vše směrový podvozek s mecanum koly, aby byl použitelný pro různé konstrukce.

#### Autonomní navigace:

- AGV operuje ve vnitřním prostředí, jako jsou průmyslové haly a sklady, s dynamicky se měnícími podmínkami.
- Prostředí zahrnuje pohyb lidí, což vyžaduje soulad s bezpečnostními normami, např. ISO 13849 [17], a implementaci bezpečnostních opatření proti kolizím.
- Systém musí být schopen vytvořit, uložit a načíst mapu prostředí a autonomně se v ní orientovat na základě zadaných příkazů.
- Navigace využívá laserové skenery Leuze RSL200 nebo RSL400; další senzory též firmy jsou přípustné, ale z ekonomických důvodů je jejich počet minimalizován.
- AGV musí autonomně plánovat trajektorii z počáteční polohy na zadáný cíl v mapě s ohledem na bezpečnost a při detekci překážek trajektorii přeplánovat.

#### Práce s nákladem:

- AGV musí detektovat vysoké palety nebo regály (např. stojany na Euro palety nebo specifické typy pro podjezdová AGV) a autonomně pod ně zajíždět.
- Detekce by měla ideálně probíhat pouze pomocí LIDARů; přídavné senzory jsou přípustné pouze v případě nutnosti.
- Palety mohou být umístěny libovolně; systém musí neustále skenovat prostor, detektovat palety a po potvrzení operátorem je naložit.
- Při manipulaci s nákladem jsou omezeny bezpečnostní zóny, což vyžaduje nízkou rychlosť a dynamické přepínání bezpečnostních opatření.

### **Jízda po čáře:**

- AGV musí využívat senzory Leuze OGS (sledovač čáry) a DCR (čtečka QR kódů) pro navigaci po předem definovaných trasách.
- Maximální rychlosť na rovných úsecích je 0.7 m/s, v zatáčkách 0.3 m/s, s minimálním poloměrem zatáčky 50 cm.
- Při těchto rychlostech musí být AGV schopno načíst a porovnat trojici QR kódů na rovných úsecích pro zajištění bezpečného načtení informací.
- QR kódy určují rychlosť a signalizují křížovatky, které jsou definovány dvěma rovnoběžnými čarami, jež se následně rozcházejí.
- I tato aplikace musí splňovat bezpečnostní normy pro prostředí s pohybem lidí.

## 1.1.2 Existující řešení

Tato sekce analyzuje vybraná podjezdová AGV a AMR dostupná na trhu s důrazem na jejich navigaci, bezpečnost a vhodnost pro definované požadavky. Zahrnuje řešení od firem SSI SCHÄFER, Linde Material Handling, KUKA, Asseco CEIT, a nově přidané alternativy od ServisControl, ABB, Toyota a KVADOS Robotics. Cílem je identifikovat silné stránky a nedostatky, které mohou inspirovat vlastní návrh nebo odhalit mezery pro inovace.

### 1.1.2.1 SSI SCHÄFER WEASEL

#### Navigace:

Firma SSI SCHÄFER nabízí řadu AGV, mezi které patří i model WEASEL, určený pro přepravu menších nákladů. WEASEL využívá optickou navigaci, při níž se pohybuje podél vyznačené čáry na podlaze, kterou snímá pomocí optických senzorů. Tato čára může být rychle a jednoduše aplikována a přizpůsobena podle potřeb provozu, což zajišťuje vysokou flexibilitu při změnách tras. WEASEL dosahuje rychlosti až 1 m/s a zvládá stoupání až 20 %. Na rozdíl od pokročilejších AMR systémů nevyužívá SLAM (Simultaneous Localization and Mapping) ani laserovou navigaci, což omezuje jeho autonomii, ale zjednoduší implementaci a snižuje náklady. Transportní příkazy jsou generovány bud' manuálně (pomocí rádiových příkazů), nebo prostřednictvím externích systémů, jako jsou PLC nebo WMS (Warehouse Management Systems), a jsou řízeny flotilovým regulátorem.



Obrázek 1.1: Ilustrační obrázek SSI SCHÄFER WEASEL [1]

#### Bezpečnost:

SSI SCHÄFER WEASEL je vybaven bezpečnostními prvky, které splňují normy ISO 13849 a IEC 61508 [18]. Bezpečnostní systém zahrnuje optické senzory pro detekci překážek na trase, avšak WEASEL není schopen překážky autonomně objíždět, ale v případě detekce se zastaví. Bezpečnostní komunikace je zajištěna prostřednictvím standardizovaných protokolů, jako je PROFIsafe integrovaný v PROFINETu, což umožňuje rychlý přenos dat mezi senzory a řídicí jednotkou. WEASEL také podporuje integraci s periferními zařízeními (např. požární systémy, brány), což zvyšuje bezpečnost v prostředí s pohybem lidí.

### **Použití:**

WEASEL je navržen pro přepravu menších nákladů (do 35 kg), jako jsou přepravky, kartony nebo závěsné zboží, a je vhodný pro prostředí, kde je potřeba flexibilní a jednoduchá automatizace transportních úkolů, například v komisionovacích nebo výrobních provozech. Díky své nenáročné infrastruktuře (optická čára) a možnosti rychlé změny tras je ideální pro sklady s proměnlivými požadavky. WEASEL však není primárně určen pro podjezdové aplikace s vysokými paletami, což omezuje jeho využití v některých scénářích požadovaných v této práci.

### **Výhody:**

1. Jednoduchá a rychlá instalace optické trasy bez nutnosti složité infrastruktury.
2. Flexibilita při změnách tras a škálovatelnost systému (možnost přidání vozíků).
3. Nízké pořizovací a provozní náklady ve srovnání s pokročilými AMR.
4. Integrace s WMS a ERP systémy prostřednictvím flotilového regulátoru.

### **Nedostatky:**

WEASEL je závislý na pevně vyznačených trasách, což omezuje jeho autonomii a schopnost reagovat na dynamické změny v prostředí. V případě překážek na trase vyžaduje zásah operátora, což může zpomalit provoz. Navíc není vhodný pro podjezdové aplikace s vysokými paletami, protože je navržen pro přepravu menších nákladů, což nesplňuje všechny požadavky této práce.

#### **1.1.2.2 Linde Material Handling C-Matic**

##### **Navigace:**

Linde Material Handling C-Matic využívá navigaci založenou na QR kódech (tzv. grid localization), které jsou umístěny na podlaze v intervalech přibližně jednoho metru. Tyto QR kódy mají unikátní identifikátor a slouží jako referenční body na digitální mapě, na základě které software vypočítává optimální trasu pro každý přepravní úkol. C-Matic se tak pohybuje po předem definovaných trasách, což zajišťuje spolehlivost v provozech s pevnou strukturou. Tento systém však neumožňuje plnou autonomii, protože vozík je závislý na fyzické infrastruktuře QR kódů a není schopen autonomně mapovat prostředí nebo reagovat na výrazné změny mimo vyznačené trasy.



Obrázek 1.2: Ilustrační obrázek Linde C-Matic [2]

### **Bezpečnost:**

C-Matic je vybaven laserovým skenerem, který detekuje pohyblivé i statické překážky v reálném čase. Pokud je překážka detekována, vozík bud' zpomalí, nebo se zastaví, dokud překážka nezmizí, což zajišťuje bezpečnost v prostředí s pohybem lidí a jiných vozíků. Dále je vybaven robustním systémem ochrany proti kolizím, zejména při otáčení na místě a couvání, a nouzovými vypínači na všech čtyřech stranách, které umožňují okamžité zastavení v případě potřeby. Bezpečnostní systém splňuje normy, jako je ISO 3691-4 [19], a je navržen pro bezpečnou interakci s lidmi a dalšími zařízeními v provozu.

### **Použití:**

C-Matic je ideální pro provozy s předem danými trasami, kde je klíčová spolehlivost a jednoduchost implementace. Je navržen pro přepravu nákladů až 1500 kg na krátké a střední vzdálenosti, zejména v prostředích s omezeným prostorem, jako jsou sklady nebo výrobní haly. Vozík dokáže podjízdět transportní stoly a přepravovat náklad mezi statickými překládacími stanicemi nebo dopravníky. Díky svým kompaktním rozměrům a bezpečnostní technologii je vhodný i pro prostředí, kde se pohybují lidé a další vozíky. Nabízí také snadnou integraci s automatizačními systémy a pokročilý fleet management.

### **Výhody:**

1. Spolehlivá navigace pomocí QR kódů v provozech s fixními trasami.
2. Kompaktní rozměry umožňují použití v prostředích s omezeným prostorem.
3. Robustní bezpečnostní systém zajišťuje bezpečnou interakci s lidmi a jinými vozíky.
4. Snadná integrace s WMS a ERP systémy díky softwarovému řízení tras.

### **Nedostatky:**

Navigace je závislá na fyzických QR kódech, což omezuje flexibilitu v dynamicky se měnících prostředích a vyžaduje údržbu těchto značek (např. čištění, výměnu při poškození). C-Matic není schopen autonomně obcházet překážky mimo vyznačenou trasu, což může vést k zastavení provozu a nutnosti zásahu operátora. Navíc není primárně navržen pro vysoce dynamická prostředí, kde by byla potřeba plná autonomie, jako u systémů s pokročilou SLAM navigací.

### **1.1.2.3 KUKA KMP 600-S diffDrive**

#### **Navigace:**

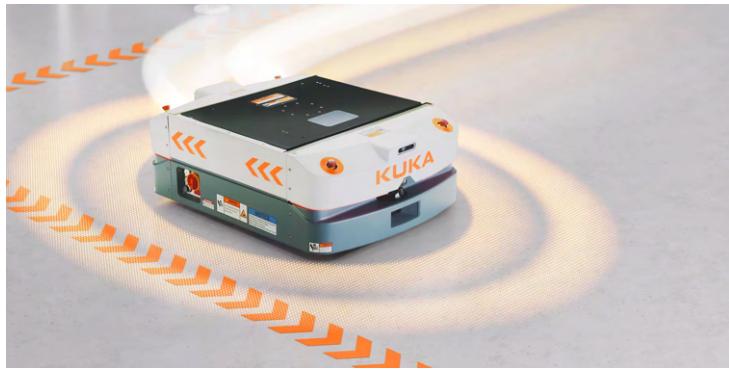
KUKA KMP 600-S diffDrive využívá SLAM s LIDARy pro autonomní navigaci a mapování, doplněné o odometrii pro zpětnou vazbu polohy. LIDARy umožňují vytváření detailních map prostředí a přesnou lokalizaci, zatímco odometrie zvyšuje přesnost při pohybu v pevně definovaných trasách.

#### **Konstrukce:**

Diferenciální podvozek s nosností 600 kg, navržený pro nízký profil a podjezdové aplikace. Konstrukce je optimalizována pro přepravu palet a regálů v průmyslových provozech.

### **Bezpečnost:**

Safety PLC (splňující ISO 13849) s protokolem Safety over EtherCAT (FSoE) řídí bezpečnostní zóny a detekci překážek pomocí LIDARů a ultrazvukových senzorů. FSoE zajišťuje rychlou a spolehlivou komunikaci mezi senzory a řídicí jednotkou, což umožňuje okamžitou reakci na překážky.



Obrázek 1.3: Ilustrační obrázek KMP 600-S diffDrive [3]

### **Použití:**

Určeno pro sklady s pevnými trasami a podjezdovými úkoly, s důrazem na integraci do průmyslových systémů. KMP 600-S je vhodný pro prostředí, kde je klíčová spolehlivost a bezpečnost při manipulaci s nákladem.

### **Výhody:**

1. Jednoduchá konstrukce diferenciálního podvozku usnadňuje údržbu.
2. Rychlá reakce díky FSoE protokolu.
3. Vhodné pro podjezdová AGV díky nízkému profilu.

### **Nedostatky:**

Omezená flexibilita pohybu kvůli diferenciálnímu podvozku, což může být nevýhodou ve srovnání s vše směrovými podvozky s mecanum koly. Navigace není plně optimalizována pro vysoko dynamická prostředí s častými změnami.

### **Navigace:**

Asseco CEIT AMR U1500XN využívá přirozenou navigaci (natural navigation) založenou na 2D laserových skenerech, které skenují prostředí a vytvářejí detailní mapu okolí. Tato technologie umožňuje vozíku orientovat se v prostoru bez nutnosti fyzické infrastruktury, jako jsou magnetické pásky nebo optické čáry, a detektovat překážky v reálném čase. Laserové skenery monitorují prostor kolem vozíku a zajišťují detekci překážek ve vzdálenosti až několika metrů, což umožňuje bezpečné manévrování v dynamickém prostředí.

### **Konstrukce:**

AMR U1500XN je podjezdový robot s nosností až 1500 kg, navržený pro manipulaci s paletami a regály. Konstrukce je nízká a plochá, což odpovídá požadavkům na podjezdová AGV. Vozík je vybaven vše směrovým pohonem, pravděpodobně s využitím mecanum kol, což umožňuje pohyb do všech směrů, včetně bočního a diagonálního pohybu, a zvyšuje flexibilitu v těsných prostorech.

### **Bezpečnost:**

AMR U1500XN je vybaven dvěma bezpečnostními laserovými skenery (jeden na každé straně), které zajišťují 360° detekci překážek. Safety PLC, certifikované podle ISO 13849-1 a IEC 62061 [20], monitoruje kritické funkce, jako je rychlosť, směr a detekce překážek. Komunikace mezi PLC a senzory probíhá přes SafetyBUS P nebo PROFIsafe, což zajišťuje rychlé a bezpečné reakce v reálném čase. Vozík je dále vybaven dynamickými bezpečnostními zónami, které se přizpůsobují rychlosti a směru pohybu, a akustickými a vizuálními varovnými signály pro bezpečnou interakci s lidmi.



Obrázek 1.4: Ilustrační obrázek AMR U1500XN [4]

### **Použití:**

Vhodné pro montážní linky a logistické procesy, kde je klíčová možnost autonomního a bezpečného pohybu v malých a komplikovaných prostorách. Díky všeobecnému pohonu je AMR U1500XN ideální pro prostředí s těsnými uličkami a potřebou přesného manévrování, například při podjízdění palet nebo regálů. Model nabízí indukční nabíjení, což umožňuje dlouhodobý a bezúdržbový provoz.

### **Výhody:**

1. Přirozená navigace bez nutnosti fyzické infrastruktury zvyšuje flexibilitu nasazení.
2. Všeobecný pohon umožňuje přesné manévrování v těsných prostorech.
3. Indukční nabíjení zvyšuje efektivitu a snižuje potřebu údržby.
4. Robustní bezpečnostní systém s 360° detekcí překážek a dynamickými zónami.

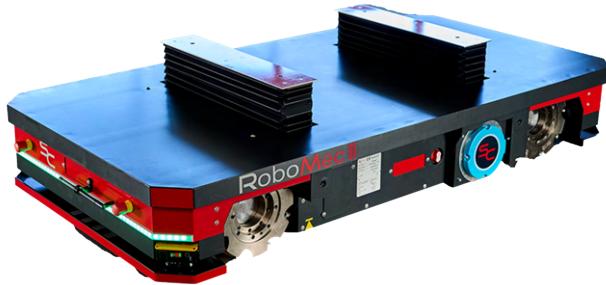
### **Nedostatky:**

Přirozená navigace může být méně efektivní ve vysoce dynamických prostředích s častými změnami ve srovnání s plnohodnotným SLAM, což může vést k omezení adaptability. Omezený počet senzorů (pouze 2D laserové skenery) může být nevýhodou při detekci složitých nebo nízkých překážek, které nejsou v rovině skenu.

#### 1.1.2.4 ServisControl

##### Navigace:

ServisControl, česká firma zaměřená na průmyslovou automatizaci, nabízí AMR řešení, jako je model RoboMec, který využívá přirozenou navigaci (natural navigation) založenou na laserových skenerech (LIDAR). Tato technologie umožňuje vozíku vytvářet mapu prostředí a pohybovat se autonomně bez nutnosti fyzické infrastruktury, jako jsou magnetické pásky nebo optické čáry. LIDAR skenuje okolí a zajišťuje detekci překážek v reálném čase, což umožňuje základní autonomní pohyb a vyhýbání se překážkám v dynamickém prostředí.



Obrázek 1.5: Ilustrační obrázek RoboMec [5]

##### Konstrukce:

RoboMec je nízkoprofilový platformový AMR s nosností až 1000 kg, navržený pro přepravu nákladů v průmyslových provozech. Konstrukce je optimalizována pro podjezdové aplikace, s výškou platformy umožňující podjíždění palet nebo regálů. Vozík je vybaven všeobecným pochodem s mecanum koly, což umožňuje pohyb do všech směrů, včetně bočního a diagonálního pohybu, a zvyšuje flexibilitu v těsných prostorech.

##### Bezpečnost:

RoboMec je vybaven laserovými bezpečnostními skenery, které monitorují prostor kolem vozíku a definují dvě varovné zóny a jednu bezpečnostní zónu pro zastavení v případě detekce překážky. Dále obsahuje bezpečnostní nárazníky pro eliminaci rizika kolize. Safety PLC splňuje normy ISO 13849 a EN 3691-4 a zajišťuje rychlé reakce na potenciální rizika. Bezpečnostní systém je zařazen do kategorie SIL 2/PL d – Cat.3, což zajišťuje bezpečnou interakci s lidmi a dalšími zařízeními v provozu.

##### Použití:

RoboMec je vhodný pro sklady a výrobní haly, kde je klíčová flexibilita a přesné manévrování v omezeném prostoru. Díky všeobecnému pochodu a nízkému profilu zvládá podjezdové aplikace, jako je přeprava palet nebo regálů, a je ideální pro prostředí s dynamickým pohybem lidí a jiných vozíků. Vozík je vybaven bezkontaktním nabíjením s výkonem 500 W (s možností upgrade na 1500 W), což zajišťuje efektivní a bezúdržbový provoz.

##### Výhody:

1. Přirozená navigace s LIDAR umožňuje autonomní pohyb bez fyzické infrastruktury.
2. Všeobecný pochodek s mecanum koly zvyšuje flexibilitu v těsných prostorech.

3. Bezkontaktní nabíjení zvyšuje efektivitu a snižuje potřebu údržby.
4. Robustní bezpečnostní systém zajišťuje bezpečnou interakci s lidmi.

#### **Nedostatky:**

Přirozená navigace může být méně efektivní ve vysoce dynamických prostředích s častými změnami ve srovnání s pokročilými SLAM systémy, což může omezit její adaptabilitu. Omezený počet senzorů může být nevýhodou při detekci složitých nebo nízkých překážek, které nejsou v rovině skenu.

#### **1.1.2.5 ABB AMR Solutions**

##### **Navigace:**

ABB nabízí řadu AMR, například model Flexley Tug, který využívá pokročilou SLAM navigaci s LIDARy a 3D kamerami. Systém umožňuje autonomní mapování prostředí a plánování tras v reálném čase, což zajišťuje vysokou flexibilitu v dynamických provozech. Navigace je doplněna o senzory pro detekci překážek a bezpečné manévrování kolem nich.



Obrázek 1.6: Ilustrační obrázek Flexley Tug [6]

##### **Konstrukce:**

Flexley Tug je navržen pro tahání vozíků, ale ABB nabízí i podjezdové modely, jako je Flexley Mover, s nosností až 2000 kg. Konstrukce je nízká a vybavená zvedací plošinou, což odpovídá požadavkům na podjezdová AGV.

##### **Bezpečnost:**

ABB využívá Safety PLC s certifikací podle ISO 13849 a IEC 61508. Bezpečnostní prvky zahrnují laserové skenery, 3D kamery, nouzové zastavení a varovné systémy. Protokol Safety over EtherCAT (FSoE) zajišťuje rychlou komunikaci mezi senzory a řídicí jednotkou, což umožňuje okamžitou reakci na překážky.

##### **Použití:**

Vhodné pro dynamická prostředí, jako jsou sklady a výrobní haly, kde je potřeba vysoká flexibilita a bezpečnost. Flexley Mover je ideální pro podjezdové aplikace s paletami a regály.

##### **Výhody:**

1. Vysoká flexibilita díky SLAM navigaci a autonomnímu plánování tras.
2. Robustní bezpečnostní systém s redundantními senzory.
3. Možnost škálování a integrace s průmyslovými systémy.

### **Nedostatky:**

Vyšší pořizovací náklady kvůli pokročilým technologiím. Složitost systému může zvyšovat nároky na údržbu a školení obsluhy.

#### **1.1.2.6 Toyota Autopilot**

##### **Navigace:**

Toyota Autopilot CDI120 využívá přirozenou navigaci (natural navigation) založenou na laserovém skeneru umístěném blízko podlahy. Tato technologie umožňuje vozíku orientovat se v prostoru pomocí rozpoznávání statických prvků prostředí, jako jsou stěny, regály nebo pevné objekty, bez nutnosti dodatečné infrastruktury, například magnetických pásek nebo optických čar. CDI120 kombinuje laserovou navigaci s odometrií pro přesnou lokalizaci a autonomní pohyb, přičemž je schopen detektovat překážky v reálném čase a přizpůsobit svou trasu.



Obrázek 1.7: Ilustrační obrázek CDI120 [7]

##### **Konstrukce:**

CDI120 je podjezdový horizontální přepravník s nosností až 1200 kg, navržený pro manipulaci s paletami bez vidlic (forkless). Konstrukce je kompaktní a nízká, což umožňuje práci v těsných prostorách, kde by jiné AGV nebo manuální vozíky měly problémy s manévrováním. Vozík je vybaven diferenciálním pohonem, který zajišťuje přesné otáčení na místě, a drive-through stanicemi, které umožňují přímý průjezd při vyzvednutí a uložení palet. CDI120 je napájen lithium-iontovými bateriemi, které jsou až o 30 % energeticky úspornější a podporují automatické nabíjení (scheduled a opportunity charging).

##### **Bezpečnost:**

CDI120 využívá Safety PLC certifikované podle norem ISO 13849 a ISO 3691-4, což zajišťuje vysokou úroveň funkční bezpečnosti. Bezpečnostní systém zahrnuje laserové skenery s detekčními poli vpředu a po stranách, které monitorují prostor a přizpůsobují bezpečnostní zóny směru a rychlosti pohybu. Vozík je dále vybaven modrým LED varovným světlem, které upozorňuje na jeho přítomnost, a nouzovými vypínači (E-stops) pro okamžité zastavení v případě potřeby. Tyto prvky zajišťují bezpečnou interakci s lidmi a jinými vozíky v provozu.

##### **Použití:**

CDI120 je vhodný pro sklady a výrobní haly s opakovánými přepravními úkoly typu A-B nebo doplňování zásob (replenishment). Díky podjezdové konstrukci a drive-through stanicím může přepravovat palety přímo pod regály, což optimalizuje využití prostoru. Vozík je navržen

pro prostředí, kde je klíčová efektivita a jednoduchá integrace s WMS systémy nebo Toyota T-ONE softwarem pro správu flotily, což umožňuje koordinaci s dalšími Autopilot vozíky, například automatickými stohovači.

### Výhody:

1. Efektivní přeprava v těsných prostorech díky kompaktnímu designu a diferenciálnímu pohonu.
2. Energeticky úsporné lithium-iontové baterie s automatickým nabíjením.
3. Snadná integrace s WMS a T-ONE softwarem pro optimalizaci logistiky.

### Nedostatky:

Přirozená navigace není tak pokročilá jako plnohodnotný SLAM, což může omezit flexibilitu v komplexních a vysoce dynamických prostředích. CDI120 není všeobecný, což může být nevhodné ve srovnání s AMR s mecanum koly, zejména v prostředích vyžadujících boční nebo diagonální pohyb.

#### 1.1.2.7 myFABER

##### Navigace:

myFABER Q3-600, distribuovaný společností KVADOS Robotics, je autonomní mobilní robot (AMR), který využívá kombinaci inerciální a vizuální navigace pro přesnou lokalizaci a pohyb v průmyslových prostředích. Inerciální navigace zahrnuje senzory pro měření zrychlení a otáčení, zatímco vizuální navigace využívá přední kamery a laserové skenery pro rozpoznávání prostředí a detekci překážek. Tento hybridejný přístup umožňuje robotu pohybovat se v komplexních prostorech s vysokou přesností, a to i bez nutnosti fyzické infrastruktury, jako jsou magnetické pásky nebo optické čáry. Robot je schopen autonomě navigovat a vyhýbat se překážkám v reálném čase, což zvyšuje jeho flexibilitu v dynamických provozech.



Obrázek 1.8: Ilustrační obrázek myFABER Q3-600 [8]

##### Konstrukce:

myFABER Q3-600 je podjezdový AMR s nosností až 600 kg, navržený pro manipulaci s paletami a jinými náklady v průmyslových skladech. Jeho konstrukce je nízká a kompaktní, s hmotností 145 kg, což umožňuje podjíždění regálů a efektivní využití prostoru. Robot je vybaven dvoukolovým diferenciálním pohonem, který zajišťuje plynulý a kontrolovaný pohyb i v těsných prostorech, a umožňuje otáčení na místě. Díky svému designu je ideální pro podjezdové

aplikace, kde je potřeba přesouvat palety mezi pevnými stanicemi nebo dopravníky.

#### **Bezpečnost:**

myFABER Q3-600 je vybaven pokročilým bezpečnostním systémem, který zahrnuje laserové skenery pro detekci překážek, nárazové pásky a tlačítka nouzového zastavení (scram button). Bezpečnostní laserové skenery monitorují prostor kolem robota a vytvářejí dynamické bezpečnostní zóny, které se přizpůsobují rychlosti a směru pohybu. Robot také disponuje přední kamerou a senzory pro měření vzdálenosti, což minimalizuje riziko kolizí v prostředí s pohybem lidí a jiných vozíků. Bezpečnostní systém splňuje normy ISO 13849 a ISO 3691-4, což zajišťuje vysokou úroveň funkční bezpečnosti. Uživatelské rozhraní zahrnuje LCD displej a zvukové výstrahy pro snadnou interakci a monitorování stavu robota.

#### **Použití:**

myFABER Q3-600 je vhodný pro sklady a výrobní haly, kde je klíčová efektivní přeprava palet a jiných nákladů v prostředí s omezeným prostorem. Díky podjezdové konstrukci a autonomní navigaci je ideální pro automatizaci přepravních úkolů, jako je přesun palet mezi dopravníky, regály nebo pracovními stanicemi. Robot podporuje bezdrátovou komunikaci a může se pohybovat v oblastech pokrytých sítí, což usnadňuje jeho integraci do stávajících systémů WMS (Warehouse Management Systems). Díky své nosnosti a flexibilitě je vhodný i pro dynamická prostředí s pohybem lidí a jiných vozíků.

#### **Výhody:**

1. Autonomní navigace bez nutnosti fyzické infrastruktury díky inerciální a vizuální technologii.
2. Kompaktní design a diferenciální pohon umožňují efektivní pohyb v těsných prostorech.
3. Robustní bezpečnostní systém zajišťuje bezpečnou interakci s lidmi a jinými zařízeními.
4. Snadná integrace s WMS systémy díky bezdrátové komunikaci.

#### **Nedostatky:**

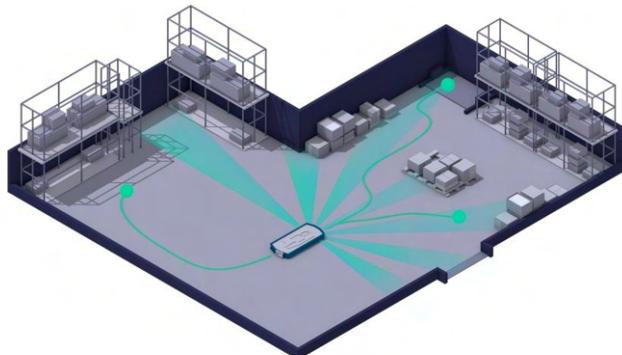
Přirozená navigace může být méně efektivní ve vysoce dynamických prostředích s častými změnami, protože není založena na plnohodnotném SLAM, což může omezit její adaptabilitu. Diferenciální pohon, ačkoliv je přesný, neumožňuje všeobecný pohyb, což může být nevýhodou ve srovnání s AMR vybavenými mecanum koly, zejména v prostředích vyžadujících boční nebo diagonální pohyb.

### 1.1.3 Existující SW řešení pro autonomní navigaci

Tato sekce představuje dva základní softwarové přístupy pro vývoj automaticky naváděných vozidel (AGV), které umožňují implementaci vlastního řešení bez nutnosti programování od základu. Jedná se o komerční systém *SIMOVE* od společnosti Siemens a open-source framework *ROS2*. Oba systémy patří mezi nejrozšířenější nástroje v oblasti vývoje AGV a autonomních mobilních robotů obecně, přičemž každý z nich nabízí specifické výhody a je podpořen širokou uživatelskou komunitou.

#### 1.1.3.1 SIMOVE

*SIMOVE* od Siemens je komplexní řídicí a rozvrhovací systém navržený pro automatizaci a správu AGV a AMR ve skladech a výrobních halách. Jedná se o uzavřené, plně integrované řešení, které je kompatibilní s průmyslovými standardy Siemens, což usnadňuje jeho nasazení v provozech již využívajících technologie této společnosti. Systém je postaven na platformě Siemens SIMATIC S7 PLC, která zajišťuje centrální řízení a monitorování všech vozíků. SIMATIC S7 je certifikován podle bezpečnostních norm ISO 13849 a IEC 61508 a dosahuje úrovně SIL 3 (Safety Integrity Level 3), což zaručuje vysokou úroveň funkční bezpečnosti v průmyslovém prostředí. Komunikace mezi vozíky, senzory a centrálním systémem probíhá přes bezpečnostní protokol PROFIsafe integrovaný do sítě PROFINET, což umožňuje rychlý a spořehlivý přenos dat v reálném čase, například pro detekci překážek nebo nouzové zastavení.



Obrázek 1.9: Ilustrační obrázek SIMOVE [9]

*SIMOVE* podporuje různé typy navigace, včetně laserové navigace s využitím LiDARů, která je vhodná pro dynamická prostředí, a navigace založené na QR kódech, která se uplatňuje v provozech s pevným mřížkovým rozvržením. Kromě toho systém zahrnuje odometrii, která poskytuje zpětnou vazbu o poloze vozíku na základě dat z enkodérů, a umožňuje tak přesné sledování ujeté vzdálenosti a korekci polohy v kombinaci s dalšími navigačními metodami. Na rozdíl od pokročilejších autonomních systémů však *SIMOVE* neumožňuje jednoduché zadání cílového bodu s libovolným plánováním trajektorie; místo toho je trajektorie předem definována – bud' manuálně nakreslena, nebo s možností nastavení povolené odchylky od trasy, což umožňuje omezené objíždění překážek v rámci předem stanovených limitů. *SIMOVE* také není schopen autonomně detektovat palety v prostoru a vyžaduje přesné zadání jejich polohy nadřazeným systémem, což omezuje jeho flexibilitu v dynamických scénářích. Další nevýhodou je absence možnosti přidávat vlastní regulátory, plánovače trajektorií nebo jiné algoritmy, protože systém je uzavřený a neumožňuje takové úpravy. Na druhou stranu *SIMOVE* vyniká v počáteční inicializaci a kalibraci senzorů a parametrů, které jsou díky předpřipraveným nástrojům rychlé a

spolehlivé, a nabízí integrovaný webserver pro pohodlné zadávání povelů a monitorování stavu flotily.

Další součástí SIMOVE je jeho fleet management systém, který optimalizuje trasy, koordinuje pohyb jednotlivých AGV, řeší přednosti na křížovatkách, shromažďuje data o jejich stavu a umožňuje snadné propojení s nadřazenými systémy, jako jsou MES (Manufacturing Execution Systems) nebo WMS (Warehouse Management Systems), což zvyšuje efektivitu logistiky. SIMOVE je tedy ideální volbou pro průmyslové aplikace, kde je klíčová spolehlivost, bezpečnost a rychlá implementace, avšak jeho uzavřená povaha a omezená flexibilita znemožňují přizpůsobení specifickým požadavkům, což může být nevýhodou pro vývoj inovativních řešení vyžadujících volnost v návrhu systému.

### 1.1.3.2 ROS2

Na rozdíl od SIMOVE je ROS2 (Robot Operating System 2) open-source framework, který poskytuje modulární a flexibilní prostředí pro vývoj robotických aplikací, včetně AGV a AMR. ROS2 nabízí širokou škálu balíčků a knihoven, z nichž nejvýznamnější pro vývoj AGV je Nav2 (Navigation 2). Tento balíček zahrnuje pokročilé nástroje pro autonomní navigaci, jako je SLAM (Simultaneous Localization and Mapping), který umožňuje robotu vytvářet mapu prostředí a současně určovat svou polohu v reálném čase, což je zvláště užitečné v dynamických prostředích s pohybem lidí. Nav2 dále podporuje AMCL (Adaptive Monte Carlo Localization), algoritmus pro přesnou lokalizaci robota v již existující mapě, který využívá data z LIDARů, kamer nebo jiných senzorů. Pro plánování trajektorie jsou integrovány algoritmy DWA (Dynamic Window Approach) a TEB (Timed Elastic Band). DWA vybírá optimální cestu na základě aktuální rychlosti a přítomnosti překážek v dynamickém okně, zatímco TEB optimalizuje trasu s ohledem na časové a prostorové omezení, což zajišťuje plynulý a efektivní pohyb.

ROS2 umožňuje vývojářům přizpůsobit navigaci a další funkce podle specifických potřeb, například implementací vlastních algoritmů pro detekci překážek nebo plánování tras. Z hlediska bezpečnosti ROS2 sám o sobě není certifikován podle průmyslových norem, jako jsou ISO 13849 nebo IEC 61508, ale lze jej integrovat s bezpečnostními PLC moduly a protokoly, například přes ROS-Industrial, aby bylo dosaženo požadované úrovně bezpečnosti (např. SIL 2 nebo SIL 3). ROS2 také podporuje distribuovanou architekturu díky svému komunikačnímu protokolu DDS (Data Distribution Service), což umožňuje efektivní správu flotily AGV a jejich koordinaci v reálném čase. Hlavní výhodou ROS2 je jeho otevřenosť a flexibilita, která umožňuje rychlé prototypování a přizpůsobení, avšak vyžaduje vyšší technickou zdatnost a čas na vývoj ve srovnání s hotovým řešením, jako je SIMOVE. Navíc absence přímé podpory od výrobce může komplikovat nasazení v průmyslových provozech s vysokými požadavky na spolehlivost a certifikaci.

## **1.1.4 Konkrétní řešení vybraná na základě průzkumu**

V předchozích částech byl uveden soupis požadavků na vlastní řešení a přehled existujících AGV a softwarových přístupů pro jejich vývoj. Průzkum ukázal, že žádné z dostupných řešení plně nesplňuje všechny požadavky, jako je kombinace autonomní navigace v dynamickém prostředí, podjezdová konstrukce, všesměrový pohyb a flexibilní detekce palet bez závislosti na fyzické infrastruktuře. Z tohoto důvodu je nutné vytvořit vlastní návrh, který však může využít řadu již existujících a ověřených komponent. Následující shrnutí popisuje, jakým způsobem lze splnit jednotlivé požadavky, přičemž některé prvky jsou inspirovány nalezenými řešeními, zatímco jiné představují inovativní přístupy k vyplnění mezer na trhu. V následujících sekčích jsou podrobněji uvedeny konkrétní důvody a principy, které budou použity pro dosažení dílčích cílů.

### **1.1.4.1 Navigace**

Na základě požadavků na autonomní navigaci v dynamickém prostředí s lidmi a překážkami byla zvolena navigace postavená primárně na 2D LIDAR senzorech, konkrétně modelech RSL400 nebo RSL200 od Leuze. Tyto senzory jsou certifikovány pro bezpečnostní aplikace a splňují normy ISO 13849, IEC 61508 a ISO 3691-4, což umožňuje jejich využití jak pro navigaci, tak pro detekci překážek, jak bylo vidět například u modelů Asseco CEIT AMR U1500XN a ServisControl RoboMec, které rovněž využívají LIDAR pro přirozenou navigaci. Zvoleným řešením je technologie SLAM (Simultaneous Localization and Mapping), která umožňuje neustálé mapování prostředí a současnou lokalizaci vozíku v něm, jak je běžné u pokročilejších AMR. Po vytvoření mapy lze tuto mapu znova použít pro přesnou navigaci, což zajišťuje flexibilitu při změnách prostředí, vyhýbání se překážkám a pohyb volným prostorem bez nutnosti fyzické infrastruktury, jako jsou čáry, magnetické pásky nebo RFID tagy, což je výhoda oproti modelům, jako je SSI SCHÄFER WEASEL nebo Linde C-Matic, které jsou závislé na optických trasách nebo QR kódech.

Vozík by měl být podjezdového typu, což odpovídá požadavkům na manipulaci s paletami v těsných prostorech, jak ukazují modely Toyota Autopilot CDI120 a myFABER Q3-600. Průzkum trhu odhalil, že všesměrový pohyb, například pomocí mecanum kol, je běžný u modelů jako Asseco CEIT AMR U1500XN nebo ServisControl RoboMec. Všesměrový pohyb však může přinášet nevýhody, jako je vyšší složitost řízení a potenciální nestabilita při vyšších rychlostech. Z tohoto důvodu bude navržen navigační stack, který podporuje jak klasický diferenciální podvozek, tak i všesměrový podvozek s mecanum koly. Prvním cílem práce je tedy vytvořit univerzální navigační stack schopný zpracovávat data z 2D LIDARů, generovat mapu prostředí a zajistit regulaci pohybu vozíku po zvolené trajektorii. To zahrnuje vývoj kinematických modelů pro oba typy podvozků a přípravu konfiguračních souborů pro snadnou úpravu parametrů, což umožní přizpůsobení různým aplikacím. Kromě autonomní navigace bude navrhena a otestována také jízda po čáře, která je stále široce využívána v praxi, jak ukazuje příklad SSI SCHÄFER WEASEL, a může sloužit jako záložní řešení pro prostředí s pevnými trasami.

### **1.1.4.2 Detekce palety**

Navrhované řešení zahrnuje přemisťování palet, což je běžnou praxí u podjezdových AGV. V praxi existují dva hlavní přístupy k detekci palet. První z nich, viděný například u Linde C-Matic, spoléhá na nadřazený systém, který přesně definuje polohu palety, a vozík poté provádí finální dománevrování (fine positioning) pomocí senzorů, jako jsou čtečky QR kódů nebo magnetické značky. Druhý přístup, často využívaný u pokročilejších AMR, zahrnuje detekci

palety z obrazových dat pomocí kamer, což však přináší problémy s ovlivnitelností světelnými podmínkami a vyžaduje výkonný hardware, jak ukazují zkušenosti z průmyslových aplikací.

Druhým cílem práce je vyvinout systém detekce palety, která může být libovolně umístěna v prostoru, a to pouze na základě 2D LIDARových dat, čímž se vyhneme použití kamer a jejich nevýhodám. Tento přístup je inovativní, průzkum neodhalil žádné existující řešení, které by detekci palet řešilo výhradně pomocí LIDARů. Bude tedy nutné ověřit, zda je možné dosáhnout dostatečné přesnosti při rozpoznávání tvaru palety a určení její polohy v prostoru. Následné dokování pod paletou by mělo být z ekonomických důvodů rovněž řešeno bez dalších senzorů, avšak pokud by přesnost LIDARové detekce nebyla dostatečná, může být nutné přidat senzory pro fine positioning, jako jsou čtečky QR kódů nebo magnetické značky, což je běžná praxe u modelů, jako je Linde C-Matic.

#### 1.1.4.3 Simulace

Před nasazením navrženého řídicího systému na reálný vozík je nezbytné otestovat jeho funkčnost v simulovaném prostředí. Třetím cílem práce je proto vytvoření komplexní simulace, která umožní otestovat navigační algoritmy, detekci palet a regulaci pohybu před jejich aplikací na fyzický systém. Simulace by měla zahrnovat vizualizaci prostředí a pohybu vozíku, přičemž by měla zohledňovat reálné fyzikální parametry, jako jsou hmotnost, setrvačnost a tření, aby co nejpřesněji simulovala chování vozíku v praxi. Tento přístup je inspirován výhodami ROS2, který nabízí široké možnosti simulací, například pomocí nástrojů jako Gazebo, což umožňuje testování různých balíčků funkcí a jejich následné přenesení na reálný systém. Simulace by měla být navržena tak, aby podporovala testování jak autonomní navigace, tak jízdy po čáře, a umožnila experimentování s různými typy podvozků.

#### 1.1.4.4 Vývojové prostředí

Na základě požadavků na flexibilitu, autonomní navigaci a možnost simulací byl pro vývoj zvolen open-source framework ROS2. Jak ukázal průzkum, ROS2 nabízí komplexní ekosystém s balíčky, jako je Nav2, které podporují SLAM, AMCL (Adaptive Monte Carlo Localization) a algoritmy pro plánování trajektorie, jako jsou DWA a TEB, což je ideální pro vývoj autonomní navigace a detekce překážek. ROS2 také umožňuje snadnou integraci vlastních algoritmů díky své architektuře založené na komunikaci typu publisher/subscriber prostřednictvím protokolu DDS (Data Distribution Service). Na rozdíl od toho je SIMOVE od Siemens vhodnější pro průmyslové aplikace s minimální potřebou úprav, kde je potřeba rychlá implementace a integrace s existujícími systémy Siemens, avšak jeho uzavřená povaha omezuje možnosti přizpůsobení, což není v souladu s požadavky na vývoj inovativního řešení.

Bezpečnostní požadavky budou zajištěny integrací Siemens PLC s bezpečnostními kartami, protože zvolené senzory RSL400 a RSL200 komunikují přes PROFINET a podporují protokol PROFIsafe, což je v souladu s normami ISO 13849 a IEC 61508, jak bylo vidět u modelů, jako je SSI SCHÄFER WEASEL. Tato kombinace umožní dosáhnout požadované úrovně bezpečnosti (např. SIL 2 nebo SIL 3) a zároveň poskytne možnost vyzkoušet SIMOVE jako alternativní řídicí systém pro srovnání s ROS2, což může být užitečné pro budoucí optimalizaci.

#### 1.1.4.5 Použití jako testovací platforma

Cílem vývoje je nejen vytvořit funkční AGV, ale také získat zkušenosti s autonomními vozidly a otestovat možnosti senzorů od Leuze v průmyslových aplikacích. Z tohoto důvodu budou navrženy menší samostatné aplikace, které mohou být využity v praxi. Jednou z nich je jízda

po čáre, která je stále velmi rozšířená, jak ukazuje příklad SSI SCHÄFER WEASEL, a umožní otestovat senzory na připraveném podvozku v kontrolovaném prostředí. Tato zkušenosť později inspirovala nápad vytvořit systém virtuálních silnic, který by v mapě definoval trasy a pravidla pohybu vozíku bez nutnosti fyzické infrastruktury. Tento přístup by kombinoval výhody autonomní navigace (flexibilitu) s přesností jízdy po čáre, což představuje inovativní řešení pro prostředí, kde je potřeba rychle měnit trasy bez fyzických úprav, a mohl by být konkurenční výhodou oproti modelům, jako je Linde C-Matic, které jsou závislé na QR kódech.

# Kapitola 2

## Hardware

---

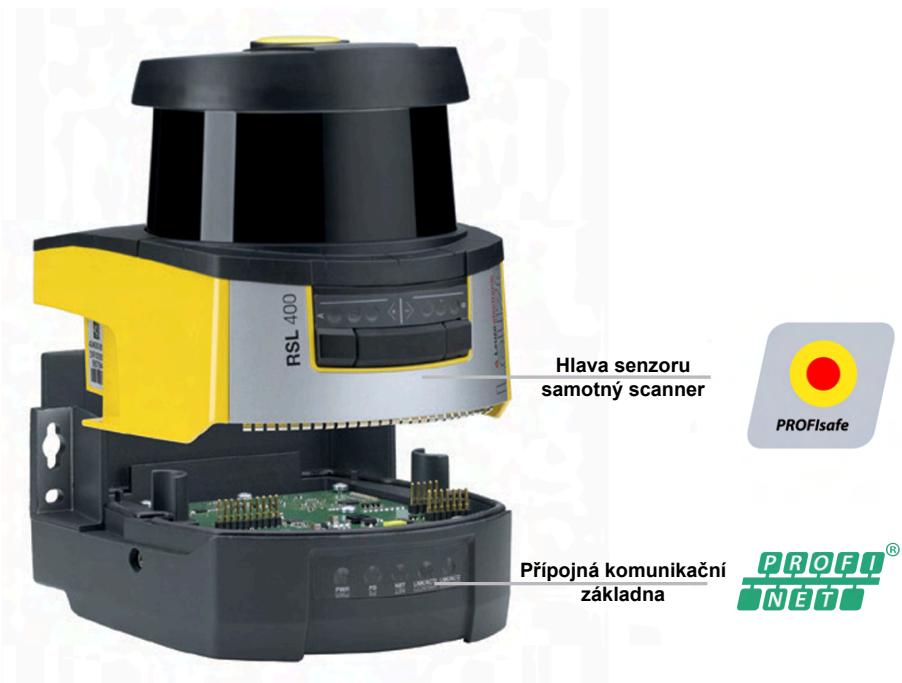
Tato kapitola se věnuje podrobnému popisu hardwarových komponent navržených a využitých v rámci této práce pro řešení navaigačních a bezpečnostních úloh autonomních robotů v průmyslovém prostředí. Hlavní pozornost je zaměřena na čtyři klíčové senzory od firmy Leuze – RSL 400, RSL 200, DCR 248i a OGS 600 – jejichž funkční principy, technické parametry a specifické vlastnosti jsou detailně popsány s důrazem na jejich přínos pro autonomní navigaci a bezpečnost. DCR 248i slouží k detekci 1D a 2D kódů pro přesné řízení pohybu podél předem definovaných tras, zatímco OGS 600 zajišťuje spolehlivé sledování optických vodicích stop. Bezpečnostní laserové skenery RSL 200 a RSL 400 pak poskytují robustní ochranu prostoru a navaigační data, přičemž RSL 200 vyniká jako nejmenší bezpečnostní LIDAR na trhu a RSL 400 nabízí pokročilé možnosti konfigurace bezpečnostních zón. Kromě senzorů kapitola zahrnuje analýzu hardwarového a komunikačního řešení čtyř vytvořených prototypů podvozků, včetně finálního podvozku s mecanum koly, které umožňují všeobecný pohyb a zvyšují manévrovatelnost v omezených prostorách. Popis zahrnuje konfiguraci podvozků, jejich mechanické vlastnosti a integrační aspekty pro řízení v rámci systému ROS2. Předložené informace tak poskytují komplexní přehled o návrhu hardwaru, jeho implementaci a praktickém využití, a mohou sloužit jako inspirace pro vývoj autonomních robotických platform v průmyslových aplikacích.

## 2.1 Použité senzory

### 2.1.1 RSL400

Senzor RSL 400 od firmy Leuze je bezpečnostní laserový skener (LIDAR) navržený pro průmyslové aplikace, splňující požadavky bezpečnostní úrovně SIL2 podle normy IEC 61508 a Performance Level d dle EN ISO 13849-1. Jeho hlavním účelem je monitorování prostoru s dosahem bezpečnostních zón až 8.25 m a celkovým měřicím rozsahem 30 m. Senzor je běžně využíván k ochraně oblastí, kde je třeba zabránit vstupu osob, přičemž narušení zóny spouští bezpečnostní opatření, jako je zastavení strojů. Při vývoji byl kladen důraz na robustnost a odolnost vůči rušivým elementům, například prachu, poletujícím částicím nebo drobným předmětům, což je zajištěno vysokým úhlovým rozlišením 0.1° a pokročilým zpracováním dat v řídicí jednotce.

Senzor se skládá ze dvou hlavních částí: optické měřicí hlavy a oddělitelné komunikační jednotky (viz obrázek 2.1). Tato modulární konstrukce umožnuje výměnu nebo konfiguraci měřicí hlavy bez přerušení komunikace v síti, protože komunikační jednotka zůstává trvale připojena. Měřicí hlava využívá laser třídy 1 (vlnová délka 905 nm), bezpečný pro lidské oko, a skenuje prostor v úhlu 270° s cyklem 40 ms, což zajišťuje rychlou odezvu i v dynamických provozech.



Obrázek 2.1: Schéma sestavy senzoru RSL 400 s měřicí hlavou a komunikační jednotkou [10].

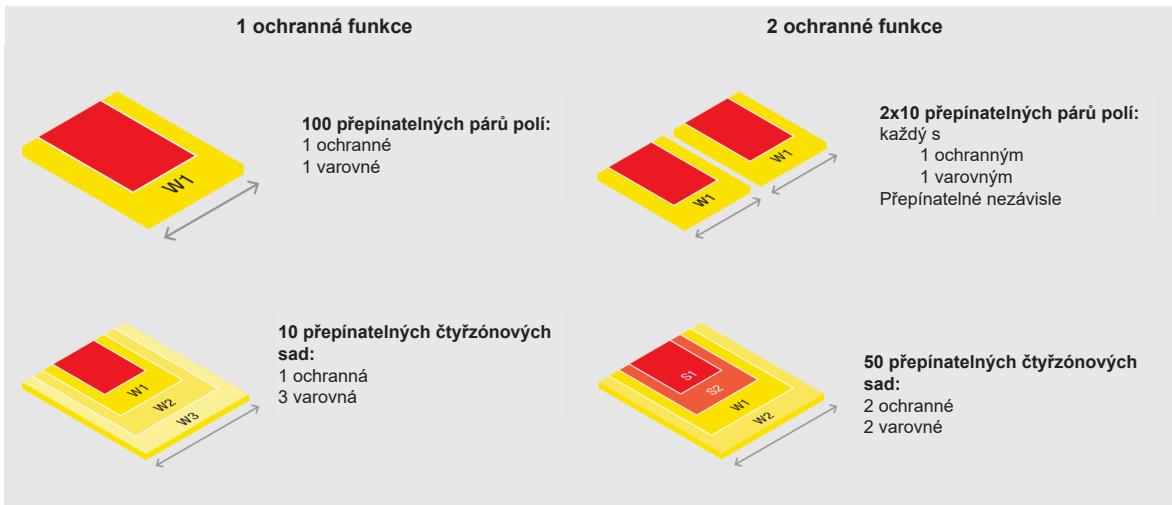
Klíčové technické parametry senzoru, podrobněji popsané v dokumentaci [11], jsou shrnutы v tabulce 2.1 níže.

Jednou z hlavních předností RSL 400 je flexibilita konfigurace bezpečnostních zón. Senzor podporuje až 200 nezávisle nastavitelných polí, která lze dynamicky přepínat v párech nebo sadách po čtyřech, což umožnuje přizpůsobení různým provozním scénářům. Díky dvěma paralelním ochranným funkcím může zařízení současně monitorovat dvě nezávislé zóny, například jedno varovné a jedno ochranné pole, nebo fungovat jako dvě virtuální jednotky v jednom. Ve

Vlastnost	Hodnota
Rozsah měření	30 m
Dosah bezpečnostních zón	8.25 m
Úhel výhledu	270°
Úhlové rozlišení	0.1°
Doba skenovacího cyklu	40 ms
Počet konfigurovatelných polí	až 200
Komunikační rozhraní	Ethernet (UDP), PROFIsafe

Tabulka 2.1: Technické parametry senzoru RSL 400.

verzi s rozhraním PROFIsafe je možné monitorovat až čtyři ochranná pole současně, což je znázorněno na obrázku 2.2. Tato funkce zvyšuje efektivitu bezpečnostního řešení v komplexních prostředích, jako jsou výrobní linky nebo sklady s pohyblivými roboty.



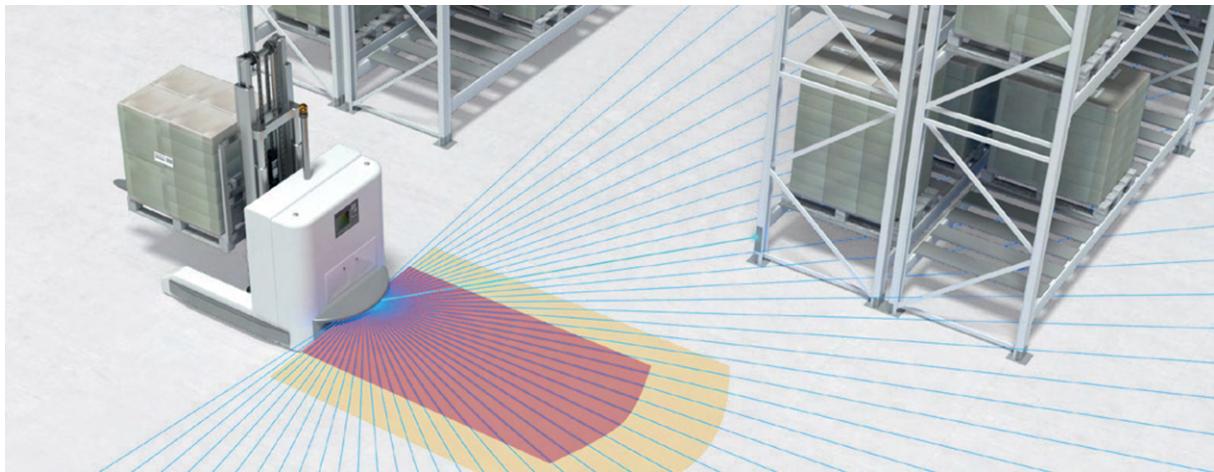
Obrázek 2.2: Konfigurace bezpečnostních zón senzoru RSL 400 s možností monitorování až čtyř polí [10].

Kromě bezpečnostních funkcí poskytuje RSL 400 i navigační data ve formě mračna bodů v cylindrických souřadnicích (vzdálenost a úhel) generovaného s frekvencí 25 Hz. Tato data jsou přenášena přes Ethernet pomocí UDP protokolu a mohou být využita pro lokalizaci a mapování (SLAM) v autonomních navigačních systémech. V této práci byla tato funkcionalita integrována do řídicího systému ROS2, kde sloužila k detekci překážek a podpoře plánování trasy v kombinaci s bezpečnostními zónami. Senzor je vybaven filtrovacími algoritmy, které eliminují falešné detekce způsobené drobnými částicemi (např. prach, hmyz) na základě analýzy velikosti a intenzity odrazu, což zajišťuje vysokou spolehlivost i v prašném prostředí.

Konfigurace senzoru probíhá prostřednictvím softwaru Sensor Studio, který umožňuje de-

finici zón, nastavení filtrů a vizualizaci skenovaných dat v reálném čase. Modulární design a podpora standardních průmyslových protokolů (Ethernet, PROFINET, PROFIsafe) usnadňuje integraci do stávajících systémů. RSL 400 tak představuje univerzální řešení, kombinující bezpečnostní a navigační funkce, což jej činí ideálním pro aplikace v autonomní mobilní robotice, kde je kladen důraz na spolehlivost a flexibilitu v reálných provozech.

Praktické využití bezpečnostních zón senzoru RSL 400 v AGV aplikacích ilustruje obrázek 2.3. Zde je znázorněn autonomní vozík přepravující paletu v prostředí skladu, kde senzor monitoruje prostor před vozidlem. Červená zóna představuje primární ochranné pole, jehož nařušení (např. vstupem osoby) okamžitě spustí zastavení vozíku, zatímco žlutá zóna funguje jako varovné pole, které může iniciovat zpomalení nebo signalizaci. Tato konfigurace zajišťuje bezpečný provoz v dynamickém prostředí s vysokou hustotou překážek, jako jsou regály a palety, a umožňuje efektivní kombinaci bezpečnosti a plynulosti pohybu.



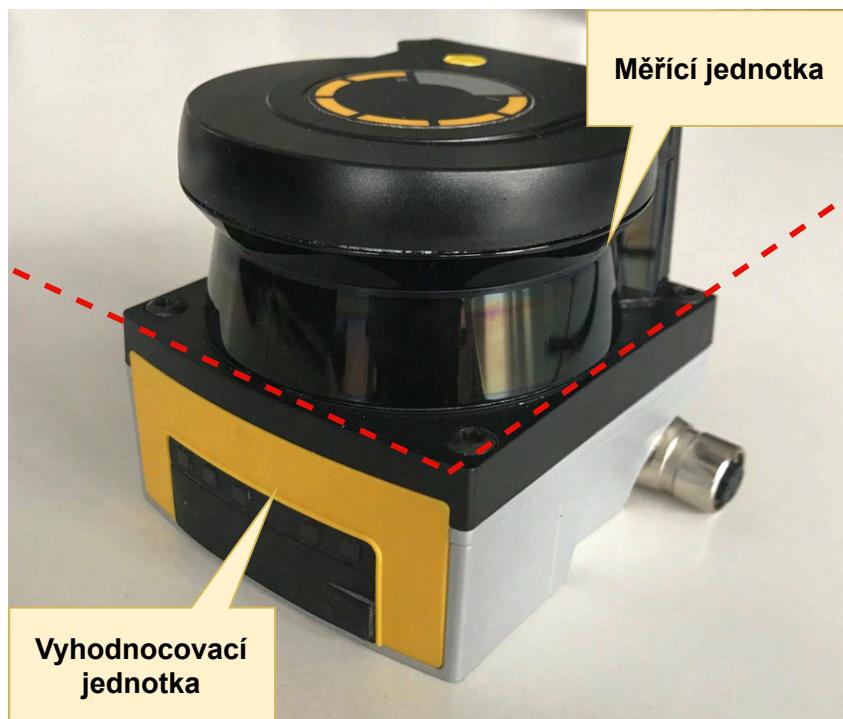
Obrázek 2.3: Znázornění bezpečnostních zón senzoru RSL 400 v AGV aplikaci [11].

## 2.1.2 RSL 200

Senzor RSL 200 od firmy Leuze je ultrakompaktní bezpečnostní laserový skener (LIDAR) navržený pro průmyslové aplikace, splňující požadavky bezpečnostní úrovně Performance Level d (PL d) podle normy EN ISO 13849-1, SIL2 dle IEC 61508 a typ 3 dle IEC 61496-3 [21] (AOPDDR). V současné době (2025) je uváděn na trh a jedná se o nejmenší dostupný bezpečnostní laserový skener, což jej činí ideálním řešením pro aplikace jako jsou autonomní vozidla (AGV), mobilní roboty (AMR) nebo stacionární bezpečnostní systémy. Senzor kombinuje vysokou bezpečnostní spolehlivost s jednoduchou integrací a pokročilými funkcemi, včetně podpory konfigurace přes Bluetooth a výstupu dat pro diagnostiku. Tato sekce popisuje konstrukci senzoru, jeho optický princip, provozní podmínky a komunikační rozhraní z toho důvodu, že funguje trochu jinak, než u LIDARů běžné.

### 2.1.2.1 Stavba senzoru a optický princip

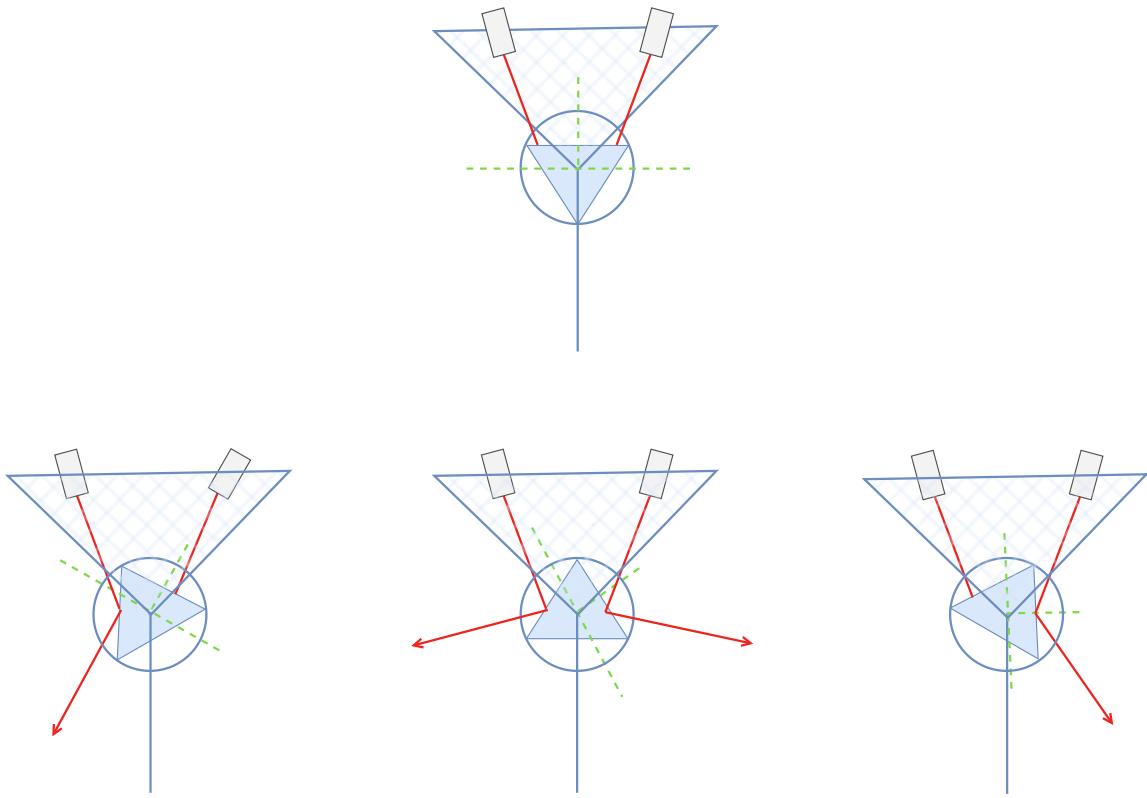
Senzor RSL 200 se skládá z měřicí jednotky a vyhodnocovací jednotky, které společně tvoří integrované zařízení odpovídající normě IEC 61496-3 (AOPDDR typ 3). Měřicí jednotka využívá technologii času letu laseru (Time-of-Flight, ToF) k měření vzdáleností a šířek pulzů v jedné rovině, přičemž data předává do vyhodnocovací jednotky. Vyhodnocovací jednotka zpracovává tato data a zajišťuje bezpečnostní funkce, jako je detekce osob nebo překážek, v požadované reakční době. Schéma sestavy senzoru je znázorněno na obrázku 2.4.



Obrázek 2.4: Schéma sestavy senzoru RSL 200 s měřicí a vyhodnocovací jednotkou.

Optický princip RSL 200 se liší od standardních LIDARů díky unikátní konstrukci rotujícího polygonu. Měřicí jednotka obsahuje dvě laserové hlavy (levou a pravou), které jsou uspořádány kolem rotujícího polygonu s třemi reflexními fazetami ve tvaru rovnostranného trojúhelníku.

níka. Tento polygon zajišťuje skenování v úhlu  $275^\circ$ , což je znázorněno na obrázku 2.5. Použití dvou hlav, které snímají zároveň, umožnilo zrychlit periodu skenu na pouhých  $25ms$ .

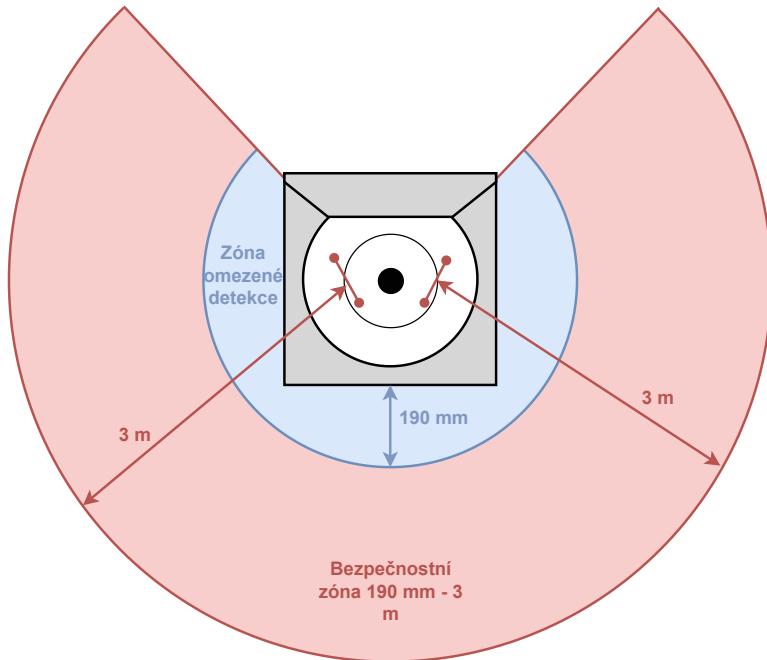


Obrázek 2.5: Princip rozmítání paprsku senzoru RSL 200 s rotujícím polygonem.

Celkový úhel skenování  $275^\circ$  umožňuje efektivní monitorování širokého prostoru, což je výhodné zejména pro AGV a AMR, kde lze pouhými dvěma senzory (umístěnými na vozíku diagonálně) zajistit plné pokrytí  $360^\circ$ . Počátek skenování je uprostřed senzoru při pohledu shora, skenovací rovinu znázorňuje obrázek 2.6 Bezpečnostní dosah sahá od 190 mm do 3.0 m od počátku měření, jak ukazuje obrázek 2.7.



Obrázek 2.6: Skenovací rovina senzoru RSL 200.



Obrázek 2.7: Bezpečnostní zóny senzoru RSL 200 s vyznačením ZLD (modrá) a bezpečnostního rozsahu (červená).

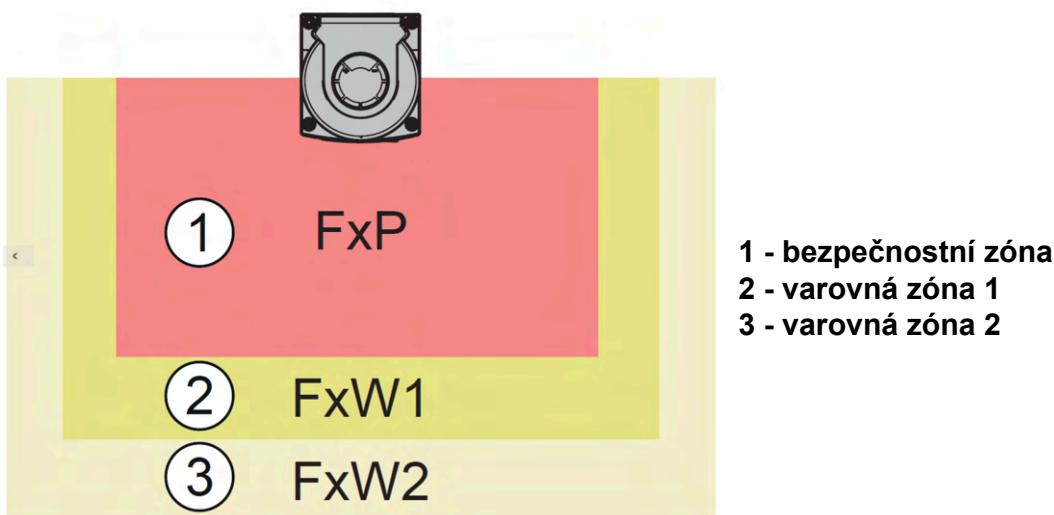
Kompaktní rozměry senzoru ( $80 \text{ mm} \times 86 \text{ mm} \times 80 \text{ mm}$ ) a jeho robustní konstrukce (krytí IP65) umožňují použití v náročných průmyslových prostředích. Senzor je vybaven otočnými M12 konektory, které usnadňují instalaci a minimalizují prostorové nároky na kabeláž. Díky integrované paměti konfigurace lze při výměně zařízení přenést nastavení bez nutnosti nového zarovnání, což snižuje prostoje.

### 2.1.2.2 Nastavení zón

Bezpečnostní senzory RSL 200 umožňují nastavení zón prostřednictvím softwaru *Sensor Studio*. Zóny se skládají z *field triples* (trojice polí: bezpečnostní pole a dvě varovné zóny, viz obrázek 2.8). Konfigurace se liší podle varianty:

- **RSL 210:** 1 *field triple*, 2 *warning fields*, pevná aktivace.
- **RSL 220:** 8 *field triples*, 16 *warning fields*, možnost pevné aktivace nebo přepínání na základě signálů (2, 3 nebo 4 vstupy).
- **RSL 230 a 235:** 32 *field triples*, 64 *warning fields*, pevná aktivace nebo přepínání (2 až 6 vstupů).

Přepínání zón u RSL 2xx umožňuje dynamickou změnu monitorovaných oblastí podle situace, například při změně směru AGV.



Obrázek 2.8: Triplet zón u RSL 200.

#### 2.1.2.3 Využití v praxi

RSL 200 je díky svým kompaktním rozměrům a širokému úhlu skenování ideální pro ochranu AGV a AMR, kde zajišťuje spolehlivé monitorování prostoru a detekci překážek. Jeho schopnost monitorovat skryté oblasti a podporovat bezpečný restart bez manuální kontroly zvyšuje efektivitu provozu. Obecně je prezentovaný a rovněž běžně používaný pro hlídání prostorů kolem strojů. V této práci byl ale senzor integrován do systému ROS2, kde slouží k detekci překážek, hlídání bezpečnostních zón a podpoře navigace díky využití získávání přesných navigačních dat s frekvencí 40Hz.

#### 2.1.2.4 Provozní podmínky

Senzor RSL 200 je navržen pro široké spektrum provozních podmínek, které jsou klíčové pro jeho využití v AGV a stacionárních aplikacích, zejména v náročných průmyslových prostředích, jako jsou sklady nebo výrobní haly. Díky robustní konstrukci s krytím IP65 je senzor odolný vůči prachu a stříkající vodě, což umožňuje jeho spolehlivé použití i v náročných prostředích nebo při teplotních výkyvech v rozmezí 0 °C až 50 °C. Tato odolnost je podpořena testováním na vibrace třídy 5M1 podle normy IEC 61496-1, což zajišťuje funkčnost i při mechanickém namáhání, například při pohybu AGV po nerovném povrchu. Provozní podmínky dále zahrnují schopnost senzoru fungovat při ambientním osvětlení až 3 klux, což je důležité pro prostředí s proměnlivým osvětlením, a v nadmořských výškách až 3000 m, díky čemuž je senzor univerzálně použitelný v různých geografických podmírkách. Tyto podmínky jsou shrnutu v tabulce 2.2 a odpovídají požadavkům na spolehlivost a bezpečnost v průmyslových aplikacích.

Vlastnost	Hodnota	Komentář
<b>Provozní teplota</b>	−50 °C až 60 °C	DIN 40040, Tabelle 10, Kennbuchstabe E (mäßig trocken)
<b>Nepodmíněná teplota</b>	−20 °C až 60 °C	DIN 40040, Tabelle 10, Kennbuchstabe E (mäßig trocken)
<b>Provozní vlhkost</b>	0% až 95%	Nekondenzující
<b>Provozní nadmořská výška</b>	0 m až 3000 m	Dle IEC 60664-1:2007 [22]
<b>Těsnost</b>	IP65 nach IEC/EN 60529	Dle IEC 60529:2013 [23], při správném upevnění krytu
<b>Schránkový tlak</b>	Zink-Druckguss, Kunsstoff	
<b>Rozměry</b>	80 mm × 86 mm × 80 mm	(B × H × T)
<b>Hmotnost</b>	Ca. 0.6 kg	
<b>Vzdálenost středu scanné roviny od spodní hrany pouzdra</b>	60 mm	
<b>Schránková třída</b>	III nach IEC/EN 61140	
<b>Trvalý šok (3 osy)</b>	Nach IEC 60068-2-27: Test 4 – 29, 100 m/s <sup>2</sup> , 16 ms, zůstatek 50 m/s <sup>2</sup> , 11 ms	Dle IEC 60068-2-27, Test 4 – 5, Kategorie 5M1, 50 m/s <sup>2</sup> , 11 ms
<b>Kmitavá zátěž (3 osy)</b>	Nach IEC 60068-2-6: 5 – 200 Hz, 5 g, zůstatek 1 g, 5 Hz	Dle IEC 60068-2-6, Test 4 – 5, Kategorie 5M1

Tabulka 2.2: Provozní podmínky senzoru RSL 200.

### 2.1.3 OGS

Senzor OGS 600 (viz obrázek 2.9) od firmy Leuze je optický systém určený k detekci vodicích stop na podlaze, jehož primárním výstupem je poloha stopy vzhledem k senzoru v milimetrech. Umožňuje identifikaci až šesti paralelních čar v měřicím poli, přičemž pro každou detekuje levou a pravou hranu a vypočítává šířku jako rozdíl těchto hodnot. Senzor je navržen pro rozpoznání světlých čar na tmavém pozadí i tmavých čar na světlém pozadí, přičemž měření probíhá s periodou 10 ms a rozlišením 0.1 mm, jak uvádí technická specifikace [12]. Komunikace je zajištěna přes rozhraní RS232 nebo RS422, přičemž v této práci byla využita varianta RS232 pro přenos polohových dat do řídicího systému.



Obrázek 2.9: Senzor OGS 600 pro detekci optických vodicích stop [12].

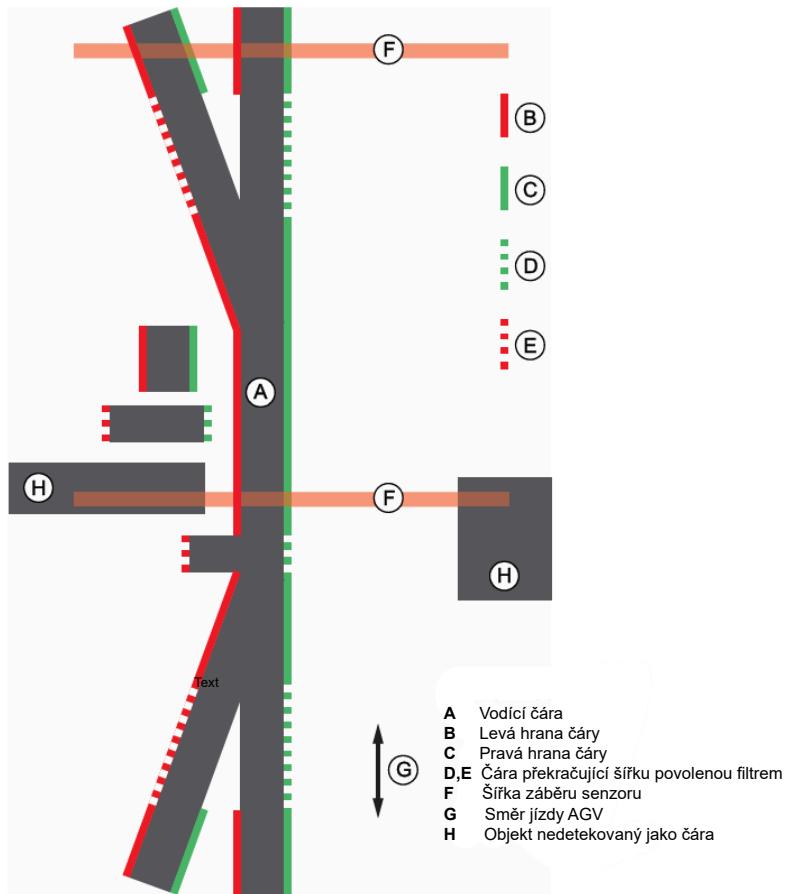
Princip detekce spočívá v analýze amplitudy odraženého červeného světla (vlnová délka 630 nm až 680 nm) emitovaného integrovaným LED zdrojem. Hrany čáry jsou určeny na základě výrazných změn amplitudy způsobených kontrastem mezi čárou a podkladem. Spolehlivá detekce běžně vyžaduje vysoký kontrast, dosažitelný optimální kombinací barev, ale i montážní výškou senzoru v rozmezí 10 mm až 80 mm. Měřicí pole má při této vzdálenosti šířku 160 mm, což umožňuje detekci více stop současně, avšak zvyšuje nároky na konzistenci povrchu. Při testování se ale ukázalo, že i dost špinavá bílá čára na flekatém šedém podkladu je detekována bez problémů a spolehlivě.

Senzor je vybaven pokročilými filtry pro potlačení rušení, jako jsou filtry amplitudového rozsahu, kontrastu a šířky čáry (nastavitelné od 10 mm do 100 mm). Filtr šířky čáry, ilustrovaný na obrázku 2.10, omezuje detekci na stopy odpovídající předem definovanému rozsahu, čímž eliminuje nežádoucí prvky, například skvrny na podlaze. Kalibrace a nastavení parametrů jsou usnadněny poskytovaným grafickým rozhraním, popsaným v dokumentaci [24]. Tento nástroj umožňuje automatickou kalibraci na základě nasnímaných vzorků prostředí a manuální doloďení parametrů, jako je práh detekce kontrastu (nastavitelný v rozmezí 0 až 255 LSB) nebo citlivost na změny osvětlení. GUI dále poskytuje vizualizaci amplitudového profilu v reálném čase, což usnadňuje optimalizaci nastavení v proměnlivých světelných podmínkách, a podporuje ukládání konfigurací pro rychlé přizpůsobení různým scénářům.

Pro zajištění spolehlivé detekce musí vodicí stopa splňovat následující požadavky:

- **Barva čáry:** Červené světlo senzoru ovlivňuje vnímání kontrastu odlišně od lidského oka. Doporučují se barvy s vysokou odrazivostí (např. bílá, RAL 9016) na tmavém podkladu nebo nízkou odrazivostí (např. černá, RAL 9005) na světlém podkladu, s minimálním rozdílem amplitudy 500 LSB. Na druhou stranu senzor si umí poradit i s velmi nízkými kontrasty a lesklými materiály.
- **Šířka čáry:** Optimální šířka čáry je 20 mm až 40 mm, s maximální povolenou hodnotou 100 mm, což vychází z šířky měřicího pole a zajišťuje jednoznačnou identifikaci.
- **Volná oblast kolem čáry:** Minimálně 30 mm na obou stranách čáry musí být bez kontrastních prvků, aby nedošlo k chybné detekci. Je ale otestovaná bezchybná detekce dvou paralelních čar s rozestupem 20 mm.

- **Kombinace barev:** Kontrast musí být dostatečný pro jednoznačné rozlišení hrany; konkrétní doporučení jsou uvedena v tabulce 2.3.



Obrázek 2.10: Fungování filtru šířky čáry pro potlačení rušivých prvků [12].

Ačkoliv by někteří mohli optické senzory pro sledování čáry považovat za triviální a cenově dostupná řešení, OGS 600 představuje průmyslový senzor s robustní konstrukcí a pokročilými funkcemi, které jej předurčují pro reálné provozní podmínky. Vysoká přesnost detekce je zajištěna použitím výkonného LED zdroje, zatímco komunikace je podporována jak přes sériovou linku RS232/422, tak přes sběrnici CANopen, s daty chráněnými kontrolním součtem CRC. Měřicí frekvence 100 Hz (perioda 10 ms) umožňuje rychlou odezvu i při dynamickém pohybu. Senzor poskytuje komplexní sadu výstupních dat, včetně šířky čáry, úhlu natočení, detekce až šesti paralelních stop, jejich vzájemného úhlu, typů křížovatek a informací o amplitudě odraženého světla. Tyto údaje umožňují odvodit charakteristiky čáry, například její barvu, což otevírá možnost rozlišování různě barevných stop pro selektivní navigaci více robotů v jednom prostředí. Nastavitelné filtry potlačují šum způsobený nečistotami nebo drobnými defekty povrchu, přičemž detekce je méně citlivá na změny osvětlení oproti kamerovým systémům, díky čemuž senzor spolehlivě funguje i v prostředí s kolísajícími světelnými podmínkami.

Barva (lidské oko)	Barva podlahy (RAL)	Číslo RAL	Ampli- tuda [LSB]	Vhodná čára
Bílá	Dopravní bílá	9016	21200	Černá
Černá	Hluboká černá	9005	400	Bílá
Červená	Rajčatová červená	3013	11800	Černá
Oranžová	Tmavá oranžová	2011	17400	Černá
Žlutá	Melounová žlutá	1028	19800	Černá
Zelená	Smaragdově zelená	6001	1200	Bílá
Modrá	Ultramarínová modrá	5002	700	Bílá

Tabulka 2.3: Doporučené kombinace barev podkladu a vodicích stop pro senzor OGS 600.

Díky integraci s grafickým rozhraním poskytuje OGS 600 flexibilní řešení pro navigaci podél optických stop. Vizualizace amplitud v reálném čase a možnost přesného nastavení filtrů umožňují přizpůsobení senzoru specifickým podmínkám, jako jsou nečistoty nebo změny osvětlení. V této práci byl senzor využit pro detekci jedné vodicí stopy o šířce 40 mm, což zajistilo stabilní navigaci robota při zachování rychlosti zpracování. OGS 600 tak představuje robustní nástroj pro průmyslové aplikace s velmi přívětivým uživatelským rozhraním, jehož výkon je podmíněn správnou kalibrací a optimalizací prostředí.

## 2.1.4 DCR

Senzor DCR 248i (2.11) od firmy Leuze je inteligentní čtečka 1D a 2D kódů, navržená pro průmyslové aplikace s možností konfigurace přes standardní sběrnice (např. PROFINET, PROFIBUS, Ethernet/IP, EtherCAT) pomocí PLC nebo webového nástroje webConfig. Podporuje široké spektrum komunikačních protokolů, včetně sériové linky RS232, která byla v této práci využita pro přenos dešifrovaného obsahu kódu po jeho úspěšném načtení. Senzor je schopen detektovat následující typy kódů:

- **1D kódy (lineární čárové kódy):** Code 128, EAN 128 (GS1-128), Code 39, Code 2/5 Interleaved, EAN 8 / EAN 13, UPC A/E, Pharmacode, Codabar (Monarch), Code 93.
- **Stacked kódy (vrstvené):** GS1 DataBar (Omnidirectional, Expanded, Limited, Truncated), GS1 DataBar (Stacked Omnidirectional, Stacked Expanded), PDF417.
- **2D kódy (maticové):** DataMatrix (ECC200), Aztec Code, GS1 Aztec Code, GS1 Data-Bar (ECC200), QR-Code, GS1 QR-Code.

DCR 248i umožňuje simultánní detekci více kódů v jednom snímku, což je výhodné pro aplikace, kde robot sleduje čáru s posloupností příkazů nebo kde je vyžadována kontrola čtení. Tato funkce však zvyšuje nároky na zpracování obrazu, což vede k prodloužení doby odezvy. Optimální výkon proto vyžaduje nalezení rovnováhy mezi počtem detekovaných kódů a rychlostí snímkování. Kalibrace senzoru je komplexní vzhledem k vysokému počtu parametrů (např. expozice, kontrast, velikost kódu), které jsou vzájemně závislé a ovlivňují kvalitu detekce.



Obrázek 2.11: Ilustrační fotografie senzoru DCR 248i [13].

Podle technické dokumentace [25] je senzor dostupný v několika optických variantách, lišících se ohniskovou vzdáleností a zorným polem. V této práci byla zvolena varianta s širším zorným polem a střední pracovní vzdáleností (např. model s rozsahem 40 mm až 310 mm dle specifikace), aby bylo zajištěno pokrytí větší plochy kolem virtuální čáry, zejména v zatačkách. Tato konfigurace však klade vyšší nároky na rychlosť snímkování při dynamickém pohybu robota, což bylo částečně kompenzováno větší vzdáleností ohniska senzoru od čáry. Pro zajištění

spolehlivé detekce za různých světelných podmínek senzor podporuje integrované LED osvětlení (červené, 617 nm) s nastavitelnou intenzitou a automatickou adaptací expozice. Přesto bylo v praxi doporučeno spoléhat se na manuální nastavení osvětlení, aby se minimalizovala rizika spojená s kolísáním světla.

Další aspekty konfigurace a použití zahrnují:

1. **Optické varianty:** Různé modely senzoru (např. DCR 248i IMF-08) se liší hloubkou ostrosti a rozlišením ( $752x480px$ , CMOS snímač), což ovlivňuje vzdálenost čtení a velikost detekovatelného kódu (min.  $0.127 mm/modul$ ). Pro tuto aplikaci byla preferována varianta optimalizovaná pro střední vzdálenosti s širším záběrem.
2. **Světelné podmínky:** Automatická adaptace expozice je dostupná, ale pro konzistentní výsledky bylo využito přisvětlení, což zlepšilo kontrast kódů na různých površích.
3. **Omezení typů kódů:** Výběr specifických typů kódů a omezení maximálního počtu detekovaných kódů v jednom snímku (v tomto případě dle požadavků maximálně 3) zkracuje dobu zpracování a zvyšuje efektivitu.
4. **Kontinální snímkování:** Při nepřetržitém čtení (režim Continuous Mode) je nutné omezit snímkovou frekvenci (např. na 10 FPS místo max. 20 FPS), aby procesor senzoru stíhal analýzu obrazu, což je kritické při vyšších rychlostech robota.

DCR 248i tak představuje univerzální řešení pro detekci kódů v navigačních úlohách, jehož výkon je však třeba pečlivě kalibrovat s ohledem na konkrétní provozní podmínky a požadavky aplikace.

## 2.2 Použité podvozky

Původním záměrem bylo vyvinout univerzální platformu, která by umožňovala otestovat možnosti senzorů firmy Leuze pro autonomně naváděná vozidla a pomohla získat zkušenosti s tímto typem úloh. I z toho důvodu jsou připraveny různé modifikace a navrženy různé přístupy, postupně byly přidávány další nápady a řešeny nové úlohy. I v současné době je řešení dále rozvíjeno a modifikováno. Tato variabilita neminula ani samotný podvozek, který byl připraven jak v klasické verzi s diferenciálními koly, tak ve variantě s vše směrovými koly, která umožňuje pohyb v libovolném směru. Již pro úplné začátky, aby bylo možné podvozek smysluplně ovládat, bylo nutné připravit kinematický model pro každou variantu, čímž se podrobněji zabývá kapitola 4.1. Tato část si klade za cíl naopak představit mechanické řešení a možnosti osazení podvozků senzory dle aplikací. Dále bude představena základní infrastruktura komunikace a zapojení.

### 2.2.0.1 Poznámka k vše směrovým kolům - mecanum wheels

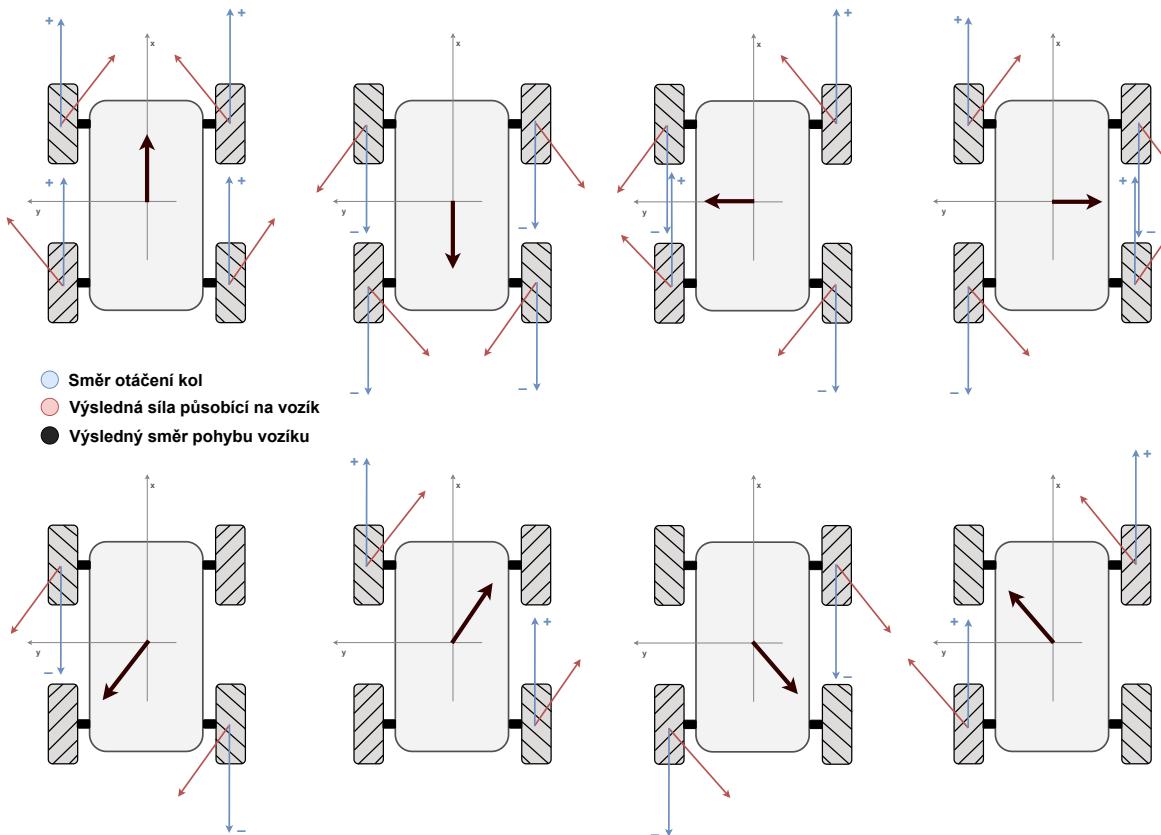
Vše směrová kola, která umožňují pohyb vozidla ve všech směrech bez nutnosti jeho rotace, představují specifické řešení pro zvýšení manévrovatelnosti, jež není v běžných podvozcích samozřejmostí. Ačkoliv se pod tento termín někdy zahrnuje pouze pohyb v základních směrech (dopředu, dozadu, vpravo, vlevo), v této práci je kladen důraz na plnou vše směrovost, tedy schopnost pohybu v libovolném směru roviny. Konkrétně jsou zde využita kola typu *Mecanum Wheels*, jejichž konstrukce je vhodná pro autonomní vozidla (AGV) a aplikace vyžadující přesné ovládání v omezeném prostoru, například ve skladech nebo výrobních halách.



Obrázek 2.12: Detail mecanum kola

Mecanum kolo (viz obrázek 2.12) se skládá z centrálního disku, na jehož obvodu jsou pod úhlem  $45^\circ$  připevněny válcovité rolny. Tyto rolny, obvykle vyrobené z materiálu s vysokou adhezí (např. gumy nebo polyuretanu), umožňují přenos síly nejen ve směru rotace kola, ale i laterálně díky jejich šikmému uložení. Pohyb vozidla je řízen kombinací rychlostí a směrů otáčení jednotlivých kol, což generuje výsledný vektor pohybu v požadovaném směru (viz obrázek 2.13). Tato kinematická vlastnost umožňuje podvozku translaci i rotaci současně, případně jejich nezávislé kombinace, což zvyšuje flexibilitu navigace. Podrobný matematický popis a kinematický model jsou uvedeny v kapitole 4.1.2.

V odborné literatuře je často diskutována optimalizace mecanum kol z hlediska konstrukčních parametrů, jako je počet a tvar rolen, úhel jejich náklonu či materiálové vlastnosti. Tyto faktory ovlivňují účinnost přenosu síly, stabilitu pohybu a odolnost proti opotřebení, což je klíčové pro robustní průmyslové aplikace. V této práci byla zvolena standardní konfigurace s důrazem na kompatibilitu s řídicím systémem a požadavky na nosnost a přesnost pohybu.



Obrázek 2.13: Možnosti pohybu vozíku v závislosti na rychlosti jednotlivých kol

### 2.2.1 DJI Robomaster EP

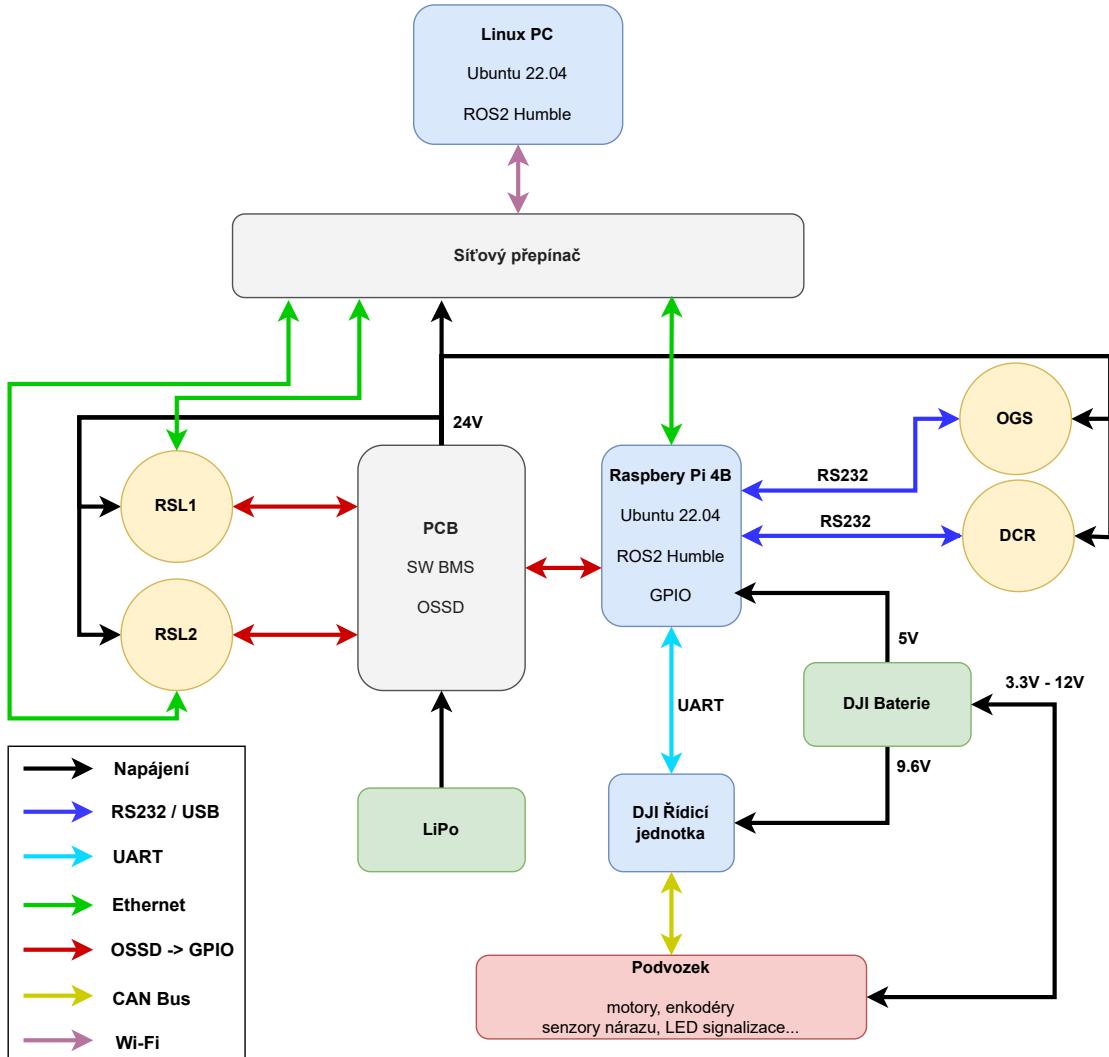
Pro první pokusy na reálném zařízení v libovolné aplikaci (at' již jde o seznámení se s principy navigačního stacku, navržení prototypu funkčního pohybu, detekci palety či základní sledování čáry) byl využit podvozek robota od firmy DJI. Model Robomaster EP (viz obrázek 2.14) umožnuje poměrně velkou míru modularity, má připravené uživatelské Python API [26] a je možné k němu připojit například výpočetní jednotku Raspberry Pi. Disponuje právě výše zmíněnými všesměrovými koly a je to poměrně levný a kvalitní podvozek, na který je možné přistavět vše potřebné.



Obrázek 2.14: DJI RoboMaster EP Core [14]

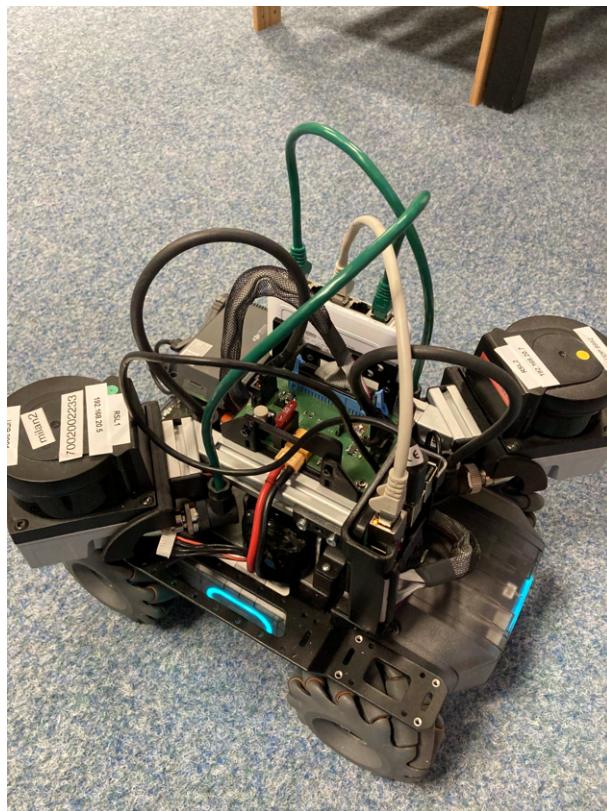
Běh veškerého vlastního SW zajíšťuje jednotka Raspberry Pi 4B, která komunikuje s řídicí jednotkou DJI pomocí zmiňovaného API. Komunikace probíhá pomocí rozhraní RS232. Raspberry Pi je napájena přímo z baterie podvozku, která umožnuje distribuovat napájení přes klasické piny 3.3V pro připojení standardních arduino snímačů a rovněž obsahuje jeden napájecí USB port. K Raspberry Pi je umožněn vzdálený přístup pomocí SSH. Kromě toho je ale využíváno distribuovatelnosti systému ROS2, který umí komunikovat napříč zařízeními (více v kapitole 3.2). Složité výpočty tak mohou být přesunuty na výkonný počítač mimo samotný vozík a Raspberry Pi slouží především pro sbírání a distribuci dat. Dále jsou k podvozku připevněny dva LIDARy typu RSL200 (viz sekce 2.1.2) se zorným polem 270°. Pro ty je připravené externí napájení pomocí LiPo baterie a navržena deska zajišťující napojení napájení se základní ochranou baterie proti podbití a podobnými základními prvky BMS, což byla práce kolegy Jana Holuba [27]. Stejně tak je to se senzory typu OGS a DCR popsaných v kapitolách 2.1.3 a 2.1.4. Tato deska rovněž umožnuje i připojení na OSSD piny senzoru a skrze ně číst informace například o narušení bezpečnostních zón, případně dynamicky měnit jejich velikost. Tuto funkci senzor podporuje, ale v této verzi podvozku nemůžeme mluvit o takzvaném safety řešení. Sen-

zor umožňuje komunikaci pomocí Profinetu, ale zde není ještě žádné PLC, které by zpracovávalo a obsluhovalo ProfiSafe komunikaci. Pro názornost je možno nahlédnout do orientačního schématu na obrázku 2.15.



Obrázek 2.15: Schéma komunikace a propojení funkčních celků prototypu postaveného na DJI podvozku

Jak již bylo řečeno, tento podvozek sloužil pro první konfrontaci navržených systémů s realitou. To je důvodem různých modifikací, jichž se dočkal. První variantou je osazení LIDARy typu RSL200 pro účely autonomní navigace (viz obrázek 2.16). Pomocí nich je zajištěn plný sken okolí vozíku.



Obrázek 2.16: DJI přestavěný pro účely autonomní navigace

Druhou variantou je osazení senzory typu OGS (detekce čáry) a typu DCR (čtení QR kódů). V této variantě (viz obrázek 2.17) musely být LIDARy odstraněny kvůli nedostatku prostoru na podvozku. Tato modifikace slouží pro aplikaci automatické jízdy podle čárového systému s křížovatkami. Kromě toho posloužila pro testování možností zmíněných senzorů jako je například maximální možná rychlosť pohybu pro spolehlivé čtení QR kódu, minimální možný kontrast mezi čárou a okolím pro bezpečnou detekci a tak podobně.

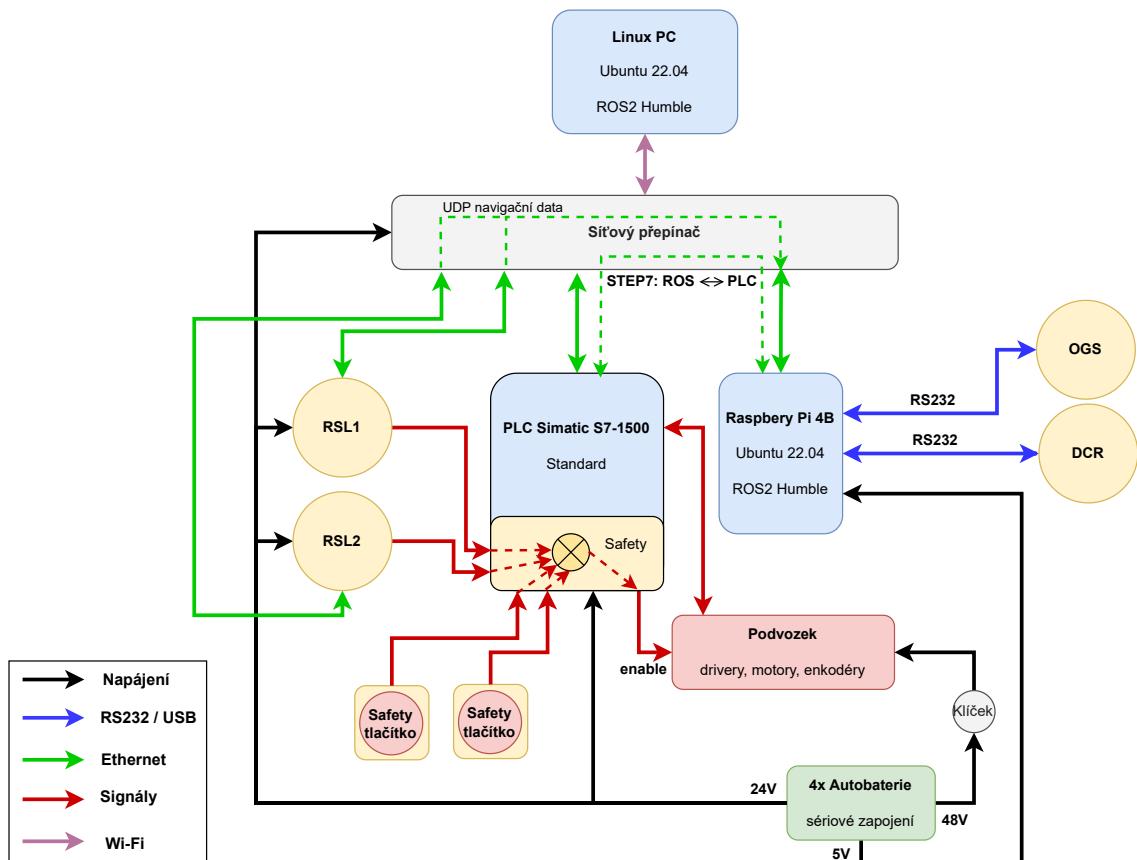


Obrázek 2.17: DJI přestavěný pro účely sledování čáry

## 2.2.2 Diferenciální podvozek

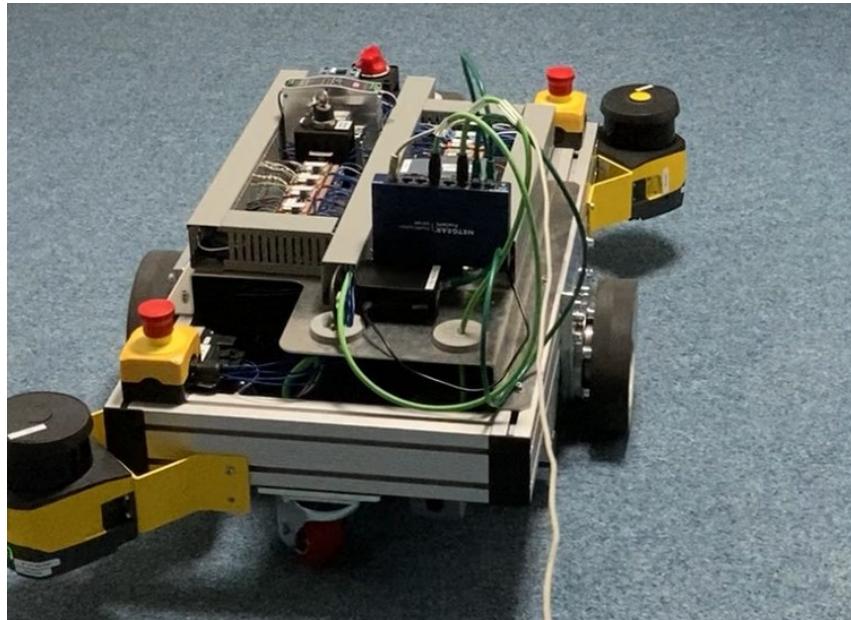
Tento podvozek slouží k pokusům na zařízení splňujícím průmyslové požadavky. Jedná se o další fázi testování algoritmů, které již byly vyladěny na předchozím DJI podvozku (viz sekce 2.2.1). Vozík splňuje bezpečnostní normy pro uvedení do praxe a stejně tak je potřeba upravit SW výbavu. Kromě toho má odlišný typ podvozku, tudíž slouží i k otestování modularity SW a odladění ovládání.

Základní řídicí jednotkou je opět Raspberry Pi 4B, která sbírá data ze senzorů (v případě RSL400 pomocí UDP, v případě DCR a OGS přes RS232). Ta je přes Wi-Fi propojena s výkonějším PC, kde probíhají náročné výpočty. V tomto případě již nejsou povely pro aktuátory posílány přímo z Raspberry Pi, ale tyto požadavky jsou komunikovány do PLC pomocí STEP7, což je aplikační protokol pro SIEMENS PLC využívající TCP komunikační protokol. V PLC jsou obsaženy klasické vstupně-výstupní (I/O) karty, speciální karty pro vyčítání enkodérů (inkrementální rotační čítač) a safety karty, které sbírají data z bezpečnostních STOP tlačítek a informace o narušení bezpečnostních zón z LIDARů. Tato karta teprve pak, když je vše v pořádku, povoluje motory (pouští napětí na *enable* pin driverů). Je zde také implementován přepočet požadovaných rychlostí kol přijatých z Raspberry Pi na napětí, které je potřeba poslat do driverů. Zabezpečení neminulo ani samotné napájení, přičemž hlavním spínačem je spuštěn hlavní okruh, který napájí PLC, Raspberry Pi, senzory, switch apod. Spuštění motorů je však ještě ale ještě zabezpečeno pomocí klíčku. Všechny tyto informace jsou názorně rozkresleny na schématu na obrázku 2.18.



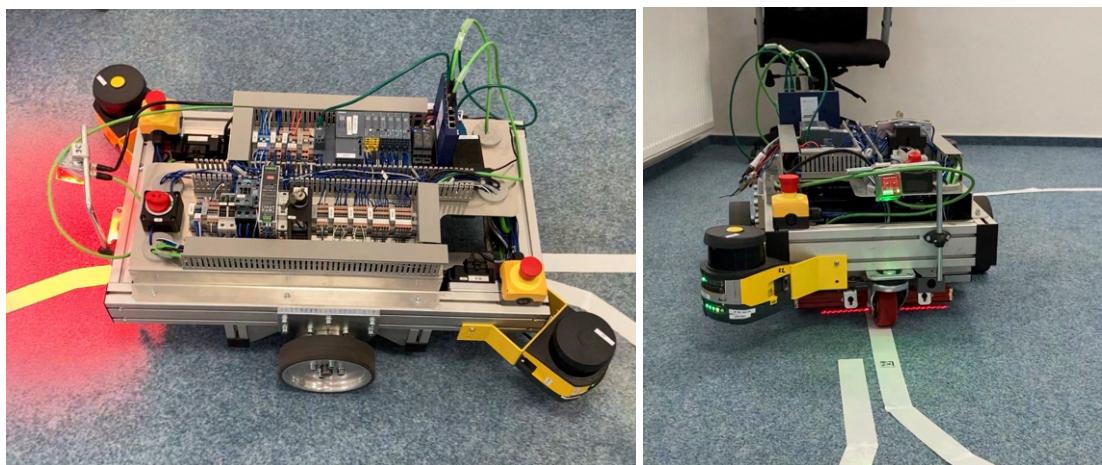
Obrázek 2.18: Schéma komunikace a propojení funkčních celků prototypu postaveného na vlastní diferenciálním podvozku

I v tomto případě jsou navrženy dvě modifikace. První (viz obrázek 2.19) obsahuje pouze LIDARy pro autonomní navigaci. V tomto případě byly primárně řešeny *varovné* a *bezpečnostní* zóny a to mimo veškeré řízení přímo v PLC. Na základě těchto zón se mění maximální povolené rychlosti případně dochází k úplnému odstavení motorů. Je to velmi důležité nejen kvůli splnění bezpečnostních norem, ale i při testování a vývoji. Vzhledem k síle motorů a rozměrům vozíku ( $1.5m \times 0.8m$ ) bylo nezbytné před prvními experimenty vyřešit spolehlivý bezpečnostní systém.



Obrázek 2.19: Diferenciální podvozek přestavěný pro účely autonomní navigace

Druhá varianta (viz obrázek 2.20) slouží opět ke sledování čárového systému. Oproti DJI podvozku je však aplikace plně připravena pro průmyslové použití. Kromě senzorů OGS a DCR, které slouží pro samotný běh algoritmu, jsou stále přítomny i LIDARy. Ty slouží pro bezpečnost. V této aplikaci vůbec není nutné přijímat navigační data, ale jsou zde důležité signály do PLC, které oznamují narušení bezpečnostních zón.

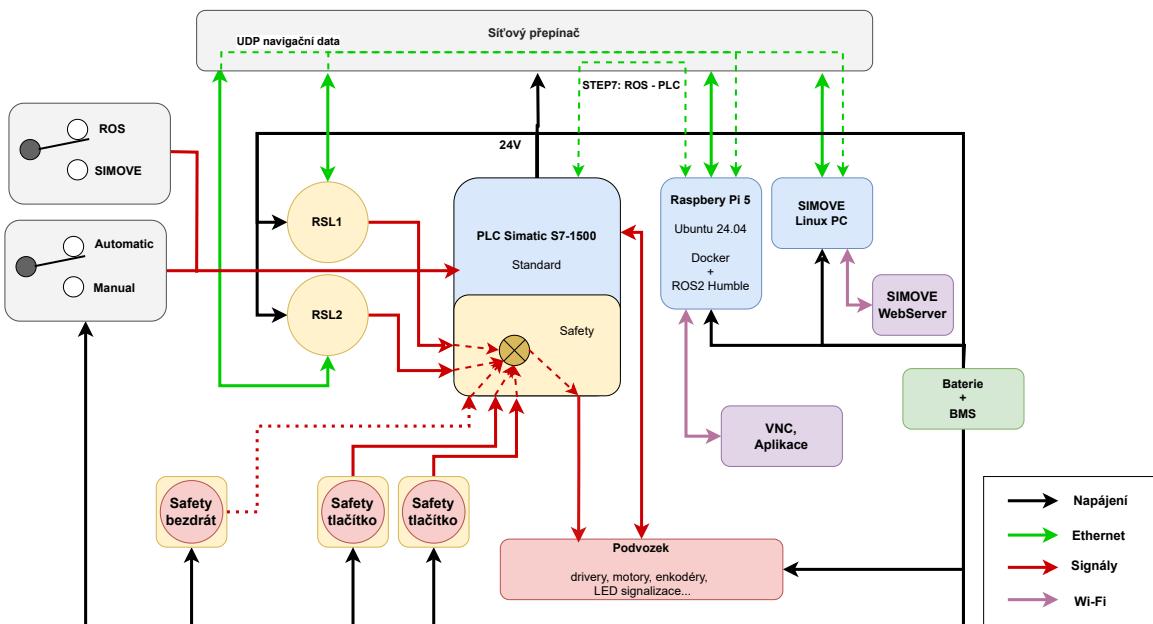


Obrázek 2.20: Diferenciální podvozek přestavěný pro účely sledování čáry

### 2.2.3 Velký všesměrový podvozek

Finálním prototypem je robustní podvozek s všesměrovými mecanum koly, navržený pro přepravu nákladů v průmyslovém prostředí díky integrované zdvihací plošině a souladu s bezpečnostními normami (viz obrázek 2.22). Tento systém vychází ze zkušeností získaných při vývoji předchozích prototypů kombinací autonomní navigace, řízení pohybu všesměrových podvozků a integrací bezpečnostních prvků založených na PLC. Řízení je tedy zajištěno kombinací Siemens PLC a výpočetní platformy s ROS2, což umožňuje splnění požadavků na bezpečnost (Safety Integrity Level, SIL) a efektivní koordinaci pohybu.

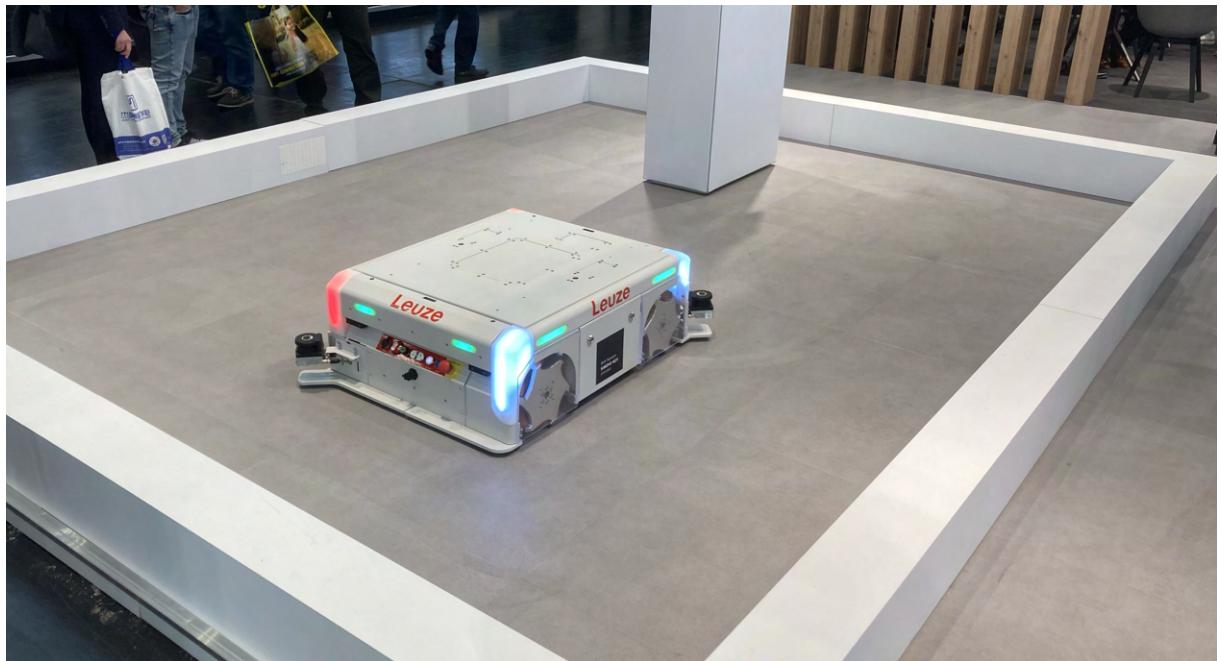
Hardwarová výbava podvozku je charakterizována vysokou kvalitou a spolehlivostí. Pojednané jednotky jsou vybaveny motory s integrovaným rozhraním ProfiSafe, které zajišťují bezpečný a spolehlivý přenos dat z enkodérů i řídicích příkazů mezi motory a PLC. Tato integrace usnadňuje připojení k Siemens PLC a eliminuje riziko ztráty dat při komunikaci. Podvozek disponuje mechanismem pro přepínání mezi manuálním a autonomním režimem v souladu s průmyslovými standardy: přechod do autonomního režimu vyžaduje restart systému a souhlasné nastavení dvou klíčových spínačů, zatímco manuální režim lze aktivovat jedním spínačem. Bezpečnost je dále posílena přítomností nouzových tlačítek, včetně bezdrátového, umožňujícího okamžité zastavení robota v kritických situacích. Světelná signalizace odpovídá normám a indikuje směr pohybu, rotaci, provozní stavu (jízda, připravenost, chyba, nutnost restartu) a přispívá tak k bezpečnému provozu v dynamickém prostředí. Propojení jednotlivých komponent je znázorněno na obrázku 2.21.



Obrázek 2.21: Schéma komunikace a propojení funkčních celků prototypu postaveného na mecanum podvozku

Další klíčovou komponentou je zdvihací plošina, navržená pro manipulaci s paletami, což rozšiřuje využitelnost podvozku v logistických aplikacích. Systém podporuje dva režimy autonomní navigace. Prvním je řídicí systém vyvinutý v ROS2, který tvoří hlavní předmět této práce a zahrnuje pokročilé algoritmy pro plánování a sledování trajektorie. Druhým je komerční systém SIMOVE od Siemensu, poskytující standardizované řešení pro autonomní pohyb. Přepínání mezi těmito systémy je implementováno tak, aby uživatel mohl flexibilně volit mezi vlastním

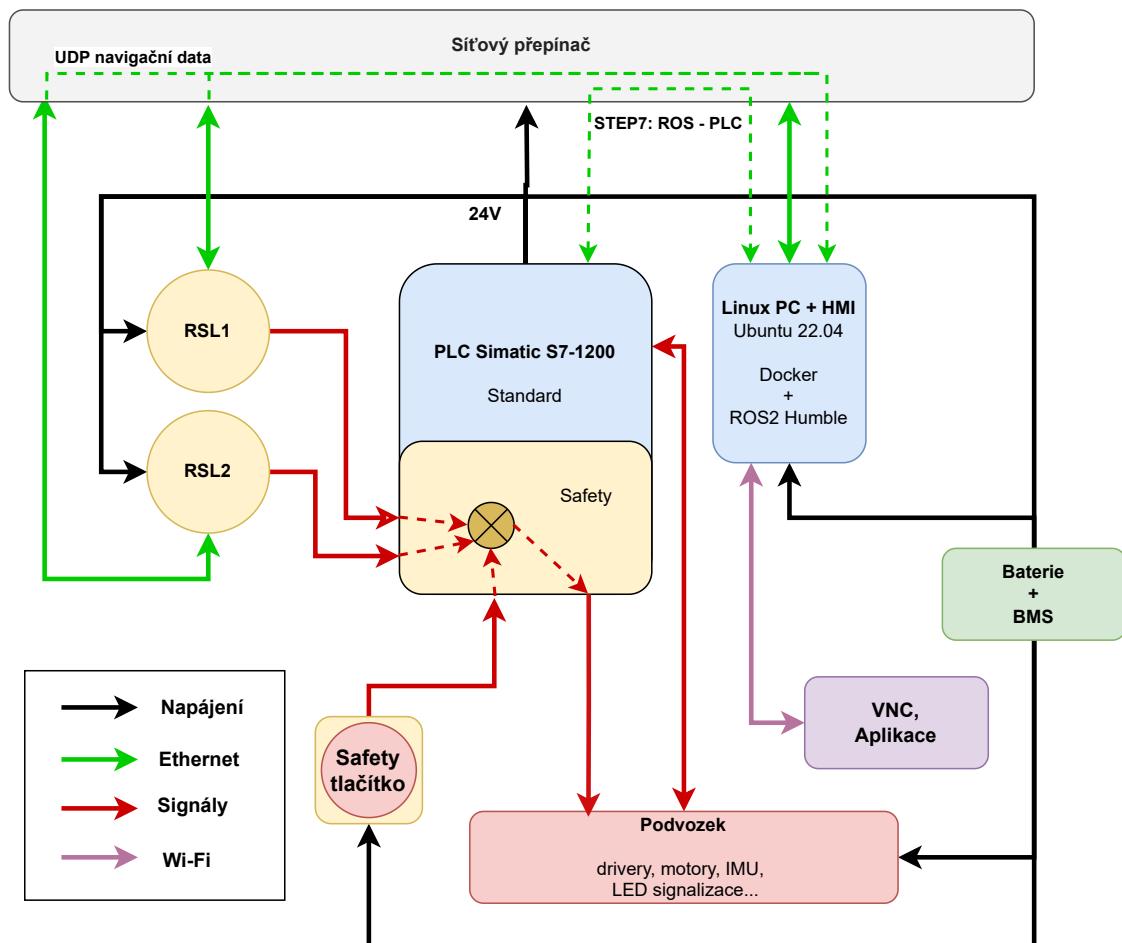
řešením a ověřenou průmyslovou platformou. Díky těmto vlastnostem představuje tento podvozek plně funkční průmyslový stroj, připravený pro nasazení v reálných provozech, jako jsou výrobní haly nebo sklady, s potenciálem dalšího rozvoje v závislosti na specifických požadavcích aplikace.



Obrázek 2.22: Fotografie AGV z veletrhu v Norimberku

## 2.2.4 TinyAGV diferenciální podvozek

Podvozek označený pracovním názvem „TinyAGV“ vychází z původního AGV, které bylo zakoupeno od externího dodavatele pro demonstrační účely senzorů firmy Leuze. Během vývoje navigačních systémů vznikl požadavek na kompletní přestavbu tohoto vozidla za maximálního využití stávajícího hardwaru. Finální vzhled vozíku je možno vidět na obrázcích 2.24 a 2.25. Řízení zajišťuje PLC Siemens SIMATIC S7-1200 v kombinaci s ROS2, nasazeným na integrovaném průmyslovém počítači (IPC) s HMI rozhraním, které umožňuje vizualizaci dat během provozu. Původní rozhraní MSI je nyní omezeno na ovládání světelné signalizace při narušení varovných a bezpečnostních zón, zatímco nově přidané PLC a bezpečnostní scannery rozšířily funkcionality. Vozidlo, založené na diferenciálním podvozku, je navrženo pro cyklickou jízdu po uzavřené trajektorii, což slouží k prezentaci schopností LIDARů na ukázkových akcích.



Obrázek 2.23: Schéma komunikace a propojení funkčních celků prototypu postaveného na malém diferenciálním podvozku

Hardwarová výbava z obrázku 2.23 však vykazuje omezení, která byla kompenzována softwarovými řešeními. PLC S7-1200 vykazuje vyšší latenci při komunikaci s ROS2 oproti výkonnějšímu S7-1500 používanému na jiných platformách, což ovlivňuje přenos dat. Motory podvozku postrádají přesné enkodéry a spoléhají pouze na Hallovy sondy, poskytující nepřesné údaje o rychlosti kol. Tato nepřesnost se projevuje i při přímém translačním pohybu, kdy vy-

čtené rychlosti jednotlivých kol nejsou synchronní, což vede k nespolehlivé odometrii. Tyto nedostatky byly pokryty softwarovou lokalizací, která využívá senzorická data k opravě odometrických chyb, a regulátorem pro korekci rychlostí kol při jízdě po dráze. Tento přístup mě zároveň přivedl k detailní analýze problematiky lokalizace a její parametrizace v podmínkách omezené hardwarové přesnosti.



Obrázek 2.24: Fotografie AGV z veletrhu v Lionu



Obrázek 2.25: Fotografie TinyAGV AGV

# Kapitola 3

## ROS2 - Robot Operating system

---

Tato část práce se soustředí na teoretický popis open-source frameworku ROS2 a jeho využití při vývoji systémů autonomní navigace robotů. Cílem je poskytnout stručný, avšak ucelený úvod do jeho architektury, principů komunikace a základních funkcí, aby čtenář porozuměl struktuře a možnostem tohoto prostředí pro konstrukci navaigačních řešení. Současně jsou představeny klíčové nástroje a knihovny, jako simulační prostředí Gazebo, vizualizační nástroj RViz a navaigační knihovna Nav2. Podrobněji jsou analyzovány vybrané komponenty Nav2, konkrétně algoritmy pro lokalizaci robota v mapě a generování trajektorií, které byly v rámci této práce využity, včetně jejich teoretických principů a odůvodnění jejich vhodnosti pro řešení daných úloh.

### 3.1 ROS2 vs ROS1

Robot Operating System (ROS) je open-source framework navržený pro vývoj a správu řídicích systémů robotů a manipulátorů. Tento rozsáhlý ekosystém zahrnuje širokou paletu softwarových nástrojů, knihoven, algoritmů a ovladačů, které usnadňují návrh, implementaci a testování robotických aplikací. ROS klade důraz na modularitu, autonomii a univerzálnost, díky čemuž není vázán na konkrétní typy robotů ani scénáře, ale poskytuje obecnou infrastrukturu pro základní funkce a komunikaci mezi komponentami. Ačkoli jeho název může evokovat dojem operačního systému, ROS ve skutečnosti není klasickým operačním systémem, nýbrž sadou knihoven a nástrojů postavených na komunikačním mechanismu typu *publisher/subscriber*. Tento přístup zajišťuje efektivní a flexibilní výměnu dat mezi jednotlivými procesy, což umožňuje jeho využití jak v simulačním prostředí, tak na reálném hardwaru.

Aktuální generace ROS, označovaná jako ROS2, představuje významný evoluční krok oproti původní verzi ROS1. ROS2 reaguje na požadavky moderní robotiky, jako jsou vyšší spolehlivost, podpora real-time systémů a distribuce procesů napříč heterogenními platformami. Klíčovou změnou je přechod od centralizované komunikace na robustní middleware založený na standardu Data Distribution Service (DDS), což zvyšuje škálovatelnost a odolnost systému. ROS2 tak nabízí nejen nástroje pro řízení a zpracování dat, ale i podporu více programovacích jazyků (např. C++, Python), což umožňuje vývojářům kombinovat různé komponenty a přizpůsobovat je specifickým potřebám jejich projektů.

Vývoj v ROS2 probíhá v tzv. *workspace*, což je strukturované prostředí, kde jsou projekty organizovány do modulárních balíčků (*packages*). Tyto balíčky zahrnují řídicí algoritmy, ovladače hardwaru, simulační nástroje nebo vizualizační rozhraní, přičemž jejich hierarchická organizace podporuje přehlednost a znovupoužitelnost kódu. Díky této modularitě lze ROS2 využít pro široké spektrum aplikací, od akademických simulací v laboratorním prostředí po nasazení v průmyslových provozech. Tato část se zaměří na detailní popis architektury ROS2, jeho komunikačních mechanismů, integračních možností s nástroji jako Gazebo a RViz a praktických aspektů jeho využití při vývoji robotických systémů.



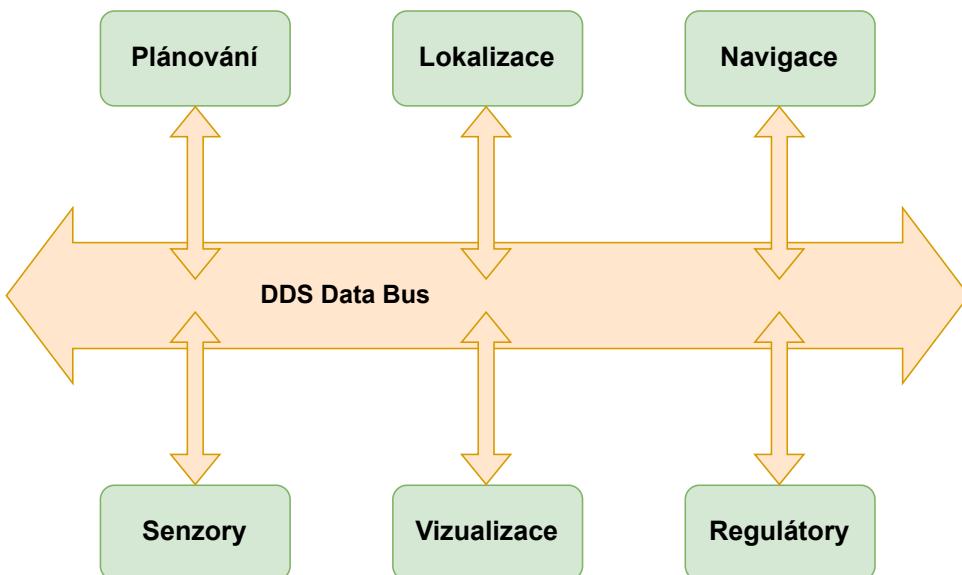
Obrázek 3.1: Ilustrační obrázek ROS2 [15].

## 3.2 Architektura ROS2

Robot Operating System (ROS) byl původně vyvinut v akademickém prostředí jako nástroj pro výzkum a výuku robotických systémů. Díky své flexibilitě a open-source povaze si však postupně získal oblibu nejen mezi akademiky, ale i v průmyslovém sektoru. Druhá generace, ROS2, představuje významný pokrok oproti původní verzi, jelikož přináší vylepšení zaměřená na splnění požadavků moderních robotických aplikací, včetně komerčního a průmyslového využití. Klíčovou inovací je nová komunikační vrstva založená na standardu Data Distribution Service (DDS), který nahrazuje centralizovaný model ROS1 decentralizovanou publikovační architekturou. Tento přístup zajišťuje efektivní, škálovatelnou a robustní komunikaci mezi komponentami systému. ROS2 navíc podporuje více implementací DDS (např. Fast DDS, Cyclone DDS), což zvyšuje jeho přizpůsobivost různým scénářům.

Díky této architektuře se ROS2 hodí pro široké spektrum aplikací, včetně multiagentních systémů, kde desítky až stovky robotů spolupracují v reálném čase. Mezi hlavní oblasti využití patří komplexní robotické systémy, manipulátory a autonomní vozidla, kde je klíčová spolehlivá komunikace a koordinace mezi jednotlivými subsystémy. Decentralizovaný přístup DDS eliminuje závislost na centrálním serveru, což zvyšuje odolnost systému vůči selhání a usnadňuje jeho nasazení v distribuovaných prostředích.

ROS2 disponuje rozsáhlou knihovnou balíčků, která pokrývá základní robotické funkcionality. Mezi příklady patří Nav2 pro navigaci, SLAM Toolbox pro simultánní lokalizaci a mapování, nástroje pro plánování trajektorií, knihovny pro zpracování obrazu (např. OpenCV), vizualizační nástroj RViz (viz sekce 3.9) nebo simulační platforma Gazebo (viz sekce 3.8). Tyto balíčky umožňují vývojářům rychle navrhovat a testovat složité robotické systémy bez nutnosti vytvářet vlastní middleware pro každou komunikační vrstvu.



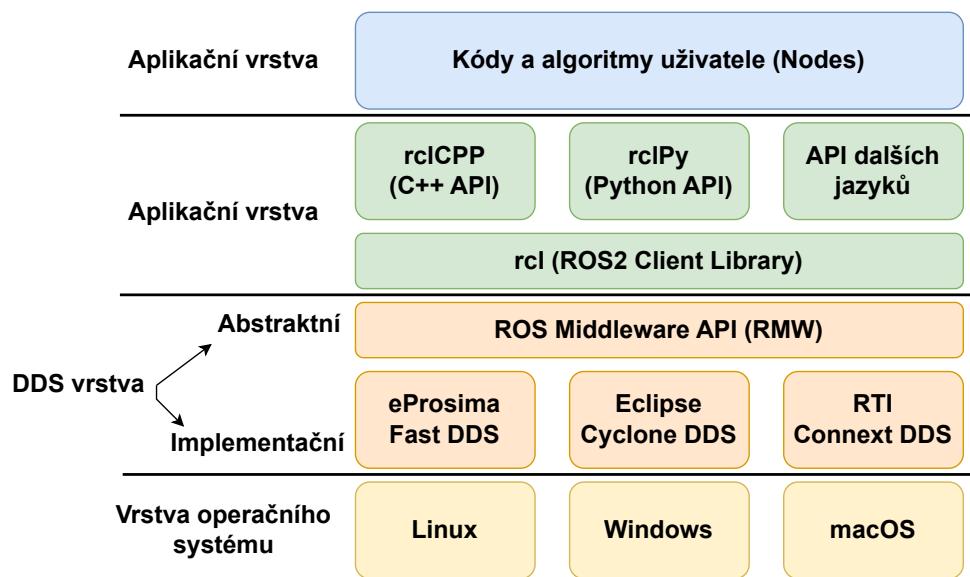
Obrázek 3.2: Schéma komunikace v DDS vrstvě ROS2.

Z architektonického hlediska je ROS2 distribuovaný systém, jehož základními stavebními kameny jsou uzly (*nodes*). Tyto uzly reprezentují samostatné procesy, jako jsou například ovladače nebo algoritmy, a komunikují prostřednictvím DDS middleware. Tento mechanismus umožňuje jejich spolupráci v reálném čase i v heterogenních prostředích, například při řízení autonomních vozidel nebo průmyslových manipulátorů. Schéma komunikace DDS, znázorněné na

obrázku 3.2, lze přirovnat k obousměrné datové „dálnici“, po níž informace proudí mezi uzly efektivně a bez zbytečných zpoždění.

DDS přináší další výhody, jako je vyšší zabezpečení, spolehlivost a optimalizovaný výkon, což z něj činí standardní volbu i v náročných odvětvích, například v obranném průmyslu, kde byl původně vyvinut. Komunikační model je založen na principu *publisher/subscriber*, kde uzly publikují data do sdíleného prostoru a ostatní uzly je mohou odebírat bez nutnosti přímé vazby mezi odesílatelem a příjemcem. Tato decentralizace zvyšuje robustnost a škálovatelnost systému.

Nad implementací DDS stojí abstraktní vrstva RMW (ROS Middleware Interface), která slouží jako rozhraní mezi ROS2 a konkrétními implementacemi DDS (viz obrázek 3.3). RMW umožňuje vývojářům vybrat si z různých DDS knihoven podle specifických požadavků projektu a dokonce kombinovat více implementací v rámci jedné aplikace. Tato flexibilita je neocenitelná při návrhu systémů s různými výkonnostními nebo bezpečnostními nároků.

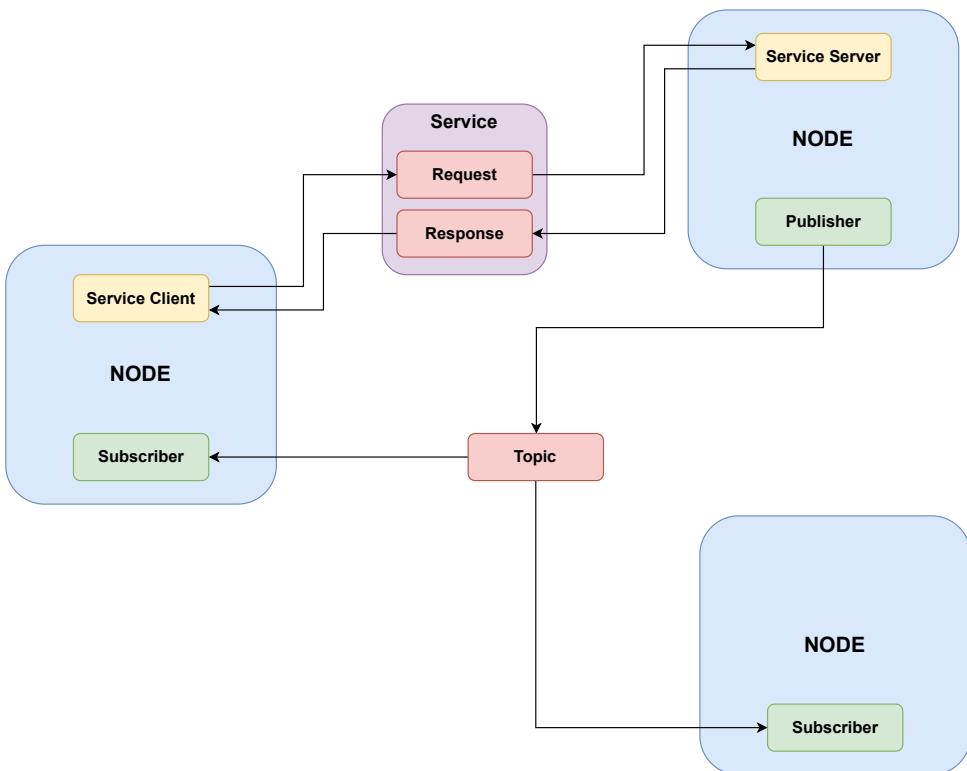


Obrázek 3.3: Layerová architektura systému ROS2 s RMW a DDS.

### 3.3 ROS2 graf a komunikační mechanismy

ROS2 graf je dynamická struktura, která definuje a organizuje komunikaci mezi jednotlivými komponentami systému. Tvoří abstraktní reprezentaci sítě propojených entit, označovaných jako uzly (*nodes*), jež jsou základními stavebními prvky každé ROS2 aplikace. Graf nejen mapuje tyto uzly, ale také způsoby, jakými si vyměňují data. Jeho decentralizovaný design, podpořený DDS middleware, zajišťuje flexibilitu a škálovatelnost, díky čemuž lze ROS2 nasadit v prostředích od jednoduchých simulací až po distribuované sítě zahrnující více robotů nebo cloudové výpočty.

Komunikace v ROS2 grafu probíhá prostřednictvím několika mechanismů, z nichž každý

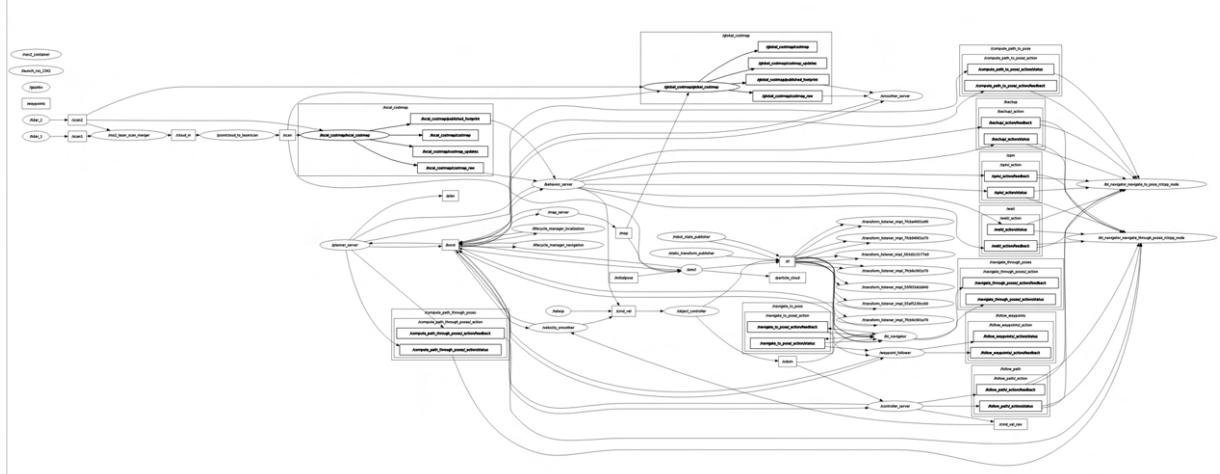


Obrázek 3.4: Princip komunikace mezi komponentami ROS2 grafu.

#### 3.3.1 Uzly a jejich role

Uzly (*nodes*) jsou základními procesy v ROS2, které vykonávají konkrétní úkoly, jako je sběr dat ze senzorů, zpracování informací nebo řízení aktuátorů. Každý uzel je samostatnou entitou, která komunikuje s ostatními uzly prostřednictvím definovaných komunikačních kanálů. Decentralizovaný přístup ROS2 znamená, že uzly fungují nezávisle a nepotřebují centrální řídicí bod, což zvyšuje odolnost systému vůči selhání.

Na obrázku 3.5 je znázorněna vizualizace ROS2 grafu pomocí nástroje *rqt*, který ukazuje propojení uzel v reálné aplikaci. Tento příklad zahrnuje uzly pro ovladač robota, LIDARy a navigaci v předem definované mapě, čímž ilustruje komplexitu a rozsah interakcí v systému.



Obrázek 3.5: Vizualizace ROS2 grafu pomocí nástroje *rqt*.

### 3.3.2 Komunikační mechanismy

ROS2 graf využívá několik typů komunikačních mechanismů, které se liší svým účelem a způsobem fungování. Tyto mechanismy zahrnují téma, služby a akce, přičemž data jsou přenášena ve formě zpráv.

- **Témata (Topics)**

Témata (*topics*) slouží jako komunikační kanály pro nepřetržitý přenos dat mezi uzly na bázi principu *publish-subscribe*. Uzly publikují (*publish*) zprávy do tématu, zatímco jiné uzly se přihlašují k odběru tématu (*subscribe*), aby tato data přijímaly. Tento mechanismus je ideální pro streamování dat, například senzorových měření nebo odometrických informací, kde není potřeba okamžité odpovědi. Každé téma má přesně definovaný typ zprávy, což zajišťuje konzistentnost a kompatibilitu mezi odesílatelem a příjemcem.

- **Služby (Services)**

Služby (*services*) umožňují synchronní komunikaci typu požadavek-odpověď (*request-response*). Uzly mohou volat službu a čekat na odpověď od jiného uzlu, což je užitečné pro operace vyžadující okamžitou interakci, například získání aktuálního stavu systému nebo spuštění specifické funkce. Na rozdíl od témat je komunikace přímá a zaručuje, že požadavek bude zpracován.

- **Akce (Actions)**

Akce (*actions*) jsou určeny pro dlouhodobé úkoly, které mohou být monitorovány, aktualizovány nebo zrušeny během jejich provádění. Tento mechanismus kombinuje výhody témat a služeb: uzel zadá akci (požadavek), přijímá průběžné aktualizace (zpětnou vazbu) a nakonec obdrží výsledek. Akce jsou vhodné například pro plánování trajektorií nebo řízení pohybu manipulátoru, kde je potřeba sledovat průběh operace.

- **Zprávy (Messages)**

Zprávy (*messages*) představují datové jednotky přenášené mezi uzly prostřednictvím výše uvedených mechanismů. Nejsou samostatným prvkem grafu na stejně úrovni jako téma, služby nebo akce, ale slouží jako nositelé informací v těchto komunikačních kanálech. Jedná se v podstatě o datové typy. Zprávy jsou definovány ve formátu `.msg` a obsahují strukturovaná pole s přesně specifikovanými datovými typy, například čísla (`int`, `float`), řetězce (`string`) nebo složitější struktury, jako jsou pole a vnořené zprávy.

Příklad jednoduché zprávy pro popis 3D bodu:

```

1  geometry_msgs/Vector3
2      float64 x
3      float64 y
4      float64 z
5

```

Příklad komplexní zprávy pro LIDARová data:

```

1  sensor_msgs/LaserScan
2      std_msgs/Header header
3      float32 angle_min
4      float32 angle_max
5      float32 angle_increment
6      float32 time_increment
7      float32 scan_time
8      float32 range_min
9      float32 range_max
10     float32[] ranges
11     float32[] intensities
12

```

Zprávy jsou klíčové pro flexibilitu ROS2, protože umožňují uzlům sdílet informace nezávisle na jejich implementaci. Uživatelé mohou definovat vlastní typy zpráv, což usnadňuje přizpůsobení systému specifickým potřebám, například přenosu 3D map nebo obrazových dat.

### 3.3.3 Quality of Service (QoS)

Quality of Service (QoS) je mechanismus, který umožňuje konfigurovat vlastnosti komunikace mezi uzly, aby vyhovovala různým požadavkům aplikací. QoS se nastavuje na úrovni publisherů a subscriberů pro dané téma a ovlivňuje parametry, jako je spolehlivost, latence nebo uchovávání dat. Hlavní parametry zahrnují:

- **Reliability (Spolehlivost):**

- *Reliable*: Zajišťuje doručení zpráv za cenu potenciálního zpoždění, vhodné pro kritická data (např. mapování).
- *Best Effort*: Doručuje zprávy bez záruky, ideální pro rychlá, méně kritická data (např. video stream).

- **Durability (Trvanlivost):**

- *Volatile*: Zprávy nejsou uchovávány pro nové odběratele.
- *Transient Local*: Uchovává poslední zprávy pro nově připojené uzly.

- **History (Historie):**

- *Keep Last*: Uchovává omezený počet zpráv v bufferu.
- *Keep All*: Uchovává všechny zprávy (omezeno pamětí).

- **Deadline, Lifespan, Latency Budget:** Definují časové limity pro doručení a platnost zpráv, což je důležité pro real-time aplikace.

QoS zvyšuje univerzálnost ROS2 tím, že umožňuje optimalizovat komunikaci pro různé scénáře – od nízkolatenčních systémů po robustní přenos v nestabilních sítích. Tento mechanismus je jedním z důvodů, proč je ROS2 vhodný pro komerční a průmyslové nasazení.

## 3.4 Transformace v ROS2

Transformace souřadnic jsou základním matematickým nástrojem v robotice, který umožňuje převádět polohy a orientace mezi různými souřadnicovými systémy. Bez nich by výpočty poloh senzorů, aktuátorů nebo celých robotů vzhledem k prostředí zahrnovaly složité trigonometrické operace, což by bylo zejména ve 3D prostoru neefektivní a náchylné k chybám. V ROS2 jsou transformace implementovány prostřednictvím knihovny *tf2* (Transform Library 2), která zjednodušuje jejich správu a využití v robotických aplikacích.

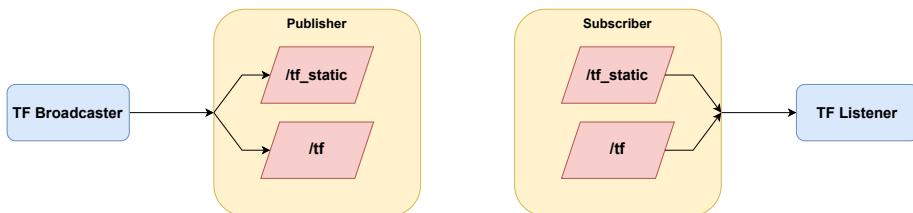
### 3.4.1 Transformační strom

Základním pojmem transformací je *frame* (rámec), což označuje souřadnicový systém přiřazený konkrétnímu objektu, například senzoru, části robota nebo globálnímu prostředí. Transformace mezi rámcemi zahrnují posuny (translace) a rotace, které definují vzájemnou polohu a orientaci těchto systémů. Tyto transformace jsou organizovány do hierarchické struktury nazývané transformační strom (*tf tree*), kde každý rámec má právě jeden nadřazený rámec (*parent*) a může mít více podřízených rámců (*children*). Tato hierarchie usnadňuje modelování složitých systémů s více komponentami.

Knihovna *tf2* v ROS2 umožňuje uzel vysílat (*broadcast*) a přijímat (*listen*) transformace v reálném čase. Transformace jsou propojeny v rámci stromu, což znamená, že data lze převádět mezi libovolnými rámcemi, pokud jsou součástí stejné struktury. Komunikace probíhá přes dva typy témat:

- `/tf`: Pro dynamické transformace, které se mění v čase (např. pohyb kol robota).
- `/tf_static`: Pro statické transformace, které zůstávají konstantní (např. pevná pozice senzoru).

Proces publikování transformací je znázorněn na obrázku 3.6. Aplikace obvykle nepřistupují k témtoto tématu přímo, ale využívají funkce *tf2*, které automaticky zpracovávají data a zajišťují konzistentnost stromu.



Obrázek 3.6: Mechanismus publikování transformací v *tf2*.

Transformační strom nachází uplatnění v následujících scénářích:

- **Transformace senzorových dat:** Převod měření z LIDARu do globálního rámce robota.
- **Navigace:** Určení polohy a orientace robota v mapě.
- **Vizualizace:** Zobrazení relativní polohy komponent v RViz (viz sekce 3.9).

### 3.4.2 Statické a dynamické transformace

Transformace v ROS2 se dělí na dva typy: statické a dynamické, které se liší svou povahou a způsobem využití.

### 3.4.2.1 Statické transformace

Statické transformace jsou neměnné v čase a popisují pevné vztahy mezi rámci, například pozici senzoru vůči tělu robota nebo vzdálenost mezi koly diferenciálního vozíku. Tyto transformace stačí publikovat jednorázově, protože se nemění během provozu systému. K jejich vysílání slouží nástroj `static_transform_publisher`, který definuje translační a rotační parametry mezi nadřazeným (*parent*) a podřízeným (*child*) rámcem. Příklad příkazu pro senzor posunutý o 0.5 m nahoru a natočený o 45°:

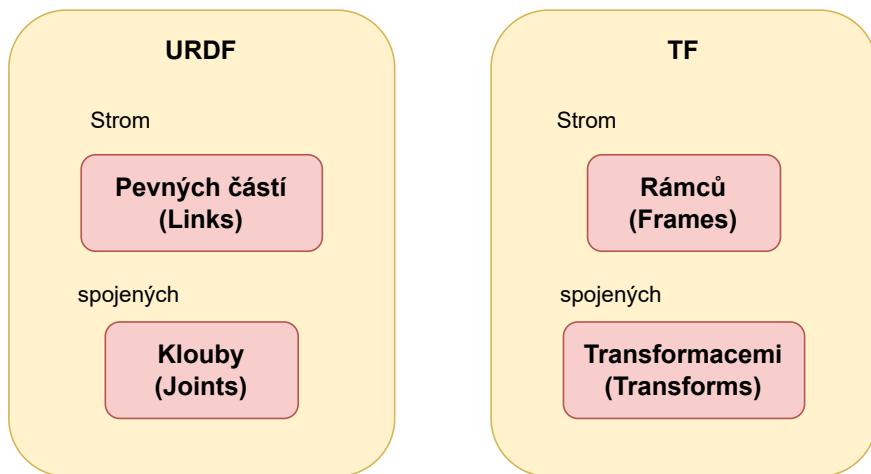
```
ros2 run tf2_ros static_transform_publisher 0 0 0.5 0.785 0 0  
robot sensor
```

Statické transformace jsou uloženy v tématu `/tf_static` a slouží jako „stálé lepidlo“ mezi komponentami, což usnadňuje konfiguraci a testování systému.

### 3.4.2.2 Dynamické transformace

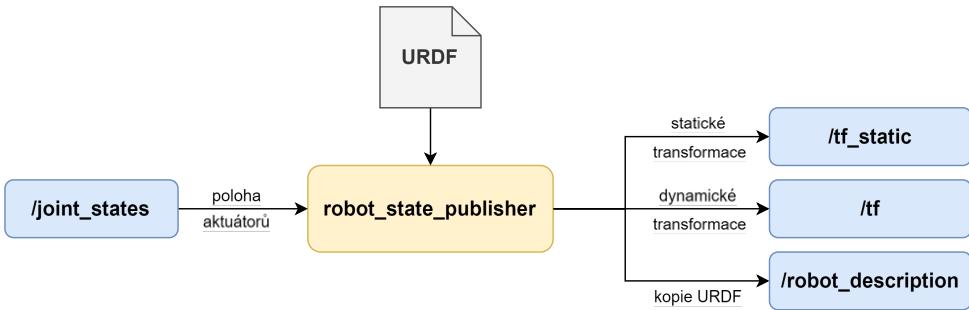
Dynamické transformace se naopak mění v čase a odrážejí pohyb nebo změny v systému, například polohu robota v mapě nebo natočení pohyblivých kloubů. Tyto transformace vyžadují pravidelnou aktualizaci, aby reflektovaly aktuální stav, a jsou publikovány do tématu `/tf`. Pravidelná aktualizace také umožňuje detektovat zastaralá data, což zvyšuje robustnost systému – pokud transformace přestane být aktualizována, systém může signalizovat chybu.

Statické transformace jsou často generovány na základě modelu robota definovaného v URDF souboru (viz sekce 3.8.2.2). URDF popisuje hierarchii komponent (*links*) a kloubů (*joints*), která je podobná transformačnímu stromu (viz obrázek 3.7). Uzel, který je pojmenovaný `robot_state_publisher` převádí URDF na transformace a publikuje je automaticky, včetně obsahu URDF do tématu `/robot_description`.



Obrázek 3.7: Paralela mezi transformačním stromem a URDF strukturou.

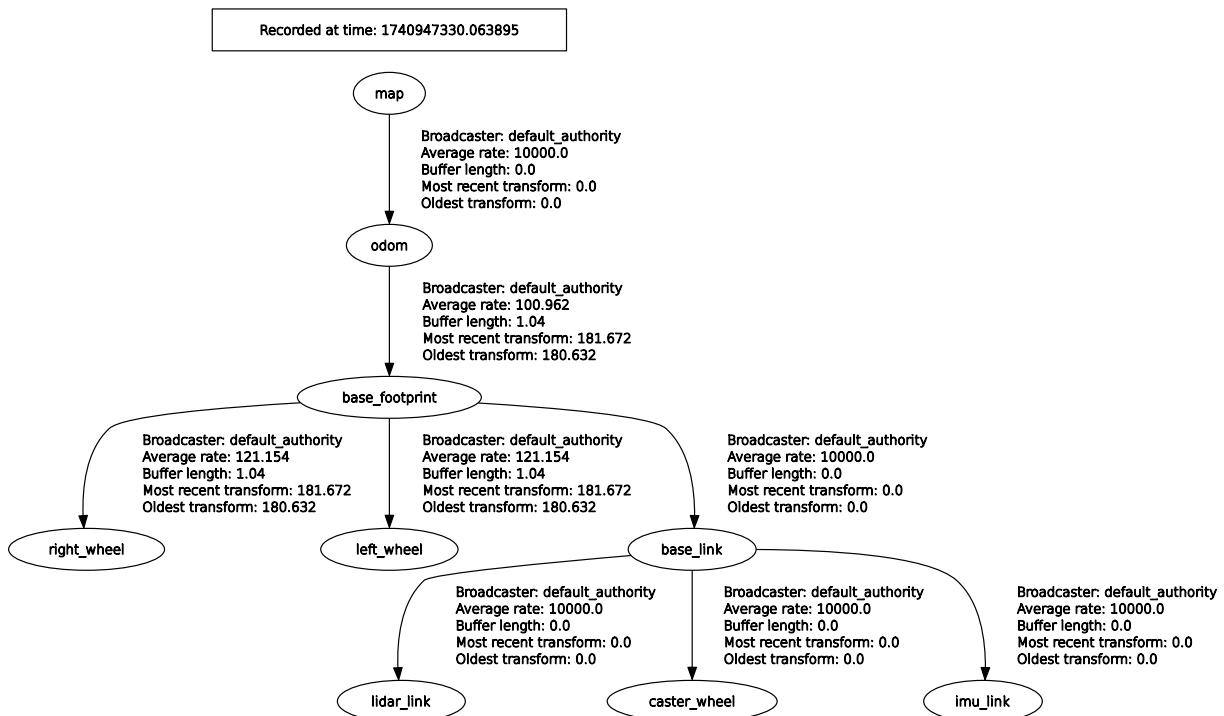
Pro dynamické klouby (např. rotace kol) jsou potřeba aktuální stavy, které jsou poskytovány ve zprávách *JointState* a publikovány do tématu `/joint_states`. Tyto zprávy obsahují informace o poloze, rychlosti nebo námaze kloubu a pocházejí buď ze simulace, nebo z reálných snímačů (enkodéry, gyroskop). Proces zpracování URDF a generování transformací je znázoren na obrázku 3.8.



Obrázek 3.8: Zpracování URDF a publikování dynamických transformací.

Rozdíl mezi statickými a dynamickými transformacemi spočívá v jejich povaze a frekvenci aktualizace. Statické transformace definují konstantní geometrické vztahy a publikují se jednou, zatímco dynamické transformace odrážejí pohyb a vyžadují nepřetržitou aktualizaci. Tato kombinace umožňuje ROS2 efektivně modelovat jak pevné struktury, tak dynamické interakce robota s prostředím.

Na obrázku 3.9 je znázorněn příklad transformačního stromu pro diferenciální vozík s integrovanými senzory a dynamickými klouby na hnacích kolech.

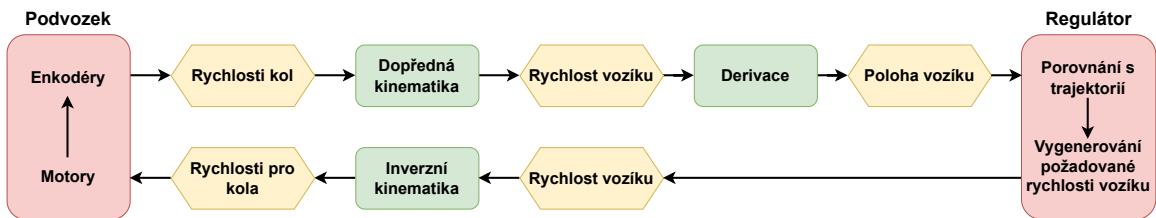


Obrázek 3.9: Znázornění transformačního stromu pomocí nástroje *rqt*.

## 3.5 Lokalizace – odometrie a AMCL

Lokalizace robota v prostředí je zásadní pro jeho autonomní navigaci, protože umožňuje určit polohu a orientaci vzhledem k předem známé mapě nebo dynamicky vytvářenému prostoru. V systému ROS2 existují předpřipravené nástroje, které tuto úlohu zjednoduší a optimalizují. Tato kapitola se zaměřuje na algoritmus **AMCL** (Adaptive Monte Carlo Localization), adaptivní metodu založenou na částicovém filtrování, který kombinuje odometrická data s měřeními ze senzorů (např. LIDAR) k přesnému odhadu polohy robota. AMCL iterativně upravuje odhad polohy a orientace tak, aby odpovídal mapě prostředí na základě porovnání aktuálních senzorických dat s touto mapou.

Výpočet polohy v reálném čase pro celou mapu je výpočetně neúnosný, zejména u rozsáhlých prostředí. AMCL proto omezuje prohledávání na oblast kolem předpokládané pozice robota, která je získána z odometrie a kinematického modelu (viz sekce 4.1). Odometrie, založená na datech z enkodérů kol, však není dostatečně spolehlivá kvůli chybám, jako jsou prokluzy kol, nerovnosti terénu nebo nepřesnosti modelu, což vede k akumulaci odchylek (driftu). Pro korekci těchto chyb se kombinuje s daty z inerciálních měřicích jednotek (IMU) a především s algoritmem AMCL, který zajišťuje robustní lokalizaci v mapě.



Obrázek 3.10: Princip integrace kinematických modelů a odometrie pro navigaci a lokalizaci

Odometrie využívá enkodéry k měření úhlových rychlostí kol, které se převádějí na lineární a úhlové rychlosti robota pomocí kinematického modelu (obr. 3.10). Tyto rychlosti jsou integrovány v čase pro odhad polohy, avšak chyby, např. při prokluzu kol nebo nárazu na překážku, způsobují nesoulad s reálnou pozicí. IMU, obsahující akcelerometry a gyroskop, poskytuje nezávislý odhad pohybu a orientace, čímž částečně kompenzuje tyto odchylky. Klíčovou roli však hraje AMCL, které spojuje odometrii, IMU a senzorická data (např. LIDAR) k eliminaci driftu a zajištění přesné lokalizace v globálním souřadném systému.

### 3.5.1 Částicový filtr

Částicový filtr je statistická metoda založená na Bayesovském odhadu, která approximuje posteriorní pravděpodobnostní rozdělení stavu robota  $p(x_t | z_{1:t}, u_{1:t})$  – tedy pravděpodobnost polohy  $x_t$  v čase  $t$  za podmínky měření  $z_{1:t}$  a řídicích vstupů  $u_{1:t}$  – pomocí diskrétní sady částic. Každá částice  $x_t^{[i]}$  (kde  $i$  je index částice) reprezentuje možný stav robota (např.  $[X, Y, \theta]$ ) a je jí přiřazena váha  $w_t^{[i]}$ , která odráží pravděpodobnost, že tento stav je správný. Tato reprezentace umožňuje zpracování multimodálních distribucí, což je výhoda oproti Kalmanovu filtru, který předpokládá unimodální (Gaussovské) rozdělení.

Proces částicového filtru probíhá v následujících krocích:

1. **Inicializace:** Generuje se počáteční sada  $N$  částic  $\{x_0^{[i]}, w_0^{[i]}\}_{i=1}^N$ , které pokrývají stavový prostor. Pokud je počáteční poloha neznámá, částice jsou rozloženy rovnoměrně po mapě; při známé přibližné poloze (např. ruční inicializace) jsou soustředěny kolem této pozice s počáteční váhou  $w_0^{[i]} = 1/N$ .
2. **Iterace:** Filtr prochází cyklem předpovědi, vážení a převzorkování:
  - **Předpověď:** Stav každé částice je aktualizován podle dynamického modelu robota a řídicích vstupů z odometrie:

$$x_t^{[i]} = f(x_{t-1}^{[i]}, u_t) + \eta_t \quad (3.1)$$

kde  $f$  je kinematický model (např. (4.18)–(4.20) pro diferenciální podvozek),  $u_t$  jsou řídicí vstupy (např. rychlosti kol  $\omega_R, \omega_L$ ) a  $\eta_t \sim \mathcal{N}(0, \Sigma)$  je náhodný šum modelující nejistotu pohybu (např. prokluz).

- **Vážení častic:** Po získání měření  $z_t$  (např. LIDAR sken) je každé částici přiřazena váha podle pravděpodobnosti měření za předpokladu daného stavu:

$$w_t^{[i]} \propto p(z_t | x_t^{[i]}) \quad (3.2)$$

Pravděpodobnost  $p(z_t | x_t^{[i]})$  je vypočtena porovnáním očekávaného skenu z mapy v pozici  $x_t^{[i]}$  s reálným skenem  $z_t$ , často pomocí modelu paprsku (beam model) nebo likelihood field modelu. Váhy jsou poté normalizovány:

$$w_t^{[i]} = \frac{w_t^{[i]}}{\sum_{j=1}^N w_t^{[j]}} \quad (3.3)$$

- **Převzorkování:** Částice jsou převzorkovány podle jejich vah, aby se zabránilo degeneraci (situaci, kdy většina částic má zanedbatelnou váhu). Částice s vyšší váhou jsou duplikovány, zatímco ty s nízkou váhou jsou eliminovány. Proces převzorkování (např. metodou systematického převzorkování) zachovává  $N$  částic a resetuje váhy na  $w_t^{[i]} = 1/N$ .

Odhad polohy robota je poté vypočten jako vážený průměr častic:

$$\hat{x}_t = \sum_{i=1}^N w_t^{[i]} x_t^{[i]} \quad (3.4)$$

Počet častic  $N$  je klíčovým parametrem: nízký počet vede k nedostatečné reprezentaci posterioru, zatímco vysoký počet zvyšuje výpočetní náročnost. Podrobnosti o částicových filtroch lze nalézt v článku Sebastiana Thruna [28].

### 3.5.2 Adaptivní částicový filtr

Adaptivní částicový filtr (APF) rozšiřuje základní částicový filtr o mechanismy pro dynamickou úpravu počtu částic a jejich rozložení, čímž optimalizuje přesnost a výpočetní efektivitu. Fixní počet částic může být problematický: příliš málo částic způsobuje špatnou aproximaci multimodální posteriorní hustoty pravděpodobnosti, zatímco příliš mnoho částic zvyšuje zátěž procesoru, což je nevhodné pro reálný čas. APF řeší tyto nedostatky přizpůsobením podle složitosti posteriorního rozdělení a kvality měření.

Klíčové vlastnosti APF zahrnují:

- Minimalizace chyby reprezentace:** APF minimalizuje chybu aproximace posteriorní hustoty pravděpodobnosti  $p(x_t | z_{1:t}, u_{1:t})$  konečnou sadou částic. Tato chyba je často kvantifikována Kullback-Leiblerovou divergencí (KL divergence) mezi skutečným rozdělením a jeho částicovou aproximací:

$$D_{KL}(p\| \hat{p}) = \int p(x_t) \log \left( \frac{p(x_t)}{\hat{p}(x_t)} \right) dx_t \quad (3.5)$$

kde  $\hat{p}(x_t)$  je aproximace pomocí částic. Počet částic  $N$  je upraven tak, aby  $D_{KL} < \epsilon$ , kde  $\epsilon$  je požadovaná mez chyby:

$$N \propto \frac{1}{\epsilon} \quad (3.6)$$

- Adaptivní inicializace:** Počáteční počet částic je nastaven podle nejistoty odometrie a měření. Např. při nízké kvalitě měření (vysoký šum) se generuje více částic.
- Dynamická úprava počtu částic:** APF analyzuje entropii posterioru nebo efektivní počet částic  $N_{\text{eff}}$ :

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^N (w_t^{[i]})^2} \quad (3.7)$$

Pokud  $N_{\text{eff}}$  klesne pod práh (např. kvůli degeneraci), přidají se částice v oblastech s vysokou nejistotou; při stabilní unimodální distribuci se počet částic redukuje.

- Převzorkování:** APF provádí převzorkování inteligentněji, např. zachovává diverzitu částic přidáním náhodných vzorků, aby se zabránilo ztrátě multimodálních hypotéz.

APF tak vyvažuje přesnost a výpočetní náročnost, což je ideální pro lokalizaci v dynamických prostředích. Detaily lze nalézt v [29].

### 3.5.3 Použití adaptivního částicového filtru pro lokalizaci

AMCL v ROS2 využívá APF pro lokalizaci s těmito přístupy:

1. **Ruční inicializace:** Poloha robota je nastavena uživatelem (např.  $[X_0, Y_0, \theta_0]$ ), což zrychluje konvergenci a eliminuje nejistotu v symetrických prostorech.
2. **Předpověď pohybu:** Při pohybu robota jsou částice aktualizovány podle odometrie (rovnice (3.1)), kde  $f$  je kinematický model a  $\eta_t$  modeluje šum odometrie.
3. **Vážení pomocí LIDARu:** Měření  $z_t$  z LIDARu váží částice podle shody s mapou (rovnice (3.2)). Např. pro likelihood field model:

$$p(z_t | x_t^{[i]}) = \prod_{k=1}^M \exp\left(-\frac{d_k^2}{2\sigma^2}\right) \quad (3.8)$$

kde  $d_k$  je vzdálenost  $k$ -tého paprsku od nejbližší překážky v mapě a  $\sigma$  je šum měření.

4. **Přidání náhodných částic:** AMCL pravidelně přidává náhodné částice rovnoměrně po mapě, aby zabránilo ztrátě polohy při špatném odhadu (např. kvůli symetrii mapy).
5. **Konvergence:** Filtr konverguje po několika iteracích (např. 25 skenů za sekundu při frekvenci LIDARu 40 ms), kdy částice s nízkou váhou jsou eliminovány a odhad se stabilizuje.

AMCL je efektivní v členitých, nesymetrických prostorech, kde unikátní rysy prostředí, jako jsou rohy, stěny nebo specifické překážky, usnadňují konvergenci částicového filtru k jednoznačnému odhadu polohy. V těchto podmínkách rychle eliminuje méně pravděpodobné hypotézy díky jednoznačné shodě mezi LIDAR skeny a mapou. V otevřených nebo symetrických prostorech, například dlouhých chodbách nebo pravidelných mřížkových skladech, však může vyžadovat více iterací k rozlišení multimodálního posteriorního rozdělení, protože podobnost měření z různých pozic zvyšuje nejistotu. V takových případech může být nutný ruční zásah (např. přesná inicializace polohy) nebo zvýšení počtu částic a frekvence přidávání náhodných vzorků (parametry `recovery_alpha_slow/fast`), aby filtr nezůstal uvězněn v nesprávném odhadu.

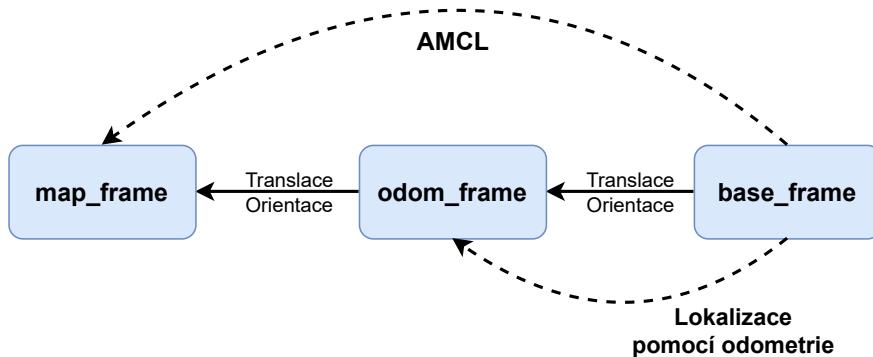
Srovnání AMCL s jinými metodami lokalizace, jako je Kalmanův filtr a mřížková lokalizace, je podrobně popsáno v literatuře [30]. Kalmanův filtr, založený na Gaussovské approximaci posteriorního rozdělení, je výpočetně efektivní a přesný v prostorech s unimodální distribucí (např. dobře strukturované prostředí s minimem symetrií), avšak selhává při multimodálních scénářích, jako jsou symetrické prostory, kde AMCL díky částicovému přístupu lépe zvládá více hypotéz. Mřížková lokalizace, která reprezentuje stavový prostor pevnou diskrétní mřížkou, poskytuje vysokou přesnost a zvládá symetrické prostory za předpokladu dostatečně husté mřížky, ale její výpočetní náročnost roste exponenciálně s rozsahem mapy, což ji činí méně praktickou pro výpočty v reálném čase oproti AMCL. V symetrických prostorech může být mřížková lokalizace teoreticky výhodnější díky systematickému pokrytí prostoru, zatímco AMCL závisí na kvalitě inicializace a adaptivní úpravě částic. Nicméně AMCL zůstává preferovanou volbou v ROS2 díky své flexibilitě a schopnosti kombinovat odometrii s LIDAR daty, což umožňuje robustní lokalizaci i v dynamických podmínkách za rozumnou výpočetní cenu.

### 3.5.4 Koncepce balíčku AMCL

Balíček AMCL je integrován do systému ROS2 jako součást knihovny Navigation2 [31] a slouží k lokalizaci robota v předem známé mapě prostředí. AMCL implementuje adaptivní částicový filtr (APF) popsaný v podkapitolách 3.5.1 a 3.5.3, který udržuje pravděpodobnostní rozdelení polohy robota  $p(x_t | z_{1:t}, u_{1:t})$  na základě odometrických dat a měření ze senzorů, typicky LIDARů. Tento odhad je průběžně aktualizován kombinací předpovědi pohybu z odometrie a vážení částic podle senzorických dat, což umožňuje korigovat akumulované chyby odometrie (drift) a zajistit přesnou polohu v globálním rámci mapy.

AMCL vyžaduje předdefinovanou statickou mapu prostředí (např. ve formátu `occupancy_grid`), kterou porovnává s aktuálními měřeními. Mapa je obvykle vytvořena předem pomocí SLAM algoritmů a obsahuje informace o překážkách a volném prostoru. AMCL přijímá data z odometrie (pohybová předpověď) a LIDARu (senzorická aktualizace), přičemž alternativně lze použít hloubkové kamery (např. Intel RealSense, Microsoft Kinect), jejichž data jsou převedena na formát `sensor_msgs/LaserScan` pomocí balíčku `depthimage_to_laserscan`. Tento proces zajišťuje kompatibilitu s širokou škálou senzorů a zvyšuje flexibilitu balíčku.

V ROS2 AMCL funguje jako uzel (node), který zpracovává vstupy a publikuje transformace mezi souřadnými systémy. Odometrie, vypočítaná z kinematického modelu a enkodérů (viz sekce 4.1), určuje relativní pohyb robota a publikuje transformaci z `base_frame` (tělo robota) do `odom_frame` (odometrický systém). Tato transformace však driftuje kvůli chybám v měření pohybu. AMCL tento drift koriguje porovnáním LIDAR skenů s mapou a publikuje transformaci z `odom_frame` do `map_frame` (globální systém mapy). Výsledná poloha robota v globálním rámci je tedy kombinací odometrického odhadu a korekce z AMCL, jak je znázorněno na obrázku 3.11.



Obrázek 3.11: Princip lokalizace pomocí AMCL v ROS2: Odometrie publikuje transformaci `base_frame` → `odom_frame`, AMCL koriguje drift a publikuje `odom_frame` → `map_frame`.

Fungování balíčku jako takového můžeme stručně shrnout poznatky z předchozích podkapitol:

1. **Inicializace:** Uživatel zadá přibližnou počáteční polohu ( $X_0, Y_0, \theta_0$ ) nebo AMCL inicializuje částice rovnoměrně po mapě. Počet částic je určen parametry `min_particles` (např. 100) a `max_particles` (např. 5000), což definuje rozsah adaptivity filtrování.
2. **Předpověď pohybu:** Při pohybu robota jsou částice aktualizovány podle odometrického

modelu:

$$x_t^{[i]} = f(x_{t-1}^{[i]}, u_t) + \eta_t \quad (3.9)$$

kde  $f$  je kinematický model (např. (4.18)–(4.20) pro diferenciální podvozek),  $u_t$  jsou rychlosti kol z enkodérů a  $\eta_t$  je šum modelovaný jako  $\mathcal{N}(0, \Sigma)$  s kovariancí nastavenou parametry `odom_alpha1`–`odom_alpha4` (např. rotace, translace).

3. **Vážení částic:** Po obdržení LIDAR skenu  $z_t$  jsou částice zváženy podle pravděpodobnosti měření:

$$w_t^{[i]} = p(z_t | x_t^{[i]}) \quad (3.10)$$

AMCL podporuje několik modelů měření, např. *likelihood field model*, kde:

$$p(z_t | x_t^{[i]}) = \prod_{k=1}^M \exp\left(-\frac{d_k^2}{2\sigma^2}\right) \quad (3.11)$$

$d_k$  je vzdálenost  $k$ -tého paprsku od nejbližší překážky v mapě v pozici  $x_t^{[i]}$ ,  $\sigma$  je šum měření (parametr `laser_z_hit`) a  $M$  je počet paprsků. Váhy jsou normalizovány ( $\sum w_t^{[i]} = 1$ ).

4. **Převzorkování a adaptivita:** AMCL dynamicky upravuje počet částic podle efektivního počtu částic  $N_{\text{eff}} = 1 / \sum(w_t^{[i]})^2$ . Pokud  $N_{\text{eff}}$  klesne pod práh (např. 50 % z celkového počtu `max_particles`), provede se převzorkování a přidají se náhodné částice (`recovery_alpha_slow/fast`) pro obnovu v případě ztráty polohy.
5. **Výstup:** AMCL publikuje odhad polohy jako transformaci `odom_frame` → `map_frame` ve formátu `geometry_msgs/PoseWithCovarianceStamped`, který zahrnuje  $X$ ,  $Y$ ,  $\theta$  a kovarianci nejistoty.

AMCL je vysoce konfigurovatelný, což umožňuje přizpůsobení různým robotickým platformám. Klíčové kategorie parametrů zahrnují:

- **Parametry filtru:** `min_particles` (např. 100) a `max_particles` (např. 5000) určují rozsah počtu částic, `resample_interval` (např. 2) frekvenci převzorkování a `recovery_alpha_slow/fast` (např. 0.001/0.1) rychlosť přidávání náhodných částic.
- **Laserový model:** `laser_likelihood_max_dist` (např. 2 m) definuje maximální vzdálenost pro shodu s mapou, `laser_z_hit` (např. 0.95) váhu správných měření a `laser_z_rand` (např. 0.05) váhu náhodných šumů.
- **Odometrický model:** `odom_alpha1`–`odom_alpha4` (např. 0.2) modelují šum rotace a translace, např.  $\alpha_1$  pro rotační šum z rotace,  $\alpha_3$  pro translační šum z translace.

Při správném nastavení parametrů (např. zvýšení `max_particles` v symetrických prostorzech nebo úprava `odom_alpha` pro prokluzující kola) AMCL dosahuje vysoké přesnosti a minimalizuje zbytečné korekce pohybu, jako jsou časté otočky robota. Například v prostředí s minimem překážek může být nutné zvýšit `laser_likelihood_max_dist`, aby se zabránilo příliš rychlé eliminaci částic.

## 3.6 Plánování trajektorie

Plánování trajektorie je základní problém autonomní navigace mobilních robotů, jehož cílem je určit bezpečnou a efektivní cestu z výchozí polohy do cílové s ohledem na prostředí, kinematická omezení robota a dynamické faktory, jako jsou pohyblivé překážky. V ROS2 je tento proces realizován v navaigacním stacku Navigation2 (*Nav2*), který integruje globální a lokální plánování. Globální plánování generuje celkovou trasu na základě statické mapy, zatímco lokální plánování upravuje pohyb v reálném čase podle aktuálních senzorických dat. Tato kapitola definuje problém plánování, odvozuje jeho řešení v mřížkovém prostoru a detailně popisuje globální plánovač NavFnPlanner, s rozšířením o lokální plánování pomocí MPPI (Model Predictive Path Integral) regulátoru.

Problém plánování trajektorie lze formulovat jako optimalizaci cesty v grafu  $G = (V, E)$ , kde  $V$  je množina uzlů, např. bodů v 2D mřížce reprezentující prostředí a  $E$  je množina hran, spojujících sousední uzly (např. v 4- nebo 8-sousednosti). Každá hrana  $e(u, v) \in E$  má váhu  $w(u, v)$ , která odráží náklad pohybu (např. vzdálenost, riziko kolize).

Cílem je najít posloupnost uzlů  $p = \langle s, v_1, \dots, g \rangle$  od startu  $s$  do cíle  $g$ , minimalizující celkový náklad:

$$\text{cost}(p) = \sum_{i=1}^k w(v_{i-1}, v_i) \quad (3.12)$$

Prostředí je obvykle reprezentováno jako *occupancy grid* – 2D mřížka, kde každá buňka má hodnotu nákladu odrážející přítomnost překážek nebo volný prostor.

### 3.6.1 Dijkstrův algoritmus jako teoretický základ

Dijkstrův algoritmus je klasická metoda pro hledání nejkratší cesty v grafu s nenegativními váhami hran, která slouží jako teoretický základ pro pochopení globálního plánování v robotice. Uvádíme jej zde, protože ilustruje princip prohledávání grafu a minimalizace nákladů, což je koncepcioně podobné metodám v ROS2, i když NavFnPlanner využívá pokročilejší techniky. Níže je podrobněji odvozen jeho postup.

Představme si graf  $G = (V, E)$ , kde  $V$  je množina uzlů (např. buňky mřížky) a  $E$  jsou hrany mezi sousedními uzly. Každá hrana  $(u, v)$  má váhu  $w(u, v) \geq 0$ , např. Euklidovskou vzdálenost mezi buňkami. Algoritmus hledá nejkratší cestu od startovního uzlu  $s$  ke všem ostatním uzlům, včetně cíle  $g$ , a pracuje takto:

#### 1. Inicializace:

- Pro každý uzel  $v \in V$  nastavíme vzdálenost od startu  $d(v) = \infty$ , kromě  $d(s) = 0$ .
- Předchůdce  $\pi(v) = \text{null}$  sleduje cestu zpět.
- Množina prozkoumaných uzlů  $S = \emptyset$ , neprozkoumaných uzlů  $V \setminus S = V$ .

#### 2. Výběr uzlu:

Z neprozkoumaných uzlů  $V \setminus S$  vybereme uzel  $u$  s minimální vzdáleností  $d(u)$ , tedy  $u = \arg \min_{v \in V \setminus S} d(v)$ . Tento krok znamená, že hledáme uzel, který je aktuálně „nejblížší“ startu podle dosavadních odhadů vzdálenosti. Například po inicializaci je  $u = s$ , protože  $d(s) = 0$  a ostatní  $d(v) = \infty$ .

### 3. Relaxace hran:

- Pro všechny sousedy  $v$  uzlu  $u$  aktualizujeme vzdálenost:

$$d(v) = \min\{d(v), d(u) + w(u, v)\} \quad (3.13)$$

- Pokud je nová vzdálenost  $d(u) + w(u, v) < d(v)$ , nastavíme  $\pi(v) = u$ . Relaxace zajišťuje, že pokud existuje kratší cesta přes  $u$ , je zohledněna.

4. **Opakování:** Přidáme  $u$  do  $S$  a opakujeme kroky 2–3, dokud  $S \neq V$  nebo dokud není  $g$  prozkoumán.
5. **Ukončení:** Po ukončení je  $d(g) = \delta(s, g)$  nejkratší vzdálenost od  $s$  do  $g$ , a cesta je rekonstruována zpětně z  $\pi(g)$  až k  $s$ :

$$p = \langle s, \pi(\pi(g)), \pi(g), g \rangle \quad (3.14)$$

Celkový náklad cesty je:

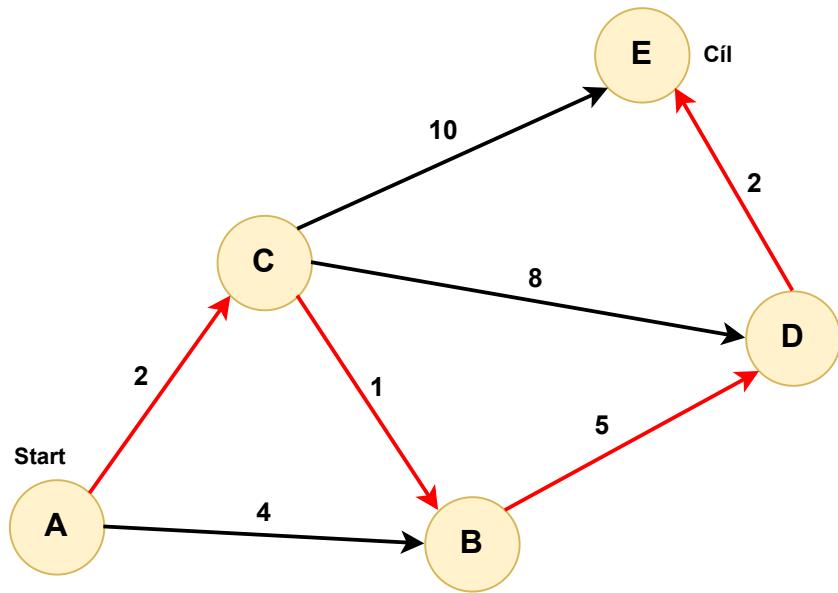
$$\delta(s, g) = \sum_{i=1}^k w(v_{i-1}, v_i) \quad (3.15)$$

#### 3.6.1.1 Příklad

Pro graf s uzly A, B, C, D, E a váhami hran  $A \rightarrow B : 4$ ,  $A \rightarrow C : 2$ ,  $B \rightarrow C : 1$ ,  $B \rightarrow D : 5$ ,  $C \rightarrow D : 8$ ,  $C \rightarrow E : 10$ ,  $D \rightarrow E : 2$  hledáme cestu od A do E (viz obrázek 3.12). Postup je následující:

1. Inicializace:  $d(A) = 0$ ,  $d(B) = d(C) = d(D) = d(E) = \infty$ ,  $S = \emptyset$ .
2. Krok 1: Vybereme A ( $d(A) = 0$ ), relaxujeme:  $d(B) = 4$ ,  $\pi(B) = A$ ;  $d(C) = 2$ ,  $\pi(C) = A$ ;  $S = \{A\}$ .
3. Krok 2: Vybereme C ( $d(C) = 2$ ), relaxujeme:  $d(B) = \min(4, 2 + 1) = 3$ ,  $\pi(B) = C$ ;  $d(D) = 8$ ,  $\pi(D) = C$ ;  $d(E) = 10$ ,  $\pi(E) = C$ ;  $S = \{A, C\}$ .
4. Krok 3: Vybereme B ( $d(B) = 3$ ), relaxujeme:  $d(D) = \min(8, 3+5) = 8$ ;  $S = \{A, C, B\}$ .
5. Krok 4: Vybereme D ( $d(D) = 8$ ), relaxujeme:  $d(E) = \min(10, 8 + 2) = 10$ ;  $S = \{A, C, B, D\}$ .
6. Krok 5: Vybereme E ( $d(E) = 10$ ), končíme.

Nejkratší cesta je  $A \rightarrow C \rightarrow B \rightarrow D \rightarrow E$  s délkou  $2 + 1 + 5 + 2 = 10$ .



Obrázek 3.12: Příklad Dijkstrova algoritmu

Dijkstrův algoritmus je optimální díky trojúhelníkové nerovnosti ( $w(u, v) \leq w(u, x) + w(x, v)$ ), ale jeho složitost  $O(|V|^2)$  (nebo  $O(|E| + |V| \log |V|)$  s prioritní frontou) je neefektivní pro velké mřížky [32]. Proto NavFnPlanner využívá pokročilejší přístup založený na potenciálových polích.

## 3.7 Prohledávání grafů a hledání nejkratší cesty

Prohledávání grafů představuje základní koncept v informatice a robotice, který se využívá k řešení problémů navigace, plánování cest a optimalizace v komplexních prostředích. V kontextu autonomních systémů, jako jsou mobilní roboty nebo vozíky, slouží prohledávání grafů k identifikaci nejfektivnějších cest mezi počáteční a cílovou pozicí v prostoru reprezentovaném grafem. Tato kapitola se zaměřuje na teoretické základy prohledávání grafů, algoritmy pro průchod strukturou grafu a podrobný popis A\* algoritmu pro hledání nejkratší cesty, s ohledem na jeho aplikaci v robotické navigaci, například pro jízdu po virtuálních silnicích definovaných v mapě generované LIDARem (viz kontext kapitoly 5.4.1).

### 3.7.1 Základní principy prohledávání grafů

Graf je matematická struktura složená z vrcholů (uzlů) a hran (spojů mezi uzly), která modeluje vztahy mezi prvky v prostoru. V robotice je graf často reprezentací prostředí, kde vrcholy představují diskrétní body (např. polohy robota) a hrany reprezentují možnosti pohybu mezi těmito body s přidruženými náklady (např. vzdálenost nebo čas). Prohledávání grafů zahrnuje systematický průchod touto strukturou za účelem nalezení cesty, dosažení cíle nebo optimalizace určitého kritéria, jako je minimální vzdálenost nebo čas.

Nejčastěji používanými strategemi prohledávání grafů jsou:

- **Prohledávání do šírky (Breadth-First Search, BFS):** Systematicky prochází všechny vrcholy na stejně úrovni před pokračováním do hlubších vrstev. BFS garantuje nalezení nejkratší cesty v neohodnoceném grafu (když jsou všechny hrany stejně nákladné), ale může být výpočetně náročný pro velké grafy.
- **Prohledávání do hloubky (Depth-First Search, DFS):** Prochází jeden vrchol až do jeho konce před návratem a pokračováním další větví. Tento přístup je efektivní z hlediska paměti, ale nemusí vždy najít nejkratší cestu a může uvíznout v nekonečných cyklech, pokud není graf acyklický.

Oba algoritmy jsou jednoduché a efektivní pro malé grafy, ale pro reálné robotické aplikace, jako je navigace po virtuálních silnicích, jsou omezené kvůli absenci ohodnocení hran a neschopnosti optimalizovat cestu podle nákladů.

### 3.7.2 Hledání nejkratší cesty pomocí A\* algoritmu

A\* (A-star) je heuristický algoritmus prohledávání grafů, který kombinuje informace o nákladech cesty doposud (od počátečního vrcholu) s odhadovanou vzdáleností k cíli (heuristikou). Tento přístup umožňuje efektivní nalezení nejkratší cesty v ohodnoceném grafu.

#### 3.7.2.1 Základní princip A\*

A\* algoritmus prochází graf pomocí prioritní fronty, kde je každý vrchol ohodnocen funkcí:

$$f(n) = g(n) + h(n),$$

kde:

- $g(n)$  je aktuální náklad cesty od počátečního vrcholu  $s$  k aktuálnímu vrcholu  $n$ ,

- $h(n)$  je heuristická funkce, která odhaduje náklad cesty od vrcholu  $n$  k cílovému vrcholu  $t$ ,
- $f(n)$  je celkový odhadovaný náklad cesty přes vrchol  $n$ .

Algoritmus A\* vždy vybírá vrchol s nejnižší hodnotou  $f(n)$  pro další průzkum, což zajišťuje, že expanduje nejvíce slibné cesty. Aby byl A\* optimální (nalezl nejkratší cestu) a kompletní (dosáhl cíle, pokud existuje), musí heuristická funkce  $h(n)$  splňovat vlastnosti konzistence (monotonicity) a přípustnosti:

$$h(n) \leq d(n, t),$$

kde  $d(n, t)$  je skutečná nejkratší vzdálenost od  $n$  k  $t$ , a pro všechny sousedy  $n'$  vrcholu  $n$ :

$$h(n) \leq h(n') + c(n, n'),$$

kde  $c(n, n')$  je náklad hrany mezi  $n$  a  $n'$ .

### 3.7.2.2 Postup algoritmu A\*

1. Inicializace: Vytvoř prioritní frontu otevřených vrcholů (*open set*) s počátečním vrcholem  $s$ , kde  $g(s) = 0$  a  $f(s) = h(s)$ . Uzavřenou množinu (*closed set*) inicializuj jako prázdnou.
2. Vyber vrchol  $n$  s nejnižší hodnotou  $f(n)$  z otevřené množiny a přesuň ho do uzavřené množiny.
3. Pokud je  $n$  cílovým vrcholem  $t$ , rekonstruuj cestu od  $t$  k  $s$  pomocí předchůdců a ukonči algoritmus.
4. Pro každý sousední vrchol  $n'$  vrcholu  $n$ :
  - Pokud  $n'$  není v uzavřené množině, spočti  $g(n') = g(n) + c(n, n')$  a  $f(n') = g(n') + h(n')$ .
  - Pokud je  $n'$  v otevřené množině a nové  $g(n')$  je menší než původní, aktualizuj hodnotu  $g(n')$  a předchůdce  $n'$ .
5. Opakuj kroky 2–4, dokud otevřená množina není prázdná nebo není nalezen cíl.

### 3.7.2.3 Příklad heuristiky

Pro robotickou navigaci v 2D prostoru (např. po virtuálních silnicích v mapě) je běžnou heuristickou funkcí euklidovská vzdálenost:

$$h(n) = \sqrt{(x_n - x_t)^2 + (y_n - y_t)^2},$$

kde  $(x_n, y_n)$  jsou souřadnice aktuálního vrcholu  $n$  a  $(x_t, y_t)$  jsou souřadnice cílového vrcholu  $t$ . Tato heuristika je přípustná, protože nikdy neodhaduje vzdálenost vyšší, než je skutečná nejkratší cesta, a konzistentní, pokud jsou pohyby omezeny na kartézský prostor bez překážek.

### 3.7.3 Výhody a omezení A\*

- **Výhody:** A\* garantuje nalezení nejkratší cesty, pokud je heuristika přípustná a konzistentní. Je efektivnější než nevázané prohledávání (BFS, DFS) díky využití heuristiky, což snižuje počet prozkoumaných vrcholů.
- **OMEZENÍ:** Výpočetní náročnost roste s velikostí grafu a složitostí prostředí. Heuristika musí být pečlivě navržena, aby byla přípustná, jinak algoritmus nemusí najít optimální řešení.

### 3.7.4 NavFnPlanner

NavFnPlanner je globální plánovač v ROS2 Nav2, navržený pro výpočet optimální cesty v nákladové mapě. Na rozdíl od Dijkstrova algoritmu, který prohledává graf od startu, NavFnPlanner využívá potenciálová pole a vlnovou propagaci od cíle, což je rychlejší a lépe škálovatelné pro 2D mřížky. Jeho výhody spočívají v efektivitě a schopnosti generovat hladké cesty bez nutnosti prohledávat celý prostor, což je klíčové pro reálný čas. Proces zahrnuje tři kroky: generování nákladové mapy, výpočet potenciálového pole a extrakci trajektorie.

#### 3.7.4.1 Generování nákladové mapy

Nákladová mapa (*costmap*) je 2D mřížka, kde každá buňka  $(i, j)$  má hodnotu nákladu  $c(i, j) \in [0, 255]$ , přičemž:

- $c(i, j) = 255$ : Nepřístupná překážka.
- $c(i, j) = 0$ : Volný prostor.
- $0 < c(i, j) < 255$ : Bezpečnostní zóna kolem překážek.

Mapa je vytvořena z dat LIDARů nebo jiných senzorů a aktualizována v Nav2 modulem *costmap\_2d*. Bezpečnostní zóny jsou generovány procesem inflace, kde náklad klesá s vzdáleností od překážky (např. exponenciálně):

$$c(i, j) = 255 \cdot e^{-\alpha d(i, j)} \quad (3.16)$$

kde  $d(i, j)$  je vzdálenost buňky od nejbližší překážky a  $\alpha$  je parametr inflace (např. nastaven v *inflation\_radius*). Tím je zajištěn bezpečný odstup robota.

Mapa je poté převedena na graf  $G = (V, E)$ , kde  $V$  jsou buňky a  $E$  jsou hrany mezi sousedy (8-sousednost). Váha hrany  $w(u, v) = c(v)$  odpovídá nákladu cílové buňky.

#### 3.7.4.2 Výpočet optimální cesty

NavFnPlanner vypočítává optimální cestu pomocí potenciálového pole, což je skalární funkce  $P(v)$  definující minimální náklad cesty od uzlu  $v$  k cíli  $g$ . Na rozdíl od Dijkstrova algoritmu, který šíří vzdálenosti od startu, NavFn propaguje potenciály od cíle zpět ke startu:

##### 1. Inicializace potenciálu:

- $P(g) = 0$  pro cílový uzel.
- $P(v) = \infty$  pro ostatní uzly.
- Inicializuje se fronta  $Q$  s  $g$  jako počátečním uzlem.

##### 2. Propagace potenciálu:

Dokud  $Q \neq \emptyset$ , vyjmeme uzel  $u$  z fronty a pro jeho sousedy  $v$  aktualizujeme potenciál:

$$P(v) = \min\{P(v), P(u) + c(v) + w(u, v)\} \quad (3.17)$$

kde  $w(u, v)$  je vzdálenost mezi buňkami (např. 1 pro přímé sousedy,  $\sqrt{2}$  pro diagonální). Pokud se  $P(v)$  změní,  $v$  je přidán do  $Q$ . Tento proces pokračuje, dokud není dosažen start  $s$  nebo dokud se potenciály nestabilizují.

3. **Výsledek:** Po stabilizaci je  $P(s)$  minimální náklad cesty od  $s$  do  $g$ .

Potenciál  $P(v)$  lze chápat jako „výšku“ v topografické mapě, kde cíl je nejnižší bod a překážky jsou vysoké hory. Gradient  $\nabla P$  ukazuje směr nejkratší cesty.

NavFn je výhodný oproti Dijkstrovi algoritmu díky:

- **Efektivitě:** Propagace od cíle omezuje prohledávání na relevantní část mřížky.
- **Hladkostí:** Potenciálové pole generuje plynulé cesty bez ostrých zlomů.
- **Škálovatelnosti:** Vhodné pro velké mapy díky vlnové propagaci.

#### 3.7.4.3 Extrakce výsledné trajektorie

Trajektorie je extrahována z potenciálového pole gradientním sestupem od startu  $s$  k cíli  $g$ :

1. **Gradientní sestup:** Počínaje  $v_0 = s$  je další bod určen:

$$v_{i+1} = v_i - \alpha \nabla P(v_i) \quad (3.18)$$

kde  $\alpha$  je krok (např. velikost buňky) a  $\nabla P(v_i)$  je gradient potenciálu vypočtený jako:

$$\nabla P(v_i) = \left( \frac{P(i+1, j) - P(i-1, j)}{2\Delta}, \frac{P(i, j+1) - P(i, j-1)}{2\Delta} \right) \quad (3.19)$$

kde  $\Delta$  je rozteč mřížky.

2. **Posloupnost bodů:** Proces pokračuje, dokud  $v_k \approx g$ , a výsledkem je posloupnost bodů  $p = \langle s, v_1, \dots, g \rangle$ .
3. **Interpolace:** Body jsou interpolovány (např. spline křivkou) pro hladší trajektorii, která je předána lokálnímu plánovači.

#### 3.7.5 Lokální plánování s MPPI

Globální trajektorie z NavFnPlanneru je statická a nezohledňuje dynamické překážky ani kinematická omezení robota v reálném čase. Tuto roli plní lokální plánování, které upravuje pohyb podle aktuálních senzorických dat a dynamického modelu. V této práci je využit MPPI (Model Predictive Path Integral) regulátor, popsaný v sekci o prediktivním řízení 4.3.4, jako lokální plánovač.

MPPI generuje optimální řídicí vstupy (např. rychlosti kol) iterativním vzorkováním trajektorií v krátkém horizontu (např. 20 kroků) a vážením podle nákladové funkce, která zohledňuje shodu s globální trajektorií, vyhýbání se překážkám a kinematická omezení (viz sekce o MPPI). Význam lokálního plánování spočívá v:

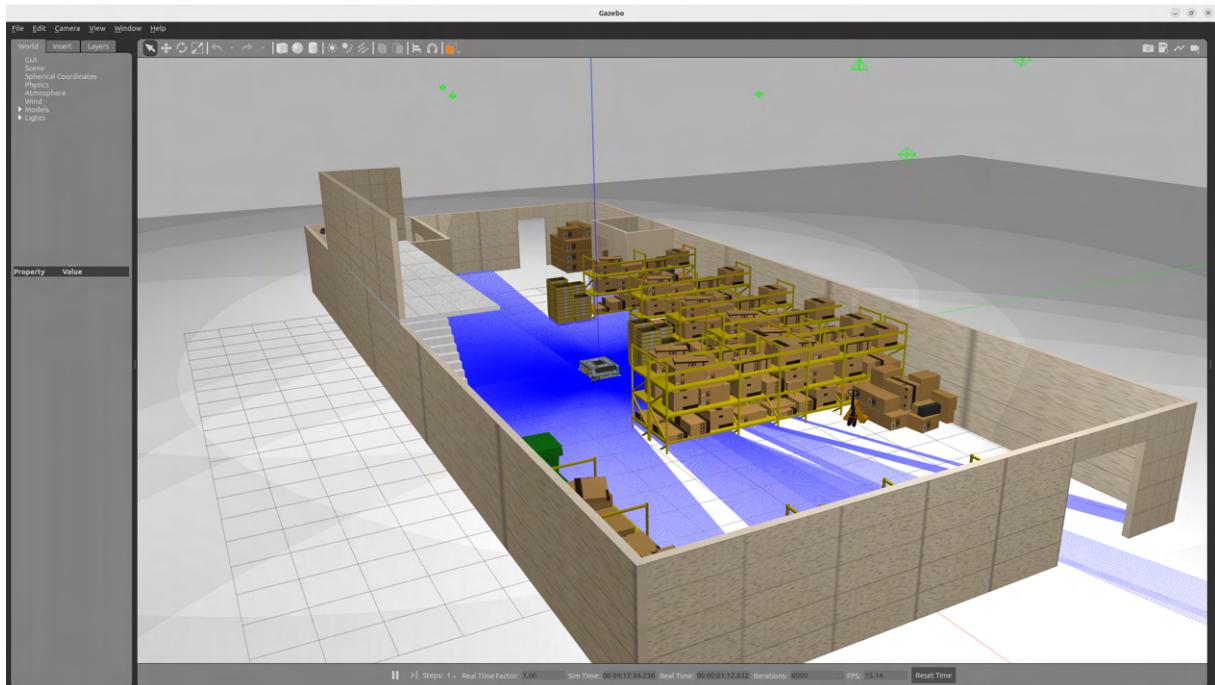
- **Reakci na dynamiku:** MPPI přeplánuje trajektorii při detekci nových překážek, což NavFnPlanner sám o sobě nezvládá.
- **Kinematická proveditelnost:** Zajistí, že pohyb odpovídá fyzickým možnostem robota (např. maximální rychlosť, poloměr otáčení).
- **Robustnost:** Kombinace globálního (NavFn) a lokálního (MPPI) plánování umožňuje efektivní navigaci v měnícím se prostředí.

MPPI tak doplňuje NavFnPlanner, čímž vytváří kompletní řešení pro autonomní navigaci v ROS2.

## 3.8 Simulační prostředí Gazebo

Gazebo (viz obrázek 3.13) je moderní simulační prostředí navržené primárně pro potřeby robotiky, které umožňuje realistické testování a vývoj robotických systémů ve virtuálním světě. Tento nástroj vyniká schopností simulovat fyzikální jevy jako jsou pohyb, kolize a interakce s prostředím, a zároveň poskytuje detailní vizualizaci robotů a jejich okolí. Dále umožňuje emulaci různých typů senzorů, což z něj činí významný prvek při vývoji robotických aplikací. V kontextu ROS2 se Gazebo etablovalo jako standardní platforma pro simulaci, která vývojářům nabízí prostředí pro testování a ladění algoritmů, například navigace, plánování trajektorií nebo lokalizace a mapování. Tato kapitola podrobně rozebírá jednotlivé aspekty Gazebo a jeho integrace s ROS2, aby čtenář získal ucelený přehled o jeho možnostech a využití v praxi.

Integrace Gazebo s ROS2 představuje významné zjednodušení procesu vývoje robotických systémů. Zatímco ROS2 poskytuje robustní infrastrukturu pro komunikaci mezi procesy, řízení robotů a zpracování dat ze senzorů, Gazebo doplňuje tuto funkcionality o realistickou simulaci fyzikálního prostředí. Tato synergie umožňuje vývojářům plynule přecházet mezi virtuálními simulacemi a reálnými experimenty, protože algoritmy vytvořené pro simulované roboty lze s minimálními úpravami aplikovat i na fyzický hardware. Gazebo je navíc podporováno širokou komunitou, která přispívá k jeho rozvoji a udržuje rozsáhlou knihovnu předpřipravených modelů robotů, prostředí a senzorů. Tato otevřená architektura, doplněná o možnost vytváření vlastních pluginů, umožňuje přizpůsobit simulace specifickým požadavkům projektů, což výrazně zrychluje vývojový cyklus.



Obrázek 3.13: Gazebo simulace AGV vybaveného LIDARy v definovaném prostředí.

### 3.8.1 Fyzikální engine

Základním pilířem Gazebo je jeho fyzikální engine, který zajišťuje realistickou simulaci dynamických procesů. Tento engine je navržen tak, aby byl vysoce modulární, což uživatelům umožňuje přizpůsobit simulaci podle konkrétních potřeb jejich robotických systémů at' už jde o dynamiku pohybu, vlastnosti prostředí anebo chování senzorů.

Systém Gazebo Physics [33] nabízí pokročilé možnosti konfigurace prostřednictvím volitelného aplikačního programovacího rozhraní (API). Díky modulárnímu přístupu lze aktivovat pouze ty funkce, které jsou pro danou simulaci nezbytné, což optimalizuje výpočetní výkon a snižuje zátěž na hardware. Klíčovou vlastností Gazebo Physics je jeho pluginové rozhraní, které umožňuje dynamicky načítat různé fyzikální enginy během běhu simulace. Tento flexibilní přístup usnadňuje přizpůsobení simulace specifickým požadavkům a podporuje integraci uživatelsky definovaných fyzikálních modelů. Výchozí implementace fyzikálního pluginu vychází z knihovny *dartsim*, což je moderní nástroj pro simulaci dynamiky tuhých těles. Tato knihovna je ideální pro aplikace vyžadující přesné modelování komplexních pohybů a interakcí. Pro simulace zaměřené na rychlé výpočty v rozlehlých prostředích, kde dynamické interakce nejsou prioritou, Gazebo nabízí alternativní engine nazvaný *Trivial Physics Engine*. Tento engine je optimalizován pro vysoký výkon a jednoduchost, díky čemuž je vhodný například pro testování kinematických algoritmů.

Fyzikální engine Gazebo zahrnuje následující klíčové funkcionality pro simulaci tuhých těles:

- **Definice modelů:** Modely jsou konstruovány pomocí formátu SDF (Simulation Description Format), který zajišťuje kompatibilitu a snadnou přenositelnost mezi různými simulacemi prostředími. Gazebo Physics využívá datové struktury *CompositeData*, které umožňují efektivní manipulaci s nativními datovými typy v rámci API. To zjednoduší přístup k fyzikálním datům a jejich integraci s dalšími komponentami simulace.
- **Kolizní objekty:** Engine podporuje širokou škálu geometrických tvarů, včetně kvádrů, koulí, válců, kuželů, kapslí, elipsoidů, síťových modelů (mesh) a výškových map (heightmaps). Uživatelé mohou definovat hmotnost, těžiště a další vlastnosti, což vede k realistickému chování objektů v simulaci.
- **Typy kloubů:** Gazebo umožňuje simulaci různých typů kloubů, například rotačních (revolute), prismatických (prismatic), pevných (fixed), kulových (ball), šroubových (screw) nebo univerzálních (universal), což pokrývá většinu potřeb robotických konstrukcí.
- **Manipulace s objekty:** Engine podporuje simulaci pohybových kroků, dynamickou změnu stavů objektů a aplikaci vnějších sil nebo momentů. Díky tomu lze realisticky modelovat interakce, jako je zvedání předmětů robotem nebo tlačení objektů v prostředí.

## 3.8.2 Návrh simulace

Jak již bylo zmíněno, Gazebo umožňuje snadné vytváření vlastních simulačních světů importem 3D modelů, například ve formátu DAE (COLLADA). Tyto modely mohou obsahovat textury a materiály, což zvyšuje vizuální věrnost a funkčnost simulace. Model robota je obvykle definován pomocí formátu URDF (Unified Robot Description Format), který lze rozšířit o pluginy pro simulaci senzorů, aktuátorů nebo specifických fyzikálních vlastností.

### 3.8.2.1 Model prostředí

Gazebo poskytuje nástroje pro tvorbu realistických simulačních světů složených ze statických i dynamických prvků. Statické prvky, jako jsou stěny, podlahy nebo pevné překážky, tvoří základní strukturu prostředí a lze je vytvořit pomocí jednoduchých geometrických tvarů (kvádry, válce, koule). Pro rychlé navrhování vnitřních prostor je k dispozici nástroj *Building Editor*, který umožňuje snadno konstruovat vícepodlažní stěnové systémy. Kromě těchto základních objektů Gazebo podporuje import komplexních 3D modelů vytvořených v programech, jako jsou Fusion 360, FreeCAD nebo Blender. Preferovaným formátem pro tyto modely je DAE, protože umožňuje zachovat textury a materiály, což přispívá k vizuálnímu i funkčnímu realismu simulace. Alternativní formáty, například STL nebo OBJ, jsou rovněž podporovány, avšak s omezenějšími možnostmi texturování.

Dynamické objekty, jako jsou pohyblivé dopravníky, vozíky nebo otevírací dveře, rozšiřují možnosti simulace o interaktivní prvky. Tyto objekty lze ovládat pomocí fyzikálních sil, skriptů nebo pluginů, což umožňuje testovat roboty v prostředí s měnícími se podmínkami. Gazebo dále nabízí detailní nastavení fyzikálních vlastností materiálů, včetně tření, elasticity a hmotnosti, což zajišťuje věrné znázornění interakcí mezi robotem a prostředím.

Textury a materiály hrají v simulaci dvojí roli: jednak zlepšují estetický dojem, jednak ovlivňují fyzikální vlastnosti, jako je odrazivost světla nebo tření povrchu. Tyto vlastnosti jsou definovány v URDF nebo SDF formátech a umožňují simulovat například kovové povrchy, drsné textury nebo průhledné materiály, což je zvláště důležité pro senzory, jako jsou kamery a LIDAR. Například správně nastavená odrazivost povrchu může ovlivnit detekci překážek LIDARovým senzorem, čímž se simulace přibližuje reálným podmínkám.

Simulační prostředí lze dále obohatit o senzory umístěné nejen na robotovi, ale i v prostředí samotném, například na stěnách nebo stropech. Příkladem může být LIDARový senzor na pevné konstrukci, který monitoruje pohyb robotů v reálném čase. Tento přístup umožňuje komplexní testování robotických algoritmů včetně jejich interakcí s dynamickým okolím. Pro velké simulace Gazebo nabízí možnosti optimalizace, jako je redukce detailů vzdálených objektů nebo omezení počtu simulovaných fyzikálních interakcí, což zvyšuje výpočetní efektivitu.

### 3.8.2.2 Model robota

URDF je standardní formát pro popis robotických modelů v Gazebo i ROS2. Tento XML formát umožňuje definovat kinematickou strukturu robota, jeho vizuální vzhled, fyzikální vlastnosti a senzory. Díky úzké integraci s ROS2 a Gazeblem je URDF ideální pro zajištění plynulého přechodu mezi simulací a reálným hardwarem.

URDF zahrnuje následující klíčové prvky:

- **Kinematická a dynamická konfigurace:** Popisuje pohyblivé části robota, klouby a jejich omezení, což je nezbytné pro simulaci pohybu.

- **Vizuální reprezentace:** Určuje vzhled robota v simulaci, včetně tvarů, barev, materiálů a textur.
- **Senzorové komponenty:** Umožňuje integrovat senzory, jako jsou například zmiňované kamery nebo LIDARy, přímo do modelu robota.
- **Gazebo pluginy:** Rozšiřují URDF o dodatečné funkce, například simulaci specifických senzorů, aktuátorů nebo řídicích mechanismů.

Níže je uveden příklad jednoduchého URDF modelu, který definuje základní robotický článek s vizuální a kolizní reprezentací:

```

1 <robot name="simple_robot">
2   <link name="base_link">
3     <inertial>
4       <mass value="5.0" />
5       <origin xyz="0 0 0.1" />
6       <inertia ixx="0.1" ixy="0" ixz="0" iyy="0.1" iyx="0" iyz="0" izz="0.1"
7     />
8     </inertial>
9     <visual>
10    <geometry>
11      <box size="1 1 0.2" />
12    </geometry>
13    <material name="blue"/>
14  </visual>
15  <collision>
16    <geometry>
17      <box size="1 1 0.2" />
18    </geometry>
19  </collision>
20 </link>
</robot>
```

Pro detailnější modely lze importovat 3D CAD soubory, které jsou ideálně převedeny do formátu DAE pomocí nástrojů, jako je Blender, aby bylo možné zachovat textury a materiály.

Jednou z nejvýznamnějších funkcí Gazebo je realistická simulace senzorů, včetně kamer, gyroskopů, akcelerometrů, LIDARů, GPS a tak dále. Tyto senzory jsou klíčové pro testování algoritmů vnímání a navigace. Například simulace LIDARu umožňuje generovat 2D nebo 3D bodová mračna, která jsou kompatibilní s algoritmy používanými v reálných robotech. LIDARový senzor je integrován do modelu robota prostřednictvím URDF a příslušného pluginu, který umožňuje nastavit parametry, jako je rozsah skenování, rozlišení, rychlosť otáčení nebo úroveň šumu. Níže je příklad konfigurace LIDARu včetně Gausovského šumu:

```

1 <gazebo reference="LIDAR_1_link">
2   <sensor name="LIDAR_1_sensor" type="ray">
3     <always_on>true</always_on>
4     <pose>0 0 0 0 0 0</pose>
5     <visualize>true</visualize>
6     <update_rate>100</update_rate>
7     <ray>
8       <scan>
9         <horizontal>
10        <samples>1351</samples>
11        <resolution>1</resolution>
12        <min_angle>-2.3562</min_angle>
13        <max_angle>2.3562</max_angle>
```

```

14      </horizontal>
15    </scan>
16    <range>
17      <min>0.06</min>
18      <max>30.0</max>
19      <resolution>0.05</resolution>
20    </range>
21    <noise>
22      <type>gaussian</type>
23      <mean>0.0</mean>
24      <stddev>0.01</stddev>
25    </noise>
26  </ray>
27  <plugin filename="libgazebo_ros_ray_sensor.so" name="LIDAR_1">
28    <ros>
29      <argument>~/out:=scan1</argument>
30    </ros>
31    <output_type>sensor_msgs/LaserScan</output_type>
32    <frame_name>LIDAR_1_link</frame_name>
33  </plugin>
34 </sensor>
35 </gazebo>

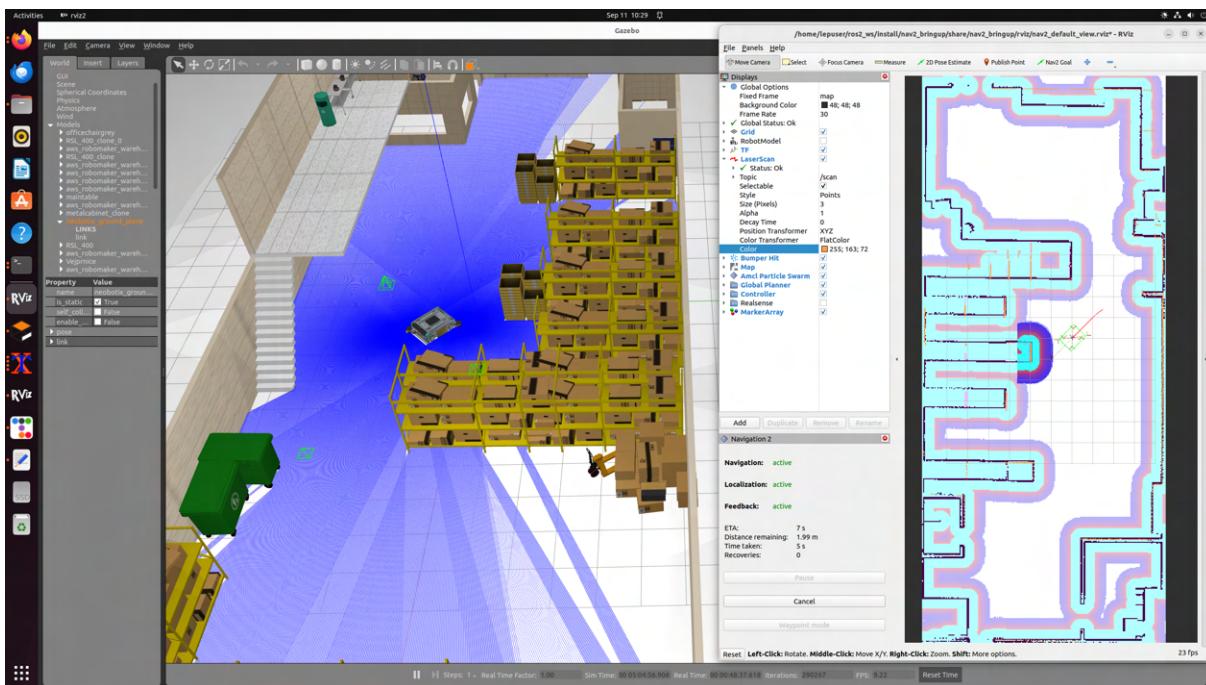
```

Simulovaná data z LIDARu jsou publikována do ROS2 témat, což umožňuje jejich zpracování stejnými algoritmy jako v reálném prostředí. Tento přístup výrazně usnadňuje vývoj a ladění bez nutnosti fyzického hardwaru.

### 3.8.2.3 Použití

Pokud je v Gazebo připraven kompletní model prostředí, robota a senzorů, lze spustit simulaci, která věrně napodobuje reálné podmínky. Tento proces nabízí jedinečnou možnost testovat a ladit algoritmy bez přístupu k fyzickému hardwaru, což je zvláště výhodné v případech, kdy je hardware nákladný, obtížně dostupný nebo kdy algoritmy ještě nejsou dostačeně otestovány. Díky realistickému modelování fyzikálních vlastností a senzorů lze simulaci považovat za téměř ekvivalentní reálnému systému, což umožňuje přímý přenos vyvinutých řešení do praxe.

Simulace v Gazebo byla v této práci použita pro testování navigačních algoritmů jako jsou SLAM, plánovače trajektorie nebo AMCL, které využívají data z LIDARu (viz obrázek 3.14). Simulované kamery zase umožňují testovat algoritmy vizuální odometrie nebo detekce objektů pomocí počítačového vidění. Kompatibilita s ROS2 navíc umožňuje spouštět a ladit kompletní ROS balíčky přímo v simulačním prostředí, včetně řízení pohybu, plánování trajektorií nebo manipulačních úloh.



Obrázek 3.14: Gazebo a RViz – testování autonomní navigace AGV v simulačním prostředí.

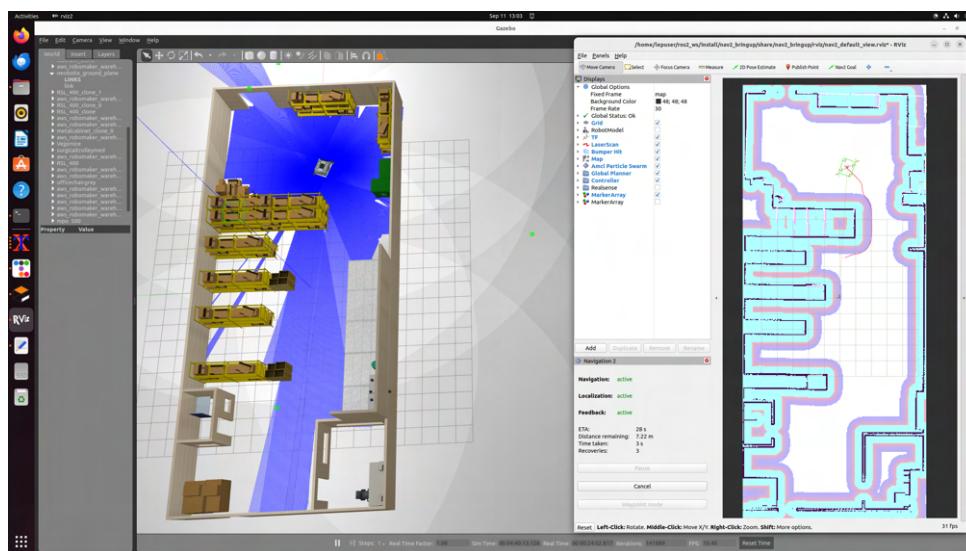
## 3.9 RViz

RViz (ROS Visualization) je výkonný interaktivní nástroj pro vizualizaci dat a interakci s robotickými systémy, vyvinutý jako klíčová součást ekosystému ROS2. Jeho primáním účelem je poskytnout vývojářům a výzkumníkům intuitivní grafické rozhraní, které usnadňuje ladění, testování a ovládání robotických aplikací prostřednictvím reprezentace široké škály datových formátů běžně používaných v robotice. V rámci ROS2 byl RViz přepracován tak, aby reflektoval přechod od první generace ROS na modernější architekturu založenou na middleware DDS (viz sekce 3.2). Tato změna přináší vyšší stabilitu, škálovatelnost a efektivitu komunikace mezi uzly, což RViz činí vhodným nástrojem pro komplexní robotické systémy.

Hlavní přínos RViz spočívá v jeho schopnosti zpracovávat a vizualizovat data v reálném čase, což výrazně zjednoduší analýzu a vývoj robotických algoritmů. Mezi jeho aplikace patří:

- **Vizualizace dat v reálném čase:** RViz umožnuje zobrazovat například skeny z LIDARů, trajektorie pohybu robota nebo generované mapy prostředí.
- **Interaktivní ovládání robota:** Uživatelé mohou definovat cílové polohy, zadávat odhady počáteční pozice nebo manuálně ladit chování algoritmů.
- **Analýza komplexních systémů:** RViz podporuje integraci dat z více senzorů a jejich kombinaci s algoritmy, jako jsou lokalizace, mapování (např. SLAM) nebo plánování trajektorií.

Na obrázku 3.15 je znázorněn praktický příklad využití RViz ve spojení s Gazebo (viz sekce 3.8). V levé části obrázku je simulované prostředí s robotem vybaveným LIDARY, zatímco v pravé části je zobrazeno grafické rozhraní RViz. RViz v tomto případě vizualizuje 2D mapu generovanou na základě LIDARových skenů, kde překážky jsou doplněny barevně vyznačenými bezpečnostními zónami. Robot je reprezentován zeleným obdélníkem, a červená čára ukazuje plánovanou trajektorii k cílovému bodu, který byl zadán přímo prostřednictvím RViz. Tento příklad ilustruje, jak RViz propojuje simulaci s analýzou a ovládáním robotického systému.



Obrázek 3.15: Ukázka nástroje RViz v kombinaci se simulací v Gazebo: vizualizace mapy a trajektorie robota.

### 3.9.1 Princip fungování

RViz funguje jako vizualizační nadstavba nad architekturou ROS2, která umožňuje zpracování a prezentaci dat získaných z robotického systému. Jeho funkcionalita je založena na odebírání zpráv z ROS2 témat a služeb (topics a services, viz sekce 3.3), které jsou publikovány jinými uzly v síti ROS2. Komunikace mezi RViz a ostatními uzly probíhá prostřednictvím middleware DDS, který zajišťuje rychlou, spolehlivou a distribuovanou výměnu dat. Tento mechanismus umožňuje RViz přijímat informace o poloze robota, senzorové výstupy nebo stav systému v reálném čase, a to i v rozsáhlých a heterogenních robotických aplikacích.

Důležitou součástí RViz je knihovna *TF2* (Transform Library 2), která spravuje transformace mezi různými souřadnicovými systémy v robotickém systému (viz sekce 3.4.1). Díky *TF2* může RViz správně interpretovat a kombinovat data pocházející z různých senzorů. Transformační matice aktualizované v reálném čase zajišťují, že všechny prvky systému jsou vykreslovány v konzistentním souřadnicovém rámci, což je nezbytné pro přesnou vizualizaci a analýzu.

Data přijatá z ROS2 témat jsou v RViz zobrazena pomocí tzv. displejů (*Displays*), což jsou konfigurovatelné moduly určené pro reprezentaci specifických typů dat, jako jsou například bodová mračna, mapy nebo obrazy. Uživatelé mají možnost přidávat, odstraňovat nebo upravovat tyto displeje podle aktuálních potřeb, což činí RViz vysoce přizpůsobivým nástrojem. Například displej pro 3D bodová mračna může být nastaven tak, aby zobrazoval pouze určitou oblast zájmu, zatímco displej pro mapu může zvýraznit bezpečnostní zóny kolem překážek.

### 3.9.2 Základní funkce a aplikace

RViz nabízí širokou škálu funkcí, které lze rozdělit do několika kategorií podle jejich využití v robotických aplikacích. Tyto funkce zahrnují vizualizaci senzorových dat, zobrazení stavu robota a interaktivní nástroje pro ovládání a analýzu.

#### 3.9.2.1 Vizualizace senzorových dat

RViz podporuje zobrazení různých typů senzorových dat, která jsou běžně generována robotickými systémy. Například pro tuto práci jsou užitečné:

- **LaserScan:** Slouží k vizualizaci 2D skenů z LIDARů. Tento typ dat je často využíván pro mapování a detekci překážek v rovině.
- **PointCloud:** Reprezentuje 3D bodová mračna, například z pokročilých LIDARů nebo stereokamer, a umožňuje detailní analýzu prostorových struktur.
- **Image:** Zobrazuje obrazová data z RGB kamer, hloubkových kamer nebo infračervených senzorů, což je užitečné pro testování algoritmů počítačového vidění.
- **Range:** Vizualizuje dosah ultrazvukových senzorů, což je vhodné pro jednoduché detekce vzdálenosti v blízkém okolí robota.

Tyto displeje lze kombinovat, například pro zobrazení 2D mapy z *LaserScan* spolu s obrazem z kamery, což usnadňuje komplexní analýzu prostředí.

### 3.9.2.2 Vizualizace stavu robota

RViz poskytuje nástroje pro zobrazení a analýzu aktuálního stavu robota v reálném čase:

- **TF (Transform Frames):** Graficky znázorňuje transformační rámce, tedy vzájemné polohy a orientace jednotlivých souřadnicových systémů robota. Například lze vizualizovat, jak se pohybuje senzor vůči tělu robota.
- **JointStates:** Zobrazuje polohy kloubů v kinematickém řetězci. V případě mobilního robota, jako je autonomní vozík, může jít o natočení kol, což je klíčové pro analýzu pohybu.
- **Odometry:** Reprezentuje vektory rychlosti a polohy robota odvozené z odometrických dat. Tato vizualizace pomáhá sledovat trajektorii a dynamiku pohybu robota v čase.

Tyto informace umožňují vývojářům rychle identifikovat chyby v konfiguraci nebo chování robota, například nesprávné natočení kol při navigaci.

### 3.9.2.3 Interaktivní ovládání a navigace

RViz obsahuje sadu interaktivních nástrojů, které rozšiřují jeho využití nad rámec pouhé vizualizace:

- **2D Pose Estimate:** Umožňuje zadat odhad počáteční polohy a orientace robota, což je zvláště užitečné při inicializaci navaigacích algoritmů, jako je částicový filtr pro lokalizaci (viz sekce 3.5.3).
- **2D Goal Pose:** Slouží k definování cílové polohy a orientace robota. Tento vstup je často předáván navaigacnímu stacku, například pro plánování trajektorie k cíli.

Tyto nástroje jsou integrovány s ROS2 komunikací, což znamená, že zadané cíle nebo odhady jsou okamžitě publikovány do příslušných témat a zpracovány dalšími uzly v systému.

### 3.9.2.4 Vizualizace map a modelů

RViz je schopen zobrazovat mapy generované algoritmy jako je SLAM a sledovat pohyb robota v těchto mapách:

- **OccupancyGrid:** Zobrazuje 2D mřížkovou mapu, kde jednotlivé buňky označují volný prostor, překážky nebo bezpečnostní zóny. Tento typ vizualizace je důležitý pro analýzu navaigacích algoritmů.
- **RobotDescription:** Umožňuje vykreslit 3D model robota přímo z jeho URDF popisu (viz sekce 3.8.2.2), což usnadňuje kontrolu souladu mezi simulací a reálným hardwarem.

Například v případě autonomního vozíku lze v RViz sledovat, jak se mapa postupně vytváří na základě LIDARových dat, a zároveň vizualizovat plánovanou trajektorii vůči aktuální poloze robota.

### 3.9.3 Praktické aplikace

RViz nachází uplatnění v široké škále robotických scénářů, zejména díky své flexibilitě a integraci s ROS2. Mezi typické příklady patří:

- **Ladění navigace:** vizualizace trajektorií, map a odometrických dat umožňuje rychle odhalit chyby v algoritmech lokalizace nebo plánování pohybu. Další běžnou chybou je správný výpočet odometrie či nastavení transformací.
- **Testování senzorů:** Kombinace dat z LIDARů, kamer a dalších senzorů v jednom rozhraní usnadňuje ověření jejich funkčnosti a přesnosti.
- **Vývoj autonomních systémů:** Interaktivní zadávání cílů a odhadů polohy zjednodušuje iterativní vývoj navaigacních algoritmů.

Vzhledem k tomu, že RViz je plně kompatibilní s Gazebo (viz sekce 3.8), lze jej využít k propojení simulace a analýzy v reálném čase. Například simulovaný robot v Gazebo publikuje senzorová data do ROS2 témat, která RViz následně vizualizuje, což umožňuje vývojářům ladit algoritmy bez nutnosti fyzického hardwaru.

# Kapitola 4

## Teoretické podklady

---

Tato kapitola se zaměřuje na teoretické základy a analýzu matematických metod navržených pro řešení navaigacních úloh autonomní robotické platformy. Nejprve jsou odvozeny kinematické modely dvou typů podvozků, umožňující přepočet pohybu robota na rychlosti kol. Následně je popsána technika spojování dat z více LIDARů do jednotné datové struktury, která optimalizuje využití senzorických informací. Dále jsou analyzovány metody prediktivního řízení pro sledování trajektorie při autonomní navigaci, zvolené pro svou schopnost zohlednit okolní prostředí a dynamické překážky při optimalizaci trasy. Alternativně je představen algoritmus Pure Pursuit, vhodný pro statické, předem definované trajektorie, s uplatněním v odlišné aplikaci. Kapitola rovněž zahrnuje představení algoritmu DBSCAN pro detekci malých objektů přímo z LIDARových dat, což je dále využito pro detekci palety. Poslední část je věnována prohledávání grafů aplikovanému na vyhledávání nejkratší cesty v systému virtuálních drah.

## 4.1 Kinematický model

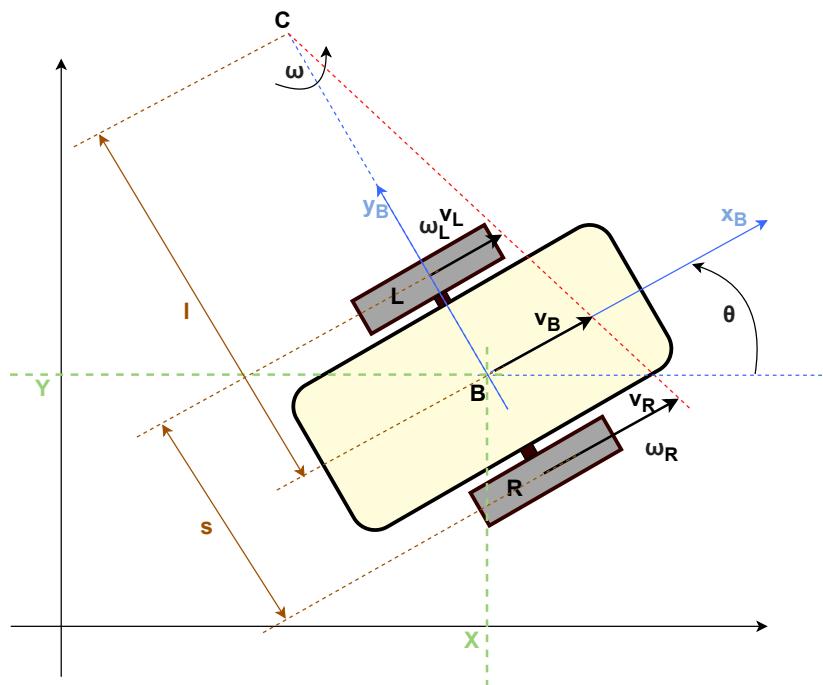
Kinematický model je základním nástrojem pro popis pohybu mobilních robotů v prostoru, což je nutné pro jejich autonomní navigaci a řízení. Tato kapitola se zaměřuje na odvození kinematických vztahů pro dva typy podvozků: diferenciální podvozek a podvozek s mecanum koly. Cílem je definovat dopřednou kinematiku, která převádí úhlové rychlosti kol na pohyb vozíku, inverzní kinematiku pro výpočet potřebných rychlostí kol z požadovaného pohybu a jejich transformaci do globálního souřadného systému pro úplný popis trajektorie v rovině. Tyto modely tvoří základ například pro prediktivní řízení (MPC / MPPI), které jsou dále rozpracovány v následujících kapitolách.

### 4.1.1 Diferenciální podvozek

Diferenciální podvozek je základní mechanismus pohybu mnoha mobilních robotů, včetně vozíku analyzovaného v této práci. Jeho kinematický model popisuje vztah mezi úhlovými rychlostmi kol a pohybem středu vozíku v lokálním i globálním souřadném systému. Tato sekce odvozuje dopřednou a inverzní kinematiku pro lokální pohyb a doplňuje transformaci do globálních souřadnic, což je nezbytné pro aplikace, jako je sledování trajektorie v globálním prostoru.

#### 4.1.1.1 Dopředná kinematika

Předpokládáme, že známe úhlové rychlosti kol  $\omega_L$  (levé kolo) a  $\omega_R$  (pravé kolo), rozvor kol  $d$  (vzdálenost mezi středy kol) a poloměr kol  $r$ . Cílem je určit lineární rychlosť  $v$  a úhlovou rychlosť  $\omega$  středu vozíku  $B$  (který je zároveň i těžištěm) kolem bodu  $C$  v lokálním souřadném systému.



Obrázek 4.1: Geometrie pohybu diferenciálního podvozku..

Z obrázku 4.1 je zřejmé, že lineární rychlosti kol  $v_L$  a  $v_R$  lze vyjádřit vzhledem k úhlové rychlosti  $\omega$  a vzdálenosti od středu otáčení  $l$ :

$$\begin{aligned} v_R &= \omega \left( l + \frac{d}{2} \right) \\ v_L &= \omega \left( l - \frac{d}{2} \right) \end{aligned} \quad (4.1)$$

Jelikož  $l$  (vzdálenost středu otáčení od středu vozíku) ani  $\omega$  neznáme, vyjádříme je z rovnic (4.1):

$$\omega = \frac{v_L}{l - \frac{d}{2}} \quad (4.2)$$

Po dosazení rovnice (4.2) do první rovnice (4.1) získáme:

$$\begin{aligned} v_R &= \frac{v_L}{l - \frac{d}{2}} \left( l + \frac{d}{2} \right) \\ v_R \left( l - \frac{d}{2} \right) &= v_L \left( l + \frac{d}{2} \right) \\ l(v_R - v_L) &= \frac{d}{2}(v_R + v_L) \\ l &= \frac{d(v_R + v_L)}{2(v_R - v_L)} \end{aligned} \quad (4.3)$$

Dosazením rovnice (4.3) do (4.2) vyjádříme  $\omega$ :

$$\begin{aligned} v_L &= \omega \left( \frac{d(v_R + v_L)}{2(v_R - v_L)} - \frac{d}{2} \right) \\ v_L &= \frac{\omega d}{2} \left( \frac{v_R + v_L}{v_R - v_L} - 1 \right) \\ v_L &= \frac{\omega d}{2} \left( \frac{2v_L}{v_R - v_L} \right) \\ \omega &= \frac{v_R - v_L}{d} \end{aligned} \quad (4.4)$$

Lineární rychlosť středu vozíku  $v$  lze zapsat jako:

$$v = \omega \cdot l$$

Po dosazení (4.3) a (4.4):

$$v = \frac{v_R - v_L}{d} \cdot \frac{d(v_R + v_L)}{2(v_R - v_L)} = \frac{v_R + v_L}{2} \quad (4.5)$$

Lineární rychlosti kol souvisí s úhlovými rychlostmi přes poloměr  $r$ :

$$v_R = r\omega_R$$

$$v_L = r\omega_L \quad (4.6)$$

Po dosazení (4.6) do (4.4) a (4.5) získáme finální rovnice dopředné kinematiky v lokálním souřadném systému:

$$v = \frac{r}{2}(\omega_R + \omega_L) \quad (4.7)$$

$$\omega = \frac{r}{d}(\omega_R - \omega_L) \quad (4.8)$$

#### 4.1.1.2 Inverzní kinematika

Inverzní kinematika je nezbytná pro řízení robota, protože umožňuje z požadovaných veličin  $v$  a  $\omega$  vypočítat potřebné úhlové rychlosti kol  $\omega_R$  a  $\omega_L$ . Známe  $v$ ,  $\omega$ , rozvor  $d$  a poloměr  $r$ .

Pro odvození využijeme rovnic dopředné kinematiky (4.7) a (4.8):

$$\frac{2v}{r} - \omega_L = \omega_R \quad (4.9)$$

$$\frac{\omega d}{r} + \omega_L = \omega_R \quad (4.10)$$

Porovnáním (4.9) a (4.10):

$$\begin{aligned} \frac{2v}{r} - \omega_L &= \frac{\omega d}{r} + \omega_L \\ \frac{2v - \omega d}{r} &= 2\omega_L \\ \omega_L &= \frac{2v - \omega d}{2r} \end{aligned} \quad (4.11)$$

Dosazením (4.11) do (4.10):

$$\begin{aligned} \omega_R &= \frac{\omega d}{r} + \frac{2v - \omega d}{2r} = \frac{2\omega d + 2v - \omega d}{2r} = \frac{2v + \omega d}{2r} \\ \omega_R &= \frac{2v + \omega d}{2r} \end{aligned} \quad (4.12)$$

Finální rovnice inverzní kinematiky v lokálním souřadném systému vyjdou:

$$\omega_R = \frac{2v + \omega d}{2r} \quad (4.13)$$

$$\omega_L = \frac{2v - \omega d}{2r} \quad (4.14)$$

### 4.1.1.3 Pohyb v globálním souřadném systému

Lokální kinematický model definovaný rovnicemi (4.7) a (4.8) popisuje pohyb relativně k ose robota. Pro kompletní popis pohybu v globálním souřadném systému  $(X, Y, \theta)$ , kde  $X$  a  $Y$  jsou souřadnice středu vozíku a  $\theta$  je jeho orientace vůči globální ose  $X$ , musíme transformovat lokální rychlosti  $v$  a  $\omega$  do globálních souřadnic. Orientace  $\theta$  je úhel mezi lokální osou  $x$  vozíku a globální osou  $X$ , a úhlová rychlosť  $\omega = \dot{\theta}$  popisuje její změnu v čase.

Pohyb robota v globálním souřadném systému je dán diferenciálními rovnicemi odvozenými z geometrie pohybu (viz obrázek 4.1), kde lineární rychlosť  $v$  je orientována podle  $\theta$ :

Globální rychlosti  $\dot{X}$  a  $\dot{Y}$  jsou projekcí lokální rychlosti  $v$  podle orientace  $\theta$ :

$$\dot{X} = v \cos \theta \quad (4.15)$$

$$\dot{Y} = v \sin \theta \quad (4.16)$$

Úhlová rychlosť  $\omega$  přímo určuje změnu orientace:

$$\dot{\theta} = \omega \quad (4.17)$$

Dosazením rovnic dopředné kinematiky (4.7) a (4.8) do (4.15)–(4.17) získáme kompletní kinematický model v globálním systému:

$$\dot{X} = \frac{r}{2}(\omega_R + \omega_L) \cos \theta \quad (4.18)$$

$$\dot{Y} = \frac{r}{2}(\omega_R + \omega_L) \sin \theta \quad (4.19)$$

$$\dot{\theta} = \frac{r}{d}(\omega_R - \omega_L) \quad (4.20)$$

Tyto rovnice popisují vývoj polohy a orientace robota v globálním systému na základě úhlových rychlostí kol. Pro numerickou simulaci lze tyto diferenciální rovnice integrovat v čase (např. Eulerovou metodou), čímž získáme trajektorii  $(X(t), Y(t), \theta(t))$  pro dané  $\omega_R$  a  $\omega_L$ .

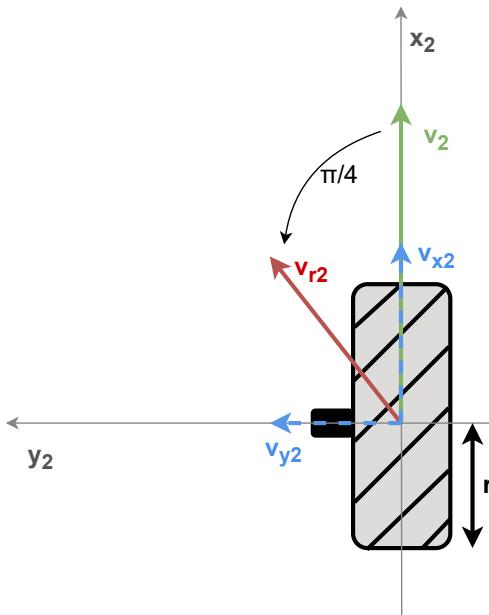
## 4.1.2 Podvozek s mecanum kolys

Podvozek s mecanum kolys představuje pokročilý mechanismus pohybu, který umožňuje robotovi pohyb libovolným směrem díky speciální konstrukci kol. Tato kola jsou vybavena válečky připevněnými pod úhlem  $45^\circ$ , které se mohou volně otáčet, což přenáší síly z rotace kol do různých směrů v závislosti na směru otáčení. Tato vlastnost umožňuje vozíku dosáhnout plné pohyblivosti v rovině ( $v_x, v_y$ ) a zároveň rotace kolem osy  $z$  ( $\omega$ ). Kinematický model mecanum podvozku je složitější než u diferenciálního podvozku, a proto v této podkapitole začínám inverzní kinematikou. Tento přístup je výhodnější, protože odvození rychlostí jednotlivých kol z požadovaného pohybu vozíku lépe vystihuje fyzikální principy přenosu sil přes válečky a usnadňuje pochopení celkové dynamiky systému.

### 4.1.2.1 Inverzní kinematika

Úhel  $45^\circ$ , pod kterým jsou válečky připevněny, určuje, jak se vektor lineární rychlosti kola přenáší na výsledný pohyb vozíku. Z obrázku 4.2 je zřejmé, že lineární rychlosť kola  $v_2$  odpovídající úhlové rychlosti je dána vztahem:

$$\omega_2 = \frac{v_2}{r} \quad (4.21)$$



Obrázek 4.2: Přenášení sil na mecanum kolu 2 (pravé přední)

V důsledku volně rotujících válečků je tato rychlosť  $v_2$  přenesena na výslednou lineární rychlosť  $v_{r2}$ , která je pootočena o  $45^\circ$ :

$$v_{r2} = v_2 \sin\left(\frac{\pi}{4}\right) \approx v_2 \cdot 0.7 \quad (4.22)$$

Z rovnice (4.22) vyplývá, že výsledná síla je zredukována přibližně o 30 % kvůli sklonu válečků. Komponenty sil rovnoběžné s osou válečků nepřispívají k pohybu, zatímco kolmé komponenty mají plný účinek, což vede k tomuto částečnému přenosu (70 %).

Na základě obrázku 4.2 určíme složky rychlosti  $v_{r2}$  ve směru os  $x_2$  a  $y_2$ :

$$v_{x2} = v_{r2} \cos(\gamma) \quad (4.23)$$

$$v_{y2} = v_{r2} \sin(\gamma) \quad (4.24)$$

Složky (4.23) a (4.24) sečteme (obě působí v kladném směru os  $x_2$  a  $y_2$ ):

$$v_{x2} + v_{y2} = v_{r2} (\cos(\gamma) + \sin(\gamma)) \quad (4.25)$$

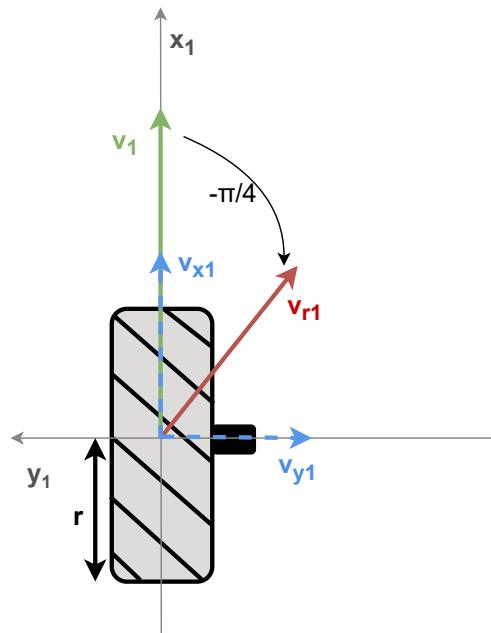
Po dosazení (4.22) do (4.25):

$$v_{x2} + v_{y2} = v_2 \sin\left(\frac{\pi}{4}\right) (\cos(\gamma) + \sin(\gamma)) = v_2 \frac{\sqrt{2}}{2} \sqrt{2} = v_2 \quad (4.26)$$

S využitím (4.21) tedy platí:

$$\omega_2 = \frac{v_2}{r} = \frac{v_{x2} + v_{y2}}{r} \quad (4.27)$$

Pro kolo číslo 3 (pravé zadní) platí stejný princip. U kol 1 a 4 (levé přední a zadní) se liší směr natočení válečků, ale postup je analogický.



Obrázek 4.3: Přenášení sil na mecanum kolu 1 (levé přední)

Znovu vycházíme ze vztahu mezi úhlovou a lineární rychlostí:

$$\omega_1 = \frac{v_1}{r} \quad (4.28)$$

uvážíme sklon válečků:

$$v_{r1} = v_1 \sin\left(\frac{\pi}{4}\right) \approx v_1 \cdot 0.7 \quad (4.29)$$

A určíme příspěvky složek rychlosti ve směru os  $x_3$  a  $y_3$  podle obrázku 4.3:

$$v_{x1} = v_{r1} \cos(\gamma) \quad (4.30)$$

$$v_{y1} = v_{r1} \sin(\gamma) \quad (4.31)$$

Protože  $v_{x1}$  působí v kladném směru osy  $x$  a  $v_{y1}$  v záporném směru osy  $y$  (viz 4.3), jednotlivé složky odečteme. (Také je dobré upozornit, že v tomto případě  $\gamma = -45^\circ$ ):

$$v_{x1} - v_{y1} = v_{r1} (\cos(\gamma) - \sin(\gamma)) \quad (4.32)$$

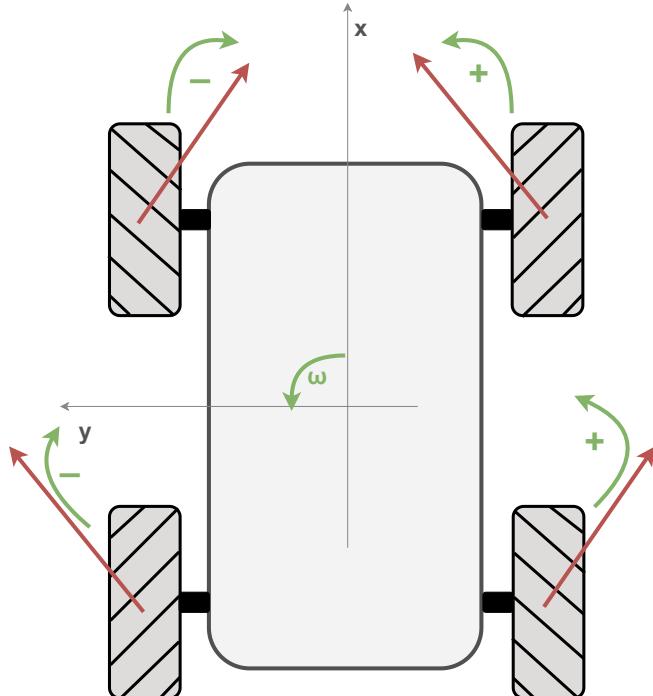
Po dosazení:

$$v_{x1} - v_{y1} = v_1 \sin\left(\frac{\pi}{4}\right) \left( \cos\left(-\frac{\pi}{4}\right) - \sin\left(-\frac{\pi}{4}\right) \right) = v_1 \frac{\sqrt{2}}{2} \sqrt{2} = v_1 \quad (4.33)$$

Tedy:

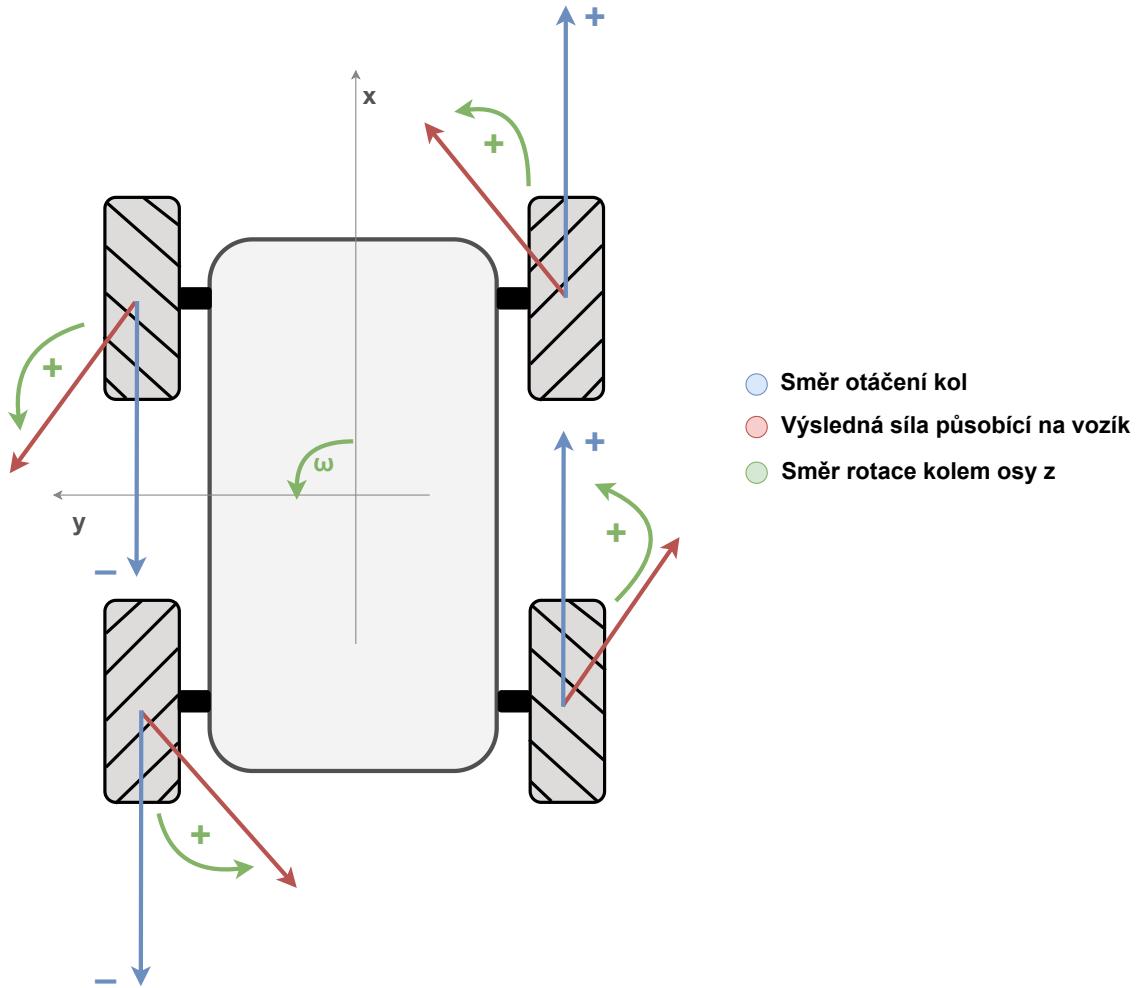
$$\omega_1 = \frac{v_1}{r} = \frac{v_{x1} - v_{y1}}{r} \quad (4.34)$$

Dále je třeba zohlednit rotaci vozíku kolem osy  $z$ .



Obrázek 4.4: Zohlednění vlivu otáčení vozíku kolem osy  $z$

Z obrázku 4.4 je patrné, že rotace jednoho kola v kladném směru (dopředu) táhne jeho roh ve směru vektoru  $v_{ri}$ , což způsobuje otáčení vozíku. Pravá kola (2 a 3) při kladné rotaci vyvolávají otáčení proti směru hodinových ručiček ( $\omega_i > 0$ ), levá kola (1 a 4) opačně. Pro rotaci namísto ( $v_x = v_y = 0$ ) musí pravá kola rotovat dopředu a levá dozadu (viz obrázek 4.5).

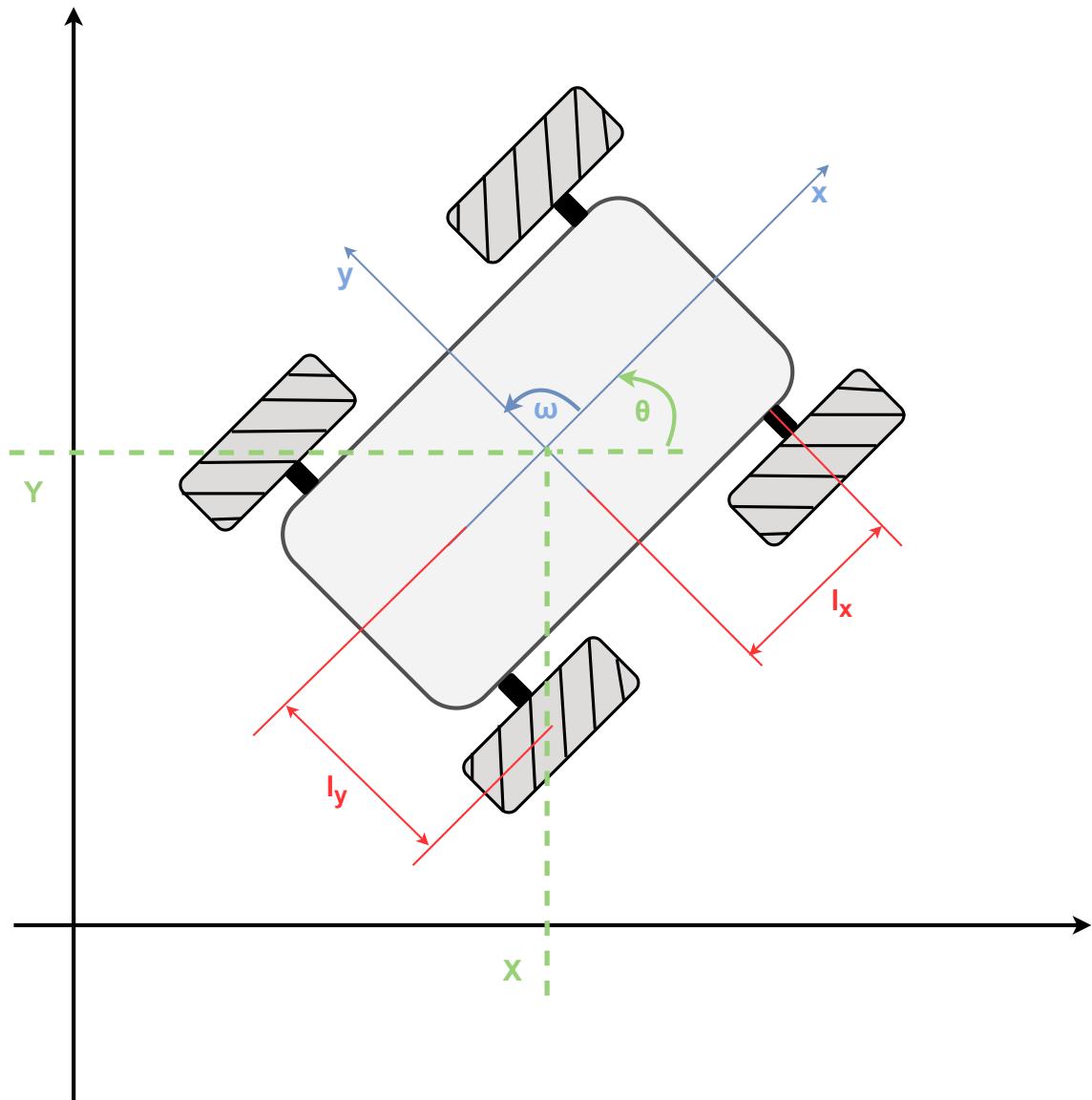


Obrázek 4.5: Otáčení v kladném směru úhlové rychlosti  $\omega$

Rotace závisí na vzdálenostech  $l_X$  a  $l_Y$  od středu vozíku k osám kol (viz obrázek 4.6), které určují moment síly působící na otáčení. Finální inverzní kinematika zohledňuje  $v_x$ ,  $v_y$  a  $\omega$ :

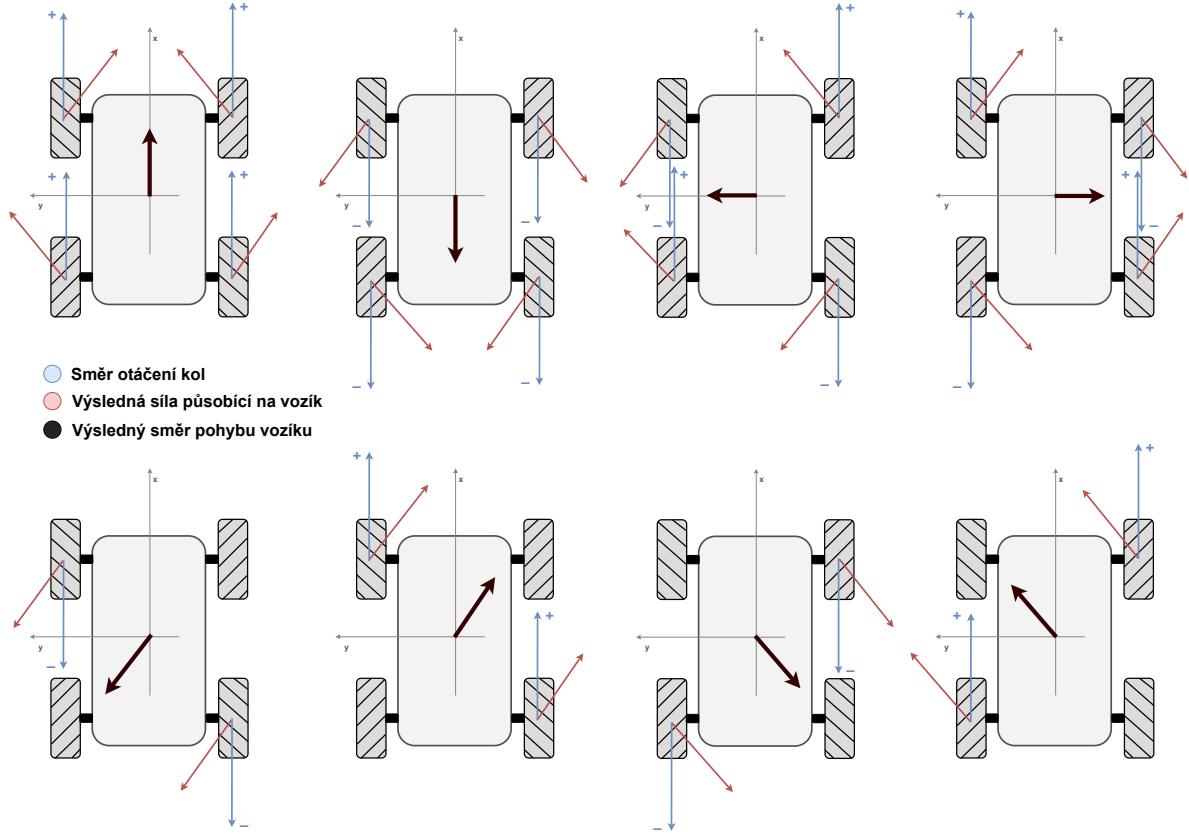
$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & -1 & -(l_X + l_Y) \\ 1 & 1 & (l_X + l_Y) \\ 1 & 1 & -(l_X + l_Y) \\ 1 & -1 & (l_X + l_Y) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \quad (4.35)$$

Na tom samém obrázku 4.6 je rovněž naznačen vztah mezi globálním souřadným systémem a lokálním souřadným systémem robota. Souřadné systémy jednotlivých kol jsou vůči lokálnímu souřadanému systému vztaženy pomocí vzdáleností  $l_X$  a  $l_Y$ . Vztah mezi lokálním a globálním souřadným systémem je řešen v části dále, vztah mezi souřadnými systémy kol a souřadným systémem robota (pouze posunutí) jsou řešeny ve vztahu (4.35).



Obrázek 4.6: Vztah globálního a lokálního souřadného systému

Na obrázku 4.7 je naznačen princip pohybu vozíku v různých směrech v závislosti na směru otáčení a rychlosti jednotlivých kol.



Obrázek 4.7: Možnosti pohybu vozíku v závislosti na rychlosti jednotlivých kol

#### 4.1.2.2 Dopředná kinematika

Dopředná kinematika mecanum podvozku představuje základní krok k pochopení, jak se pohyb jednotlivých kol překládá na celkový pohyb vozíku v jeho lokálním i globálním souřadném systému. Na rozdíl od inverzní kinematiky, která určuje rychlosti kol na základě požadovaného pohybu, dopředná kinematika odvozuje lineární rychlosti  $v_x, v_y$  a úhlovou rychlosť  $\omega$  vozíku z měřených úhlových rychlostí kol. Tento přístup je nezbytný pro odometrii a integraci s navigačními algoritmy, protože umožňuje rekonstruovat trajektorii vozíku na základě dat z enkodérů.

Využijeme výsledné rovnice z inverzní kinematiky (4.35):

$$r\omega_1 = v_x - v_y - (l_X + l_Y)\omega \quad (4.36)$$

$$r\omega_2 = v_x + v_y + (l_X + l_Y)\omega \quad (4.37)$$

$$r\omega_3 = v_x + v_y - (l_X + l_Y)\omega \quad (4.38)$$

$$r\omega_4 = v_x - v_y + (l_X + l_Y)\omega \quad (4.39)$$

a jejich vhodným sečtením a odečtením vyseparujeme  $v_x, v_y$  a  $\omega$ :

$$r(\omega_1 + \omega_2 + \omega_3 + \omega_4) = 4v_x \quad (4.40)$$

$$r(-\omega_1 + \omega_2 + \omega_3 - \omega_4) = 4v_y \quad (4.41)$$

$$r(-\omega_1 + \omega_2 - \omega_3 + \omega_4) = 4(l_X + l_Y)\omega \quad (4.42)$$

Úpravou (4.40)–(4.42) získáme finální rovnice dopředné kinematiky:

$$v_x = \frac{r}{4}(\omega_1 + \omega_2 + \omega_3 + \omega_4) \quad (4.43)$$

$$v_y = \frac{r}{4}(-\omega_1 + \omega_2 + \omega_3 - \omega_4) \quad (4.44)$$

$$\omega = \frac{r}{4(l_X + l_Y)}(-\omega_1 + \omega_2 - \omega_3 + \omega_4) \quad (4.45)$$

Maticově:

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \frac{r}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ \frac{-1}{l_X+l_Y} & \frac{1}{l_X+l_Y} & \frac{-1}{l_X+l_Y} & \frac{1}{l_X+l_Y} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \quad (4.46)$$

#### 4.1.2.3 Pohyb v globálním souřadném systému

Lokální kinematický model  $(v_x, v_y, \omega)$  popisuje pohyb vozíku v jeho vlastním souřadném systému. Pro popis pohybu v globálním souřadném systému  $(X, Y, \theta)$ , kde  $X$  a  $Y$  jsou globální souřadnice středu vozíku a  $\theta$  je orientace vůči globální ose  $X$ , transformujeme lokální rychlosti do globálních souřadnic. Orientace  $\theta$  určuje směr lokální osy  $x$  vozíku vůči globální ose  $X$ , a úhlová rychlosť  $\dot{\theta} = \dot{\theta}$  popisuje její změnu.

Globální rychlosti  $\dot{X}$  a  $\dot{Y}$  jsou projekcí lokálních rychlostí  $v_x$  a  $v_y$  podle  $\theta$ :

$$\dot{X} = v_x \cos \theta - v_y \sin \theta \quad (4.47)$$

$$\dot{Y} = v_x \sin \theta + v_y \cos \theta \quad (4.48)$$

$$\dot{\theta} = \omega \quad (4.49)$$

Dosazením rovnic dopředné kinematiky (4.43)–(4.45) do (4.47)–(4.49) získáme kompletní globální kinematický model:

$$\dot{X} = \frac{r}{4}(\omega_1 + \omega_2 + \omega_3 + \omega_4) \cos \theta - \frac{r}{4}(-\omega_1 + \omega_2 + \omega_3 - \omega_4) \sin \theta \quad (4.50)$$

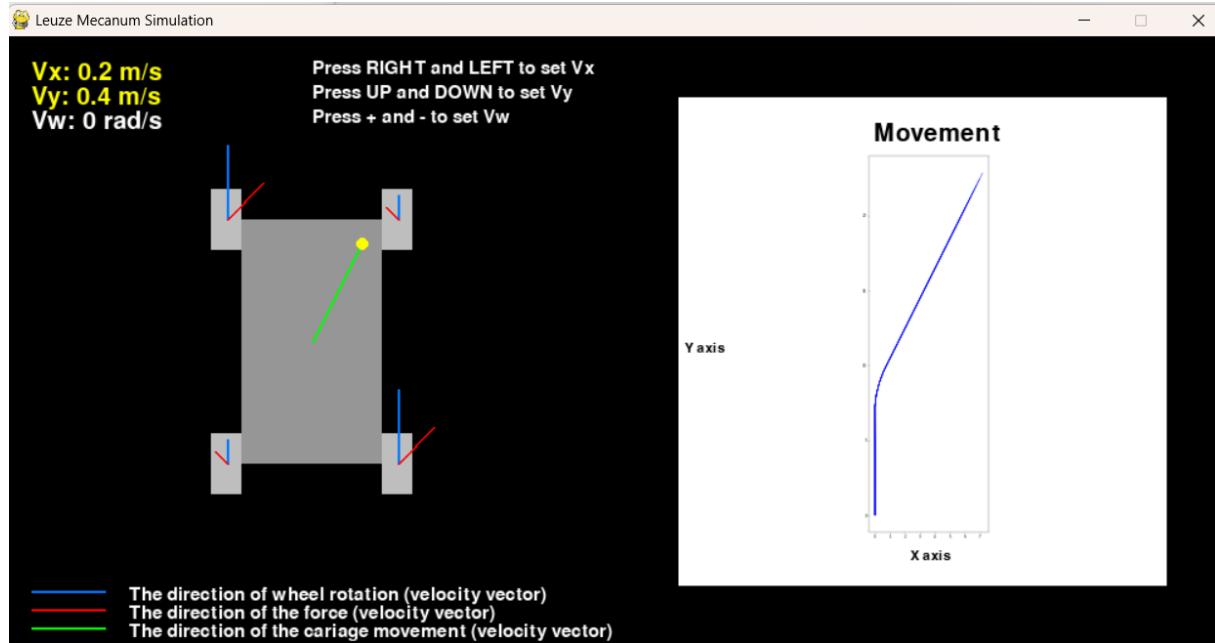
$$\dot{Y} = \frac{r}{4}(\omega_1 + \omega_2 + \omega_3 + \omega_4) \sin \theta + \frac{r}{4}(-\omega_1 + \omega_2 + \omega_3 - \omega_4) \cos \theta \quad (4.51)$$

$$\dot{\theta} = \frac{r}{4(l_X + l_Y)}(-\omega_1 + \omega_2 - \omega_3 + \omega_4) \quad (4.52)$$

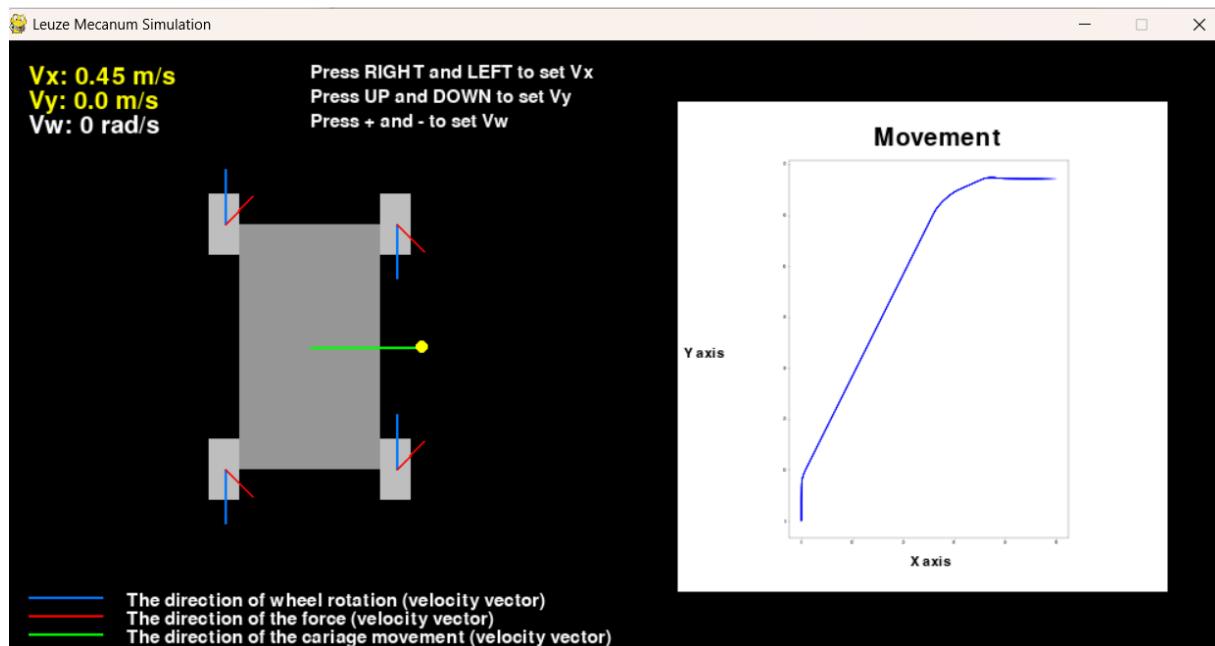
Tyto diferenciální rovnice popisují vývoj polohy a orientace vozíku v globálním systému na základě úhlových rychlostí kol (takzvaná odometrie). Numerická integrace (např. Eulerovou metodou) umožňuje vypočítat trajektorii  $(X(t), Y(t), \theta(t))$  pro dané  $\omega_1, \omega_2, \omega_3, \omega_4$ .

#### 4.1.2.4 Simulační aplikace

Pro otestování chování modelu jsem vytvořil jednoduchou aplikaci, která vizualizuje jednotlivé vektory. Uživatel zadává požadovaný směr a velikost lineární rychlosti a úhlovou rychlosť kolem osy  $z$ , inverzní kinematika počítá rychlosť kol, a dopředný model zpětně určuje polohu v prostoru, která je vizualizována jako trajektorie. K práci je přiloženo ukázkové [Video 1](#) [1] i aplikace.



Obrázek 4.8: Ukázka aplikace pro test funkčnosti kinematického modelu



Obrázek 4.9: Ukázka aplikace pro test funkčnosti kinematického modelu 2

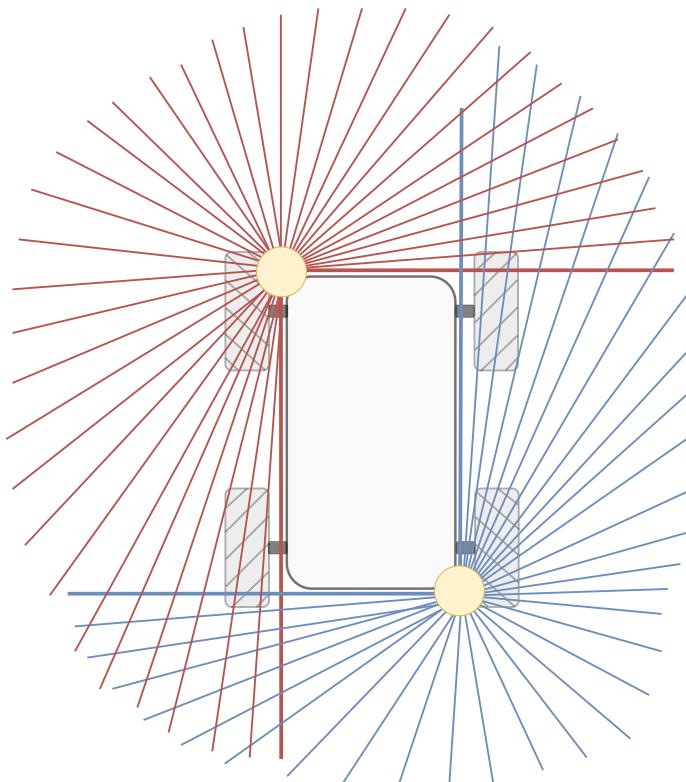
[1] [https://youtu.be/S\\_b62l5s\\_JA](https://youtu.be/S_b62l5s_JA)

## 4.2 Spojení dvou LIDARových skenů do virtuálního LIDARu

Tato kapitola se zabývá teoretickými a praktickými aspekty spojení dat z dvou 2D LIDARových senzorů umístěných na autonomním vozíku (AGV) do jednoho virtuálního skenu, jehož souřadný systém je umístěn v počátku souřadné soustavy vozíku. Tento přístup je nezbytný pro zpracování dat v rámci navigačního stacku ROS2, který podporuje pouze jeden zdroj LIDARových dat, a usnadňuje navigaci a mapování v reálném čase.

### 4.2.1 Problém a konfigurace LIDARů

Použité LIDARY, konkrétně modely RSL200 nebo RSL400, mají omezený úhel skenování  $270^\circ$ , což nestačí pro pokrytí celého prostoru kolem vozíku. Navíc normy pro navigaci AGV stanovují maximální přípustnou výšku umístění LIDARu – čím nižší umístění, tím lepší detekce nízkých překážek. Z těchto důvodů není možné umístit jeden LIDAR na vrchol vozíku, ani kdyby disponoval plným úhlem  $360^\circ$ . Místo toho jsou dva LIDARY umístěny v opačných rozích vozíku, což zajišťuje pokrytí tříčtvrtelního rozhledu ( $270^\circ$  na každém senzoru) a efektivní pokrytí prostoru kolem obdélníkové konstrukce vozíku. Tato konfigurace je znázorněna na obrázku 4.10.



Obrázek 4.10: Rozmístění dvou LIDARů na obdélníkovém vozíku.

Problém spočívá v tom, že každý LIDAR poskytuje data označená úhly v rozmezí  $-135^\circ$  až  $135^\circ$  vzhledem k vlastnímu středu souřadné soustavy, který je posunut relativně k počátku souřadné soustavy vozíku. Data z obou LIDARů tedy mapují různé části prostoru, ale jejich kombinace vyžaduje transformaci do společného referenčního rámce.

## 4.2.2 Transformace dat do virtuálního LIDARu

ROS2 není primárně navržen pro přímou podporu více LIDARů v rámci navigačního stacku, který typicky zpracovává data pouze z jednoho senzoru. Aby bylo možné využít oba skeny pro navigaci a mapování, je nutné vytvořit virtuální LIDAR, jehož souřadný systém je umístěn v počátku souřadné soustavy vozíku. LIDARy poskytují data ve formě cylindrických souřadnic  $(r, \theta)$ , kde  $r$  je vzdálenost překážky a  $\theta$  je úhel relativně k senzoru. Pro účely navigace a mapování je však nutné převést tato data do kartézských souřadnic  $(x, y)$  v globálním rámci vozíku:

$$x = r \cdot \cos(\theta),$$

$$y = r \cdot \sin(\theta).$$

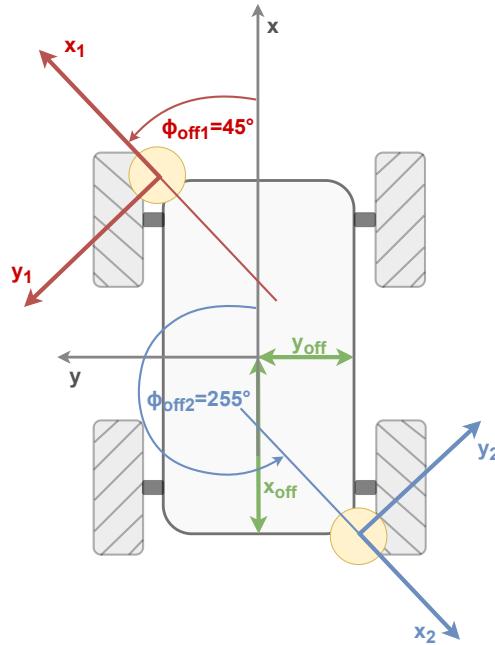
Následně je třeba provést transformaci těchto bodů z lokálních souřadnic LIDARů do globálního rámce vozíku. Transformace zahrnuje rotaci a translaci, které jsou definovány transformačními maticemi v rámci *tf2* knihovny ROS2 (viz sekce 3.4). Pokud je poloha LIDARu vůči vozíku definována jako posun  $(x_{offi}, y_{offi})$  a rotace  $\phi_{offi}$ , bod  $(x, y)$  v lokálním rámci LIDARu je transformován do globálního rámce vozíku podle vzorce:

$$x' = x_i \cdot \cos(\phi_{offi}) - y_i \cdot \sin(\phi_{offi}) + x_{offi},$$

$$y' = x_i \cdot \sin(\phi_{offi}) + y_i \cdot \cos(\phi_{offi}) + y_{offi},$$

kde  $(x', y')$  jsou nové souřadnice bodu v globálním rámci.

Tato transformace je implementována v reálném čase, protože LIDARy publikují data každých 25 ms (u RSL200) nebo 40 ms (u RSL400). Aby bylo zpracování co nejrychlejší, byly použity předpočítané tabulky pro rotace a translace, což minimalizuje výpočetní zátěž a snižuje latenci. Výsledná data jsou publikována jako *LaserScan*, což je datový typ optimalizovaný pro zpracování LIDARových skenů v ROS2.



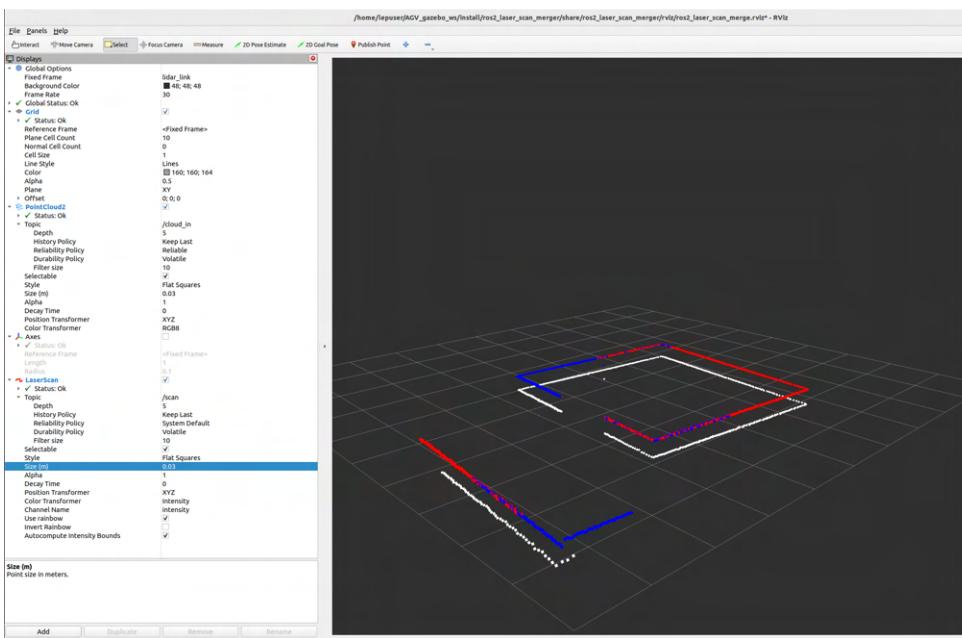
Obrázek 4.11: Sjednocení dvou LIDARových skenů do virtuálního LIDARu prostřednictvím transformace souřadnic.

### 4.2.3 Zpracování překrývajících se dat

V oblastech, kde se skeny z obou LIDARů překrývají (viz obrázek 4.10), je nutné vybrat hodnotu, která je statisticky věrohodnější. Kritéria pro výběr zahrnují eliminaci outlierů, jako jsou body s příliš malou intenzitou, což může naznačovat chyby měření (např. roztríštěný odraz na hraně nebo vícenásobný odraz). Intenzita je dodatečná informace poskytovaná LIDARem, která vyjadřuje sílu odrazu laserového paprsku – plný odraz indikuje pevnou překážku, zatímco slabá intenzita může signalizovat šum, odraz od prachu, hmyzu nebo nepřesné měření. Tento filtr pomáhá zlepšit přesnost virtuálního skenu.

### 4.2.4 Vizualizace a výsledky

Výsledný virtuální sken je znázorněn na obrázku 4.12, kde je zobrazen v nástroji RViz (viz sekce 3.9). Bílá data představují finální *LaserScan* virtuálního LIDARu, zatímco červená a modrá data ukazují původní skeny z jednotlivých LIDARů, což umožňuje identifikovat překrývající se oblasti. Transformace posouvá data do správné polohy a orientace vzhledem k počátku souřadné soustavy vozíku, což zajišťuje konzistentní zpracování v navigačním zásobníku. Publikace dat ve formátu *LaserScan* přináší výhody, jako je kompatibilita s existujícími ROS2 balíčky, které jsou optimalizovány pro tento datový typ.



Obrázek 4.12: Vizualizace spojení LIDARových skenů v RViz: červená a modrá data představují původní skeny, bílá data výsledný virtuální *LaserScan*.

## 4.3 Prediktivní řízení

Prediktivní řízení, známé také jako Model Predictive Control (MPC), představuje pokročilou metodu řízení dynamických systémů, která kombinuje predikci budoucího chování systému s optimalizací řídicích akcí při respektování předem definovaných omezení. Tato technika, původně vyvinutá pro potřeby průmyslových procesů, jako jsou chemické závody nebo rafinerie ropy, nachází v současnosti široké uplatnění i v moderních aplikacích, například v energetických systémech, autonomní robotice či sledování trajektorií, což je rovněž předmětem této práce. Klíčovým principem MPC je opakování řešení optimalizačního problému na základě matematického modelu systému, přičemž se v každém časovém kroku určuje optimální posloupnost řídicích akcí a implementuje pouze její první prvek. V této práci však MPC jako takové není použito, ale je zde odvozeno, protože z něj vychází algoritmus MPPI (Model Predictive Path Integral), který byl pro regulaci po trajektorii a lokální plánování zvolen jako vhodnější alternativa. Původně bylo plánováno využití MPC, ale MPPI se ukázal jako efektivnější, protože je méně výpočetně náročný než MPC – nevyžaduje přímé řešení složitých optimalizačních problémů, ale využívá stochastickou optimalizaci. Navíc MPPI lépe zvládá nelinearity dynamického systému, je robustnější vůči nejistotám v prostředí (např. neočekávaným překážkám), umožňuje rychlejší přizpůsobení dynamickým změnám a jeho implementace v reálném čase je jednodušší.

Obecně lze optimalizační problém prediktivního řízení formulovat následovně: pro daný diskrétní časový okamžik  $k$  a systém popsáný stavovým modelem hledáme posloupnost řídicích vstupů  $\mathbf{u} = [u_k, u_{k+1}, \dots, u_{k+h_c-1}]^T$ , která minimalizuje ztrátovou funkci  $J$ . Tato funkce obvykle zohledňuje odchylku predikovaných výstupů od referenčních hodnot a penalizaci na velikost řídicích akcí. Optimalizační problém je omezen predikčním horizontem  $h_p$ , horizontem řízení  $h_c$  a fyzickými či provozními omezeními systému (např.  $u_{min} \leq u_k \leq u_{max}$ ), přičemž platí, že  $h_c < h_p$ . Matematicky lze obecný tvar ztrátové funkce zapsat jako:

$$J = \sum_{i=k}^{k+h_p-1} (y_i - r_i)^T Q (y_i - r_i) + \sum_{i=k}^{k+h_c-1} u_i^T R u_i,$$

kde  $y_i$  je predikovaný výstup,  $r_i$  referenční hodnota,  $Q \in \mathbb{R}^{p \times p}$  a  $R \in \mathbb{R}^{m \times m}$  jsou váhové matici určující prioritu přesnosti regulace a energetické náročnosti řízení. Optimalizační problém je poté vyřešen v každém kroku a proces se opakuje s posouvajícím se horizontem (tzv. receding horizon control, RHC), což je obecnější koncept, jehož je MPC specifickou realizací založenou na explicitním modelu systému.

Na rozdíl od klasických regulátorů, jako je proporcionalně-integračně-derivační (PID) regulátor, umožňuje MPC explicitně zohlednit budoucí vývoj systému a předem reagovat na predikované události. Díky této prediktivní schopnosti a flexibilitě při zpracování omezení je MPC zvláště vhodné pro složité, vícerozměrné (MIMO) systémy. Většinou je implementováno jako digitální řízení s diskrétním časovým krokem  $T_s$ , ačkoli se zkoumají i možnosti jeho realizace pomocí analogových obvodů pro zvýšení rychlosti výpočtů.

V této kapitole bude podrobněji popsán princip fungování MPC, včetně matematického základu, predikčních modelů a optimalizačních technik, jako je například metoda *move blocking*, která snižuje výpočetní náročnost při zachování kvality regulace.

### 4.3.1 Matematické základy MPC

Algoritmus MPC vychází z modelu řízeného systému. Potřebujeme znát jeho diskrétní stavovou reprezentaci v podobě matic  $A$ ,  $B$ ,  $C$ ,  $D$  a vzorkovací periody  $T_s$ . Model systému musí

být co nejpřesnější, protože i malé odchylky od reality vedou k chybám v predikci, což snižuje kvalitu regulace. Klíčovými parametry jsou predikční horizont  $h_p$  (počet kroků do budoucna, na kterých je prováděna predikce) a horizont řízení  $h_c$  (počet kroků, pro které se optimalizuje řízení).

#### 4.3.1.1 Formulace stavového modelu

Diskrétní systém s  $n$  stavů,  $m$  vstupy a  $p$  výstupy je popsán stavovým modelem

$$\begin{aligned}\mathbf{x}_{k+1} &= A\mathbf{x}_k + B\mathbf{u}_k \\ \mathbf{y}_k &= C\mathbf{x}_k + D\mathbf{u}_k,\end{aligned}\tag{4.53}$$

kde  $\mathbf{x}_k \in \mathbb{R}^{n \times 1}$  je stavový vektor,  $\mathbf{u}_k \in \mathbb{R}^{m \times 1}$  vstupní vektor,  $\mathbf{y}_k \in \mathbb{R}^{p \times 1}$  výstupní vektor,  $A \in \mathbb{R}^{n \times n}$  matice dynamiky,  $B \in \mathbb{R}^{n \times m}$  vstupní matice,  $C \in \mathbb{R}^{p \times n}$  výstupní matice a  $D \in \mathbb{R}^{p \times m}$  průchodová matice.

#### 4.3.1.2 Predikce stavů a výstupů

Predikce chování systému vychází z modelu (4.53). Budoucí stav a výstupy lze rozvinout například takto:

$$\begin{aligned}\mathbf{x}_{k+2} &= A\mathbf{x}_{k+1} + B\mathbf{u}_{k+1} = A^2\mathbf{x}_k + AB\mathbf{u}_k + B\mathbf{u}_{k+1} \\ \mathbf{y}_{k+1} &= C\mathbf{x}_{k+1} + D\mathbf{u}_{k+1} = CA\mathbf{x}_k + CB\mathbf{u}_k + D\mathbf{u}_{k+1}\end{aligned}\tag{4.54}$$

Predikce stavů až do kroku  $h_p$  je vyjádřena v maticové formě:

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{x}_{k+2} \\ \vdots \\ \mathbf{x}_{k+h_p} \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{h_p} \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{h_p-1}B & A^{h_p-2}B & \cdots & B \end{bmatrix} \begin{bmatrix} \mathbf{u}_k \\ \mathbf{u}_{k+1} \\ \vdots \\ \mathbf{u}_{k+h_p-1} \end{bmatrix}$$

$$\mathbf{x}_{k+1:h_p} = P_x \mathbf{x}_k + H_x \mathbf{u}_{k:h_p-1}\tag{4.55}$$

kde  $P_x \in \mathbb{R}^{h_p n \times n}$  a  $H_x \in \mathbb{R}^{h_p n \times h_p m}$  jsou predikční matice stavů.

Analogicky pro výstupy:

$$\begin{bmatrix} \mathbf{y}_k \\ \mathbf{y}_{k+1} \\ \vdots \\ \mathbf{y}_{k+h_p-1} \end{bmatrix} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{h_p-1} \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} D & 0 & \cdots & 0 \\ CB & D & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{h_p-2}B & CA^{h_p-3}B & \cdots & D \end{bmatrix} \begin{bmatrix} \mathbf{u}_k \\ \mathbf{u}_{k+1} \\ \vdots \\ \mathbf{u}_{k+h_p-1} \end{bmatrix}$$

$$\mathbf{y}_{k:h_p-1} = P_y \mathbf{x}_k + H_y \mathbf{u}_{k:h_p-1}\tag{4.56}$$

kde  $P_y \in \mathbb{R}^{h_p p \times n}$  a  $H_y \in \mathbb{R}^{h_p p \times h_p m}$  jsou predikční matice výstupů.

### 4.3.1.3 Metoda move blocking

Horizont řízení  $h_c$  je obvykle menší než horizont predikce  $h_p$ , protože největší vliv mají první zásahy (viz obrázek 4.13). Po každém kroku se problém znovu optimalizuje na posunutém horizontu podle principu RHC. Horizont řízení  $h_c$  nesmí přesáhnout  $h_p$ , jinak by predikce nebyla úplná. Čím vyšší jsou  $h_p$  a  $h_c$ , tím lepší může být regulace, ale roste výpočetní náročnost.

Metoda *move blocking* snižuje tuto náročnost tím, že optimalizuje pouze  $h_c$  vstupů a zbývající vstupy až do  $h_p$  zůstávají konstantní:

$$\mathbf{u}_{k:h_p-1} = \begin{bmatrix} \mathbf{u}_k \\ \vdots \\ \mathbf{u}_{k+h_c-1} \\ \vdots \\ \mathbf{u}_{k+h_p-1} \end{bmatrix} = \begin{bmatrix} I \\ \vdots \\ I \\ \vdots \\ I \end{bmatrix} \begin{bmatrix} \mathbf{u}_k \\ \vdots \\ \mathbf{u}_{k+h_c-1} \end{bmatrix}$$

$$\mathbf{u}_{k:h_p-1} = M_b \mathbf{u}_{k:h_c-1} \quad (4.57)$$

kde  $M_b \in \mathbb{R}^{h_p m \times h_c m}$  je transformační matice a  $I \in \mathbb{R}^{m \times m}$  jednotková matice.

Upravené predikční matice pak jsou:

$$\mathbf{x}_{k+1:h_p} = P_x \mathbf{x}_k + H_{b,x} \mathbf{u}_{k:h_c-1}$$

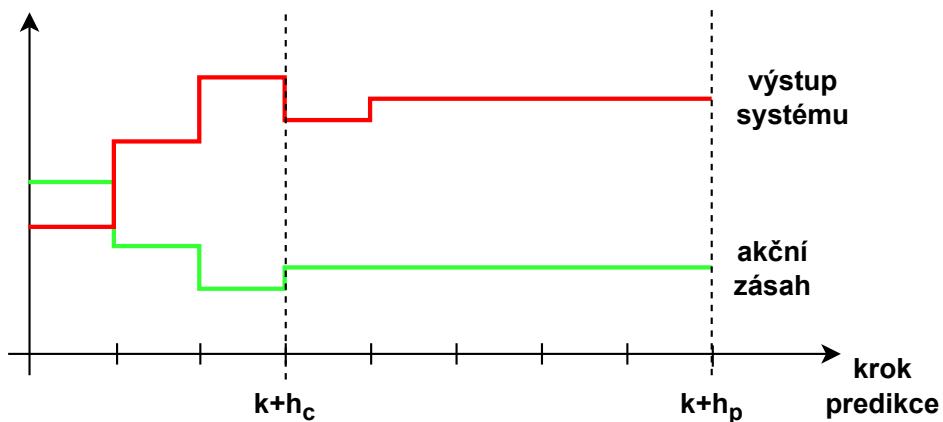
$$\mathbf{y}_{k:h_p-1} = P_y \mathbf{x}_k + H_{b,y} \mathbf{u}_{k:h_c-1} \quad (4.58)$$

kde  $H_{b,x} \in \mathbb{R}^{h_p n \times h_c m}$  a  $H_{b,y} \in \mathbb{R}^{h_p p \times h_c m}$  jsou upravené predikční matice.

Finální upravené vztahy pro predikci dostávají tedy následující tvar:

$$\mathbf{x}_{k+1:h_p} = P_x \mathbf{x}_k + H_{b,x} \mathbf{u}_{k:h_c-1}$$

$$\mathbf{y}_{k:h_p-1} = P_y \mathbf{x}_k + H_{b,y} \mathbf{u}_{k:h_c-1} \quad (4.59)$$



Obrázek 4.13: Zobrazení principu horizontů

### 4.3.2 Formulace a řešení optimalizačního problému

Ztrátová funkce definovaná v úvodu vyjadřuje cíl MPC – minimalizovat odchylku výstupů od referencí a penalizovat nadměrné řídicí zásahy. Pro její praktické použití ji převeďeme do maticové formy. Obecný tvar je:

$$J = \sum_{i=k}^{k+h_p-1} e_i^T Q e_i + \sum_{i=k}^{k+h_c-1} \mathbf{u}_i^T R \mathbf{u}_i, \quad (4.60)$$

kde  $e_i = \mathbf{y}_i - r_i$  je regulační chyba,  $r_i \in \mathbb{R}^{p \times 1}$  referenční hodnota,  $Q \in \mathbb{R}^{p \times p}$  váhová matice pro chyby výstupů a  $R \in \mathbb{R}^{m \times m}$  váhová matice pro vstupy.

#### 4.3.2.1 Regulace do nuly

Pro případ regulace do nuly ( $r_i = 0$ ) a s využitím predikce (4.59) lze  $J$  přepsat:

$$\begin{aligned} J &= \sum_{i=k}^{k+h_p-1} \mathbf{y}_i^T Q \mathbf{y}_i + \sum_{i=k}^{k+h_c-1} \mathbf{u}_i^T R \mathbf{u}_i \\ J &= \mathbf{y}_{k:h_p-1}^T Q_0 \mathbf{y}_{k:h_p-1} + \mathbf{u}_{k:h_c-1}^T R_0 \mathbf{u}_{k:h_c-1} \\ J &= (P_y \mathbf{x}_k + H_{b,y} \mathbf{u}_{k:h_c-1})^T Q_0 (P_y \mathbf{x}_k + H_{b,y} \mathbf{u}_{k:h_c-1}) + \mathbf{u}_{k:h_c-1}^T R_0 \mathbf{u}_{k:h_c-1} \end{aligned} \quad (4.61)$$

kde  $Q_0 = \text{diag}(Q, \dots, Q) \in \mathbb{R}^{h_p p \times h_p p}$  a  $R_0 = \text{diag}(R, \dots, R) \in \mathbb{R}^{h_c m \times h_c m}$  jsou rozšířené váhové matice.

Rozpisem (4.61) získáme kvadratický tvar:

$$\begin{aligned} J &= \mathbf{u}_{k:h_c-1}^T (H_{b,y}^T Q_0 H_{b,y} + R_0) \mathbf{u}_{k:h_c-1} + 2(P_y \mathbf{x}_k)^T Q_0 H_{b,y} \mathbf{u}_{k:h_c-1} + (P_y \mathbf{x}_k)^T Q_0 P_y \mathbf{x}_k \\ J &= \frac{1}{2} \mathbf{u}_{k:h_c-1}^T G \mathbf{u}_{k:h_c-1} + f^T \mathbf{u}_{k:h_c-1} + c \\ G &= 2(H_{b,y}^T Q_0 H_{b,y} + R_0) \\ f &= 2H_{b,y}^T Q_0 P_y \mathbf{x}_k \\ c &= (P_y \mathbf{x}_k)^T Q_0 P_y \mathbf{x}_k \end{aligned} \quad (4.62)$$

kde  $G \in \mathbb{R}^{h_c m \times h_c m}$  je Hessova matice,  $f \in \mathbb{R}^{h_c m \times 1}$  lineární člen a  $c$  konstanta nezávislá na  $\mathbf{u}$ .

Optimalizační problém tedy má finální tvar s přidanými vazebními podmínkami, které zajišťují soulad s dynamikou systému a fyzickými limity:

$$\min_{\mathbf{u}_{k:h_c-1}} \frac{1}{2} \mathbf{u}_{k:h_c-1}^T G \mathbf{u}_{k:h_c-1} + f^T \mathbf{u}_{k:h_c-1} \quad (4.63)$$

podmíněno:

$$\begin{aligned} \mathbf{x}_{k+1:h_p} &= P_x \mathbf{x}_k + H_{b,x} \mathbf{u}_{k:h_c-1}, \\ \mathbf{y}_{k:h_p-1} &= P_y \mathbf{x}_k + H_{b,y} \mathbf{u}_{k:h_c-1}, \\ \mathbf{u}_{\min} \leq \mathbf{u}_{k+i} &\leq \mathbf{u}_{\max}, \quad i = 0, 1, \dots, h_c - 1, \\ \mathbf{x}_{\min} \leq \mathbf{x}_{k+i} &\leq \mathbf{x}_{\max}, \quad i = 1, 2, \dots, h_p, \end{aligned} \quad (4.64)$$

kde  $\mathbf{u}_{\min}, \mathbf{u}_{\max} \in \mathbb{R}^{m \times 1}$  jsou minimální a maximální povolené hodnoty vstupů (např. omezení rychlosti motorů), a  $\mathbf{x}_{\min}, \mathbf{x}_{\max} \in \mathbb{R}^{n \times 1}$  jsou limity stavů (např. maximální poloha nebo rychlosť robota). První dvě podmínky zajišťují konzistenci s dynamickým modelem systému podle (4.59), zatímco nerovnosti omezují vstupy a stavы v souladu s fyzickými vlastnostmi AGV.

#### 4.3.2.2 Obecné řízení a sledování trajektorie

Pro složitější případ sledování trajektorie, kdy referenční signál  $r_i \neq 0$  odpovídá požadované dráze robota, je nutné zohlednit dynamicky se měnící referenční hodnoty  $\mathbf{r}_{k:h_p-1} = [r_k, r_{k+1}, \dots, r_{k+h_p-1}]^T$ . Ztrátová funkce zůstává ve tvaru (4.60), ale nyní zahrnuje nenulovou regulační chybu  $e_i = \mathbf{y}_i - r_i$ . Substitucí predikce výstupů z (4.59) do ztrátové funkce získáme:

$$\begin{aligned} J &= \sum_{i=k}^{k+h_p-1} (\mathbf{y}_i - r_i)^T Q (\mathbf{y}_i - r_i) + \sum_{i=k}^{k+h_c-1} \mathbf{u}_i^T R \mathbf{u}_i \\ J &= (\mathbf{y}_{k:h_p-1} - \mathbf{r}_{k:h_p-1})^T Q_0 (\mathbf{y}_{k:h_p-1} - \mathbf{r}_{k:h_p-1}) + \mathbf{u}_{k:h_c-1}^T R_0 \mathbf{u}_{k:h_c-1} \\ J &= (P_y \mathbf{x}_k + H_{b,y} \mathbf{u}_{k:h_c-1} - \mathbf{r}_{k:h_p-1})^T Q_0 (P_y \mathbf{x}_k + H_{b,y} \mathbf{u}_{k:h_c-1} - \mathbf{r}_{k:h_p-1}) + \mathbf{u}_{k:h_c-1}^T R_0 \mathbf{u}_{k:h_c-1} \\ J &= \mathbf{u}_{k:h_c-1}^T (H_{b,y}^T Q_0 H_{b,y} + R_0) \mathbf{u}_{k:h_c-1} + 2(P_y \mathbf{x}_k - \mathbf{r}_{k:h_p-1})^T Q_0 H_{b,y} \mathbf{u}_{k:h_c-1} + \\ &\quad + (P_y \mathbf{x}_k - \mathbf{r}_{k:h_p-1})^T Q_0 (P_y \mathbf{x}_k - \mathbf{r}_{k:h_p-1}) \end{aligned} \quad (4.65)$$

kde  $\mathbf{r}_{k:h_p-1} \in \mathbb{R}^{h_p p \times 1}$  je vektor referenčních hodnot v predikčním horizontu.

Rozpisem (4.65) převedeme ztrátovou funkci do kvadratického tvaru:

$$\begin{aligned} J &= \frac{1}{2} \mathbf{u}_{k:h_c-1}^T G \mathbf{u}_{k:h_c-1} + f^T \mathbf{u}_{k:h_c-1} + c \\ G &= 2(H_{b,y}^T Q_0 H_{b,y} + R_0) \\ f &= 2H_{b,y}^T Q_0 (P_y \mathbf{x}_k - \mathbf{r}_{k:h_p-1}) \\ c &= (P_y \mathbf{x}_k - \mathbf{r}_{k:h_p-1})^T Q_0 (P_y \mathbf{x}_k - \mathbf{r}_{k:h_p-1}) \end{aligned} \quad (4.66)$$

kde  $G \in \mathbb{R}^{h_c m \times h_c m}$  zůstává stejně jako u regulace do nuly, ale  $f \in \mathbb{R}^{h_c m \times 1}$  a  $c$  zohledňuje referenční signál.

Finální optimalizační problém pro sledování trajektorie zahrnuje vazební podmínky, které zajišťují, že řešení respektuje dynamiku systému a fyzická omezení robota:

$$\min_{\mathbf{u}_{k:h_c-1}} \frac{1}{2} \mathbf{u}_{k:h_c-1}^T G \mathbf{u}_{k:h_c-1} + f^T \mathbf{u}_{k:h_c-1}$$

podmíněno:

$$\begin{aligned} \mathbf{x}_{k+1:h_p} &= P_x \mathbf{x}_k + H_{b,x} \mathbf{u}_{k:h_c-1}, \\ \mathbf{y}_{k:h_p-1} &= P_y \mathbf{x}_k + H_{b,y} \mathbf{u}_{k:h_c-1}, \\ \mathbf{u}_{\min} \leq \mathbf{u}_{k+i} &\leq \mathbf{u}_{\max}, \quad i = 0, 1, \dots, h_c - 1, \\ \mathbf{x}_{\min} \leq \mathbf{x}_{k+i} &\leq \mathbf{x}_{\max}, \quad i = 1, 2, \dots, h_p, \\ \mathbf{y}_{\min} \leq \mathbf{y}_{k+i} &\leq \mathbf{y}_{\max}, \quad i = 0, 1, \dots, h_p - 1, \end{aligned} \tag{4.67}$$

kde  $\mathbf{u}_{\min}, \mathbf{u}_{\max} \in \mathbb{R}^{m \times 1}$  jsou limity vstupů (např. maximální rychlosť nebo zrychlení motorů),  $\mathbf{x}_{\min}, \mathbf{x}_{\max} \in \mathbb{R}^{n \times 1}$  omezují stavy (např. polohu nebo orientaci robota) a  $\mathbf{y}_{\min}, \mathbf{y}_{\max} \in \mathbb{R}^{p \times 1}$  zajišťují, že výstupy (např. skutečná trajektorie) zůstávají v blízkosti referenční dráhy v povolených mezích. První dvě rovnice vynucují dynamiku systému podle (4.59), zatímco nerovnosti zaručují fyzickou realizovatelnost a bezpečnost při sledování trajektorie.

Tento tvar umožňuje robotovi sledovat dynamickou trajektorii při respektování omezení. Výsledné  $\mathbf{u}_{k:h_c-1}$  je vypočteno v každém kroku, ale pouze  $\mathbf{u}_k$  je aplikováno, což odpovídá principu RHC. Tento přístup zajišťuje efektivní regulaci při zachování stability, i když nemusí plně kompenzovat poruchy či nepřesnosti modelu.

### 4.3.3 Odhad stavů pomocí klouzavého horizontu (MHE)

Metoda Moving Horizon Estimation (MHE), v češtině označovaná jako odhad na klouzavém horizontu, je pokročilá optimalizační technika pro rekonstrukci stavů dynamických systémů na základě omezené časové posloupnosti měření a matematického modelu. V kontextu této sekce, zaměřené na sledování trajektorie robotem pomocí (MPC), hraje MHE klíčovou roli jako doplňková metoda zajišťující přesné odhady stavů systému, například polohy a rychlosti robota. Tyto odhady jsou nezbytné pro kvalitní predikci budoucího chování systému v MPC, zejména v přítomnosti šumu, nelinearit nebo nepřesnosti modelu. Na rozdíl od tradičních rekurzivních metod, jako je Kalmanův filtr, který odhaduje stavy pouze na základě minulých a aktuálních měření, MHE působí jako vyhlazovač – v analogii ke Kalmanovu filtru podobně jako Rauch-Tung-Striebel (RTS) – tím, že využívá měření z celého klouzavého horizontu k rekonstrukci stavů, což vede k hladšímu a přesnějšímu odhadu.

Obecně lze optimalizační problém MHE formulovat jako minimalizaci nákladové funkce  $J$ , která zohledňuje chybu mezi měřeními a predikcí modelu, penalizaci počátečních odhadů a případně odchylky vstupů. Matematicky má tvar:

$$J = \|\mathbf{x}_{k-p} - \hat{\mathbf{x}}_{k-p}\|_{P^{-1}}^2 + \sum_{i=k-p}^k \|\mathbf{y}_i - h(\mathbf{x}_i)\|_{R^{-1}}^2 + \sum_{i=k-p}^{k-1} \|\mathbf{w}_i\|_{Q^{-1}}^2,$$

kde  $k$  je aktuální časový krok,  $p$  je délka klouzavého horizontu,  $\mathbf{y}_i$  jsou měření,  $h(\mathbf{x}_i)$  je model výstupu,  $\mathbf{x}_{k-p}$  je odhadovaný počáteční stav na začátku horizontu,  $\hat{\mathbf{x}}_{k-p}$  je předchozí odhad tohoto stavu,  $\mathbf{w}_i$  je šum procesu, a  $R$ ,  $P$ ,  $Q$  jsou váhové matice (kovarianční matice šumu měření, počátečního odhadu a šumu procesu). Tento problém je řešen v každém kroku, přičemž se horizont posouvá a poskytuje odhad  $\mathbf{x}_k$  pro aktuální čas.

#### 4.3.3.1 Formulace modelu a měření

MHE vychází z dynamického modelu stochastického systému v diskrétní formě, který popisuje vývoj stavů a vztah k měřením. Model má obecně tvar:

$$\begin{aligned} \mathbf{x}_{i+1} &= f(\mathbf{x}_i, \mathbf{u}_i) + \mathbf{w}_i \\ \mathbf{y}_i &= h(\mathbf{x}_i) + \mathbf{v}_i \end{aligned} \tag{4.68}$$

kde  $\mathbf{x}_i \in \mathbb{R}^{n \times 1}$  je stavový vektor,  $\mathbf{u}_i \in \mathbb{R}^{m \times 1}$  vstupní vektor,  $\mathbf{y}_i \in \mathbb{R}^{p \times 1}$  měřený výstup,  $f(\cdot)$  je funkce dynamiky systému,  $h(\cdot)$  je měřicí funkce,  $\mathbf{w}_i \in \mathbb{R}^{n \times 1}$  je šum procesu a  $\mathbf{v}_i \in \mathbb{R}^{p \times 1}$  šum měření.

Předpokládá se, že šumy  $\mathbf{w}_i$  a  $\mathbf{v}_i$  jsou nezávislé bílé šumy s kovariančními maticemi  $Q$  a  $R$ . Pro lineární systémy lze model zjednodušit na:

$$\begin{aligned} \mathbf{x}_{i+1} &= A\mathbf{x}_i + B\mathbf{u}_i + \mathbf{w}_i \\ \mathbf{y}_i &= C\mathbf{x}_i + \mathbf{v}_i \end{aligned} \tag{4.69}$$

kde  $A$ ,  $B$ ,  $C$  jsou matice systému odpovídající těm v MPC (viz rovnice (4.53)), přičemž  $D = 0$  pro zjednodušení.

Tento model slouží k predikci stavů v rámci klouzavého horizontu  $[k-p, k]$  na základě vstupů  $\mathbf{u}_{k-p:k-1}$  a odhadovaných stavů  $\mathbf{x}_{k-p:k}$ .

### 4.3.3.2 Optimalizační problém MHE

Optimalizační problém MHE minimalizuje nákladovou funkci  $J$  vzhledem k posloupnosti stavů  $\mathbf{x}_{k-p:k}$  a šumu  $\mathbf{w}_{k-p:k-1}$ , přičemž respektuje dynamiku systému jako omezení. Pro lineární systém lze problém zapsat jako:

$$J = \|\mathbf{x}_{k-p} - \hat{\mathbf{x}}_{k-p}\|_{P^{-1}}^2 + \sum_{i=k-p}^k \|\mathbf{y}_i - C\mathbf{x}_i\|_{R^{-1}}^2 + \sum_{i=k-p}^{k-1} \|\mathbf{w}_i\|_{Q^{-1}}^2 \quad (4.70)$$

při splnění omezení:

$$\mathbf{x}_{i+1} = A\mathbf{x}_i + B\mathbf{u}_i + \mathbf{w}_i, \quad i = k-p, \dots, k-1. \quad (4.71)$$

Přičemž:

- $\|\mathbf{z}\|_M^2 = \mathbf{z}^T M \mathbf{z}$  je vážená norma
- První člen (arrival cost) zajišťuje konzistenci s předchozím odhadem  $\hat{\mathbf{x}}_{k-p}$  váženou maticí  $P^{-1}$ ,
- Druhý člen penalizuje odchylku mezi měřeními  $\mathbf{y}_i$  a predikcí  $C\mathbf{x}_i$ ,
- Třetí člen penalizuje šum procesu  $\mathbf{w}_i$ , což zlepšuje hladkost odhadu.

MHE tak funguje jako vyhlazovač, jak již bylo v úvodu zmíněno, protože odhaduje stavy  $\mathbf{x}_{k-p:k}$  na základě všech měření v horizontu  $[k-p, k]$ , nikoli pouze predikcí z minulých dat. Na rozdíl od RTS, který zpracovává celou historii měření zpětně, MHE omezuje vyhlazování na pevnou délku horizontu  $p$ , což snižuje výpočetní náročnost a činí metodu vhodnou pro online použití.

Pro odvození maticového tvaru eliminujeme  $\mathbf{w}_i$  pomocí dynamiky (4.71):

$$\begin{aligned} \mathbf{x}_{i+1} &= A^{i+1-(k-p)} \mathbf{x}_{k-p} + \sum_{j=k-p}^i A^{i-j} B \mathbf{u}_j + \sum_{j=k-p}^i A^{i-j} \mathbf{w}_j \\ \mathbf{x}_{k-p:k} &= P_x \mathbf{x}_{k-p} + H_u \mathbf{u}_{k-p:k-1} + H_w \mathbf{w}_{k-p:k-1} \\ \mathbf{y}_{k-p:k} &= C_x \mathbf{x}_{k-p:k} + \mathbf{v}_{k-p:k} \end{aligned} \quad (4.72)$$

kde  $P_x = [I, A, \dots, A^p]^T$ ,  $H_u$  a  $H_w$  jsou matice predikce vstupů a šumu,  $C_x = \text{diag}(C, \dots, C)$ , a  $\mathbf{y}_{k-p:k} = [\mathbf{y}_{k-p}, \dots, \mathbf{y}_k]^T$ .

Substitucí do (4.70) a minimalizací vzhledem k  $\mathbf{x}_{k-p}$  a  $\mathbf{w}_{k-p:k-1}$  získáme kvadratický problém, jehož řešením je odhad  $\mathbf{x}_k$ . Po každém kroku se horizont posune a proces se opakuje.

### 4.3.4 Lokální plánovač pomocí Model Predictive Path Integral Control

MPPI je pokročilá varianta Model Predictive Control (MPC), která nahrazuje deterministickou optimalizaci stochastickým přístupem založeným na vzorkování trajektorií. Tato metoda, představená v roce 2015 v publikaci „Model Predictive Path Integral Control Using Covariance Variable Importance Sampling“ [34], je navržena pro řízení nelineárních systémů s nejistotami a šumem, což ji činí zvláště vhodnou pro aplikace, jako je sledování trajektorie robotem v této práci. Na rozdíl od klasického MPC, které řeší kvadratický optimalizační problém (viz sekce 4.3.2), MPPI využívá simulace náhodně perturbovaných vstupů a vážený průměr k výběru optimálního řídicího zásahu. Tento přístup umožňuje robustní řízení i v situacích, kde tradiční metody selhávají kvůli nelinearitám nebo neurčitostem, například při navigaci robota na hranici řiditelnosti.

V kontextu sledování trajektorie robotem MPPI doplňuje odhad stavů z MHE (viz sekce 4.3.3) tím, že zohledňuje šumová měření a nelineární dynamiku systému. Cílem je minimalizovat nákladovou funkci, která vyhodnocuje odchylku od požadované trajektorie a penalizuje nadměrné řídicí zásahy, přičemž využívá stochastické vzorkování k exploraci možných budoucích scénářů. Optimalizační problém MPPI lze formulovat jako výpočet očekávané hodnoty nákladů přes sadu náhodných trajektorií:

$$\mathbf{u}_{0:T-1}^* = \arg \min_{\mathbf{u}_{0:T-1}} \mathbb{E}_\epsilon [J(\mathbf{x}_{0:T}, \mathbf{u}_{0:T-1} + \epsilon_{0:T-1})],$$

kde  $\mathbf{u}_{0:T-1}$  je posloupnost vstupů na horizontu  $T$ ,  $\epsilon_{0:T-1}$  je náhodný šum,  $J(\cdot)$  je nákladová funkce a očekávaná hodnota  $\mathbb{E}_\epsilon$  je approximována vzorkováním. Výsledná posloupnost je aktualizována v každém kroku, přičemž se aplikuje pouze první vstup  $\mathbf{u}_0$ , podobně jako u MPC.

#### 4.3.4.1 Formulace modelu a vstupů

MPPI vychází z nelineárního dynamického modelu systému v diskrétní formě, který se aplikuje na posuvném predikčním horizontu  $h_p$ :

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t + \epsilon_{i,t}), \quad t = k, \dots, k + h_p - 1, \quad (4.73)$$

kde  $\mathbf{x}_t \in \mathbb{R}^{n \times 1}$  je stavový vektor (např. poloha a rychlosť robota),  $\mathbf{u}_t \in \mathbb{R}^{m \times 1}$  je nominální vstup,  $\epsilon_{i,t} \in \mathbb{R}^{m \times 1}$  je náhodný šum pro trajektorii  $i$  vygenerovaný z normálního rozdělení  $\mathcal{N}(0, \Sigma)$ ,  $\Sigma \in \mathbb{R}^{m \times m}$  je kovarianční matice šumu, a  $f(\cdot)$  je nelineární funkce dynamiky systému.  $h_p$  označuje predikční horizont, a  $k$  je aktuální časový krok, od kterého se predikce provádí.

Nominální vstupní sekvence  $U_k = [\mathbf{u}_k, \mathbf{u}_{k+1}, \dots, \mathbf{u}_{k+h_p-1}] \in \mathbb{R}^{m \times h_p}$  je perturbována šumem, čímž vzniká vzorkovaná sekvence  $V_i = [\mathbf{v}_{i,k}, \mathbf{v}_{i,k+1}, \dots, \mathbf{v}_{i,k+h_p-1}]$  pro trajektorii  $i$ , kde:

$$\begin{aligned} \mathbf{v}_{i,t} &= \mathbf{u}_t + \epsilon_{i,t}, \\ \epsilon_{i,t} &\sim \mathcal{N}(0, \Sigma), \quad i = 1, \dots, I, \end{aligned} \quad (4.74)$$

kde  $I$  je celkový počet vzorkovaných trajektorií,  $i$  je index konkrétní trajektorie, a  $\epsilon_{i,t}$  je nezávislý šum pro trajektorii  $i$  v čase  $t$ . Počáteční stav  $\hat{\mathbf{x}}_k$  je odhadnut pomocí Moving Horizon Estimation (MHE) v čase  $k$ .

#### 4.3.4.2 Optimalizační problém MPPI

MPPI minimalizuje očekávané náklady budoucích trajektorií na plovoucím predikčním horizontu  $h_p$ . Nákladová funkce  $J(\mathbf{x}_{k:k+h_p}, V_i)$  pro jednu trajektorii  $i$  v čase  $k$  je definována jako:

$$J(\mathbf{x}_{k:k+h_p}, V_i) = \phi(\mathbf{x}_{k+h_p}) + \sum_{l=k}^{k+h_p-1} c(\mathbf{x}_l, \mathbf{v}_{i,l}),$$

$$\mathbf{x}_{l+1} = f(\mathbf{x}_l, \mathbf{v}_{i,l}), \quad l = k, \dots, k + h_p - 1, \quad \mathbf{x}_k = \hat{\mathbf{x}}_k \text{ z MHE}, \quad (4.75)$$

kde  $\mathbf{x}_{k:k+h_p}$  je predikovaná trajektorie vycházející z odhadu  $\hat{\mathbf{x}}_k$  z MHE a modelu  $f(\cdot)$ ,  $\phi(\mathbf{x}_{k+h_p})$  je terminální náklad (např. odchylka od cílového stavu), a  $c(\mathbf{x}_l, \mathbf{v}_{i,l})$  je běžný náklad (např. odchylka od referenční trajektorie a penalizace vstupu).

Optimalizační problém spočívá v nalezení  $U_k$ , které minimalizuje očekávanou hodnotu nákladů přes všechny perturbace na horizontu  $h_p$  od času  $k$ :

$$U_k^* = \arg \min_{U_k} \mathbb{E}_\epsilon [J(\mathbf{x}_{k:k+h_p}, U_k + \epsilon_{k:k+h_p-1})]. \quad (4.76)$$

Tato očekávaná hodnota je approximována Monte Carlo metodou pomocí  $N$  vzorků:

$$\mathbb{E}_\epsilon [J(\mathbf{x}_{k:k+h_p}, U_k + \epsilon_{k:k+h_p-1})] \approx \frac{1}{N} \sum_{i=1}^N J(\mathbf{x}_{k:k+h_p}, V_i), \quad (4.77)$$

kde  $V_i = U_k + \epsilon_{i,k:k+h_p-1}$ , a  $\epsilon_{i,k:k+h_p-1} = [\epsilon_{i,k}, \dots, \epsilon_{i,k+h_p-1}]$  je sekvence šumu pro trajektorii  $i$ . Optimální vstupy jsou vypočteny pomocí váženého průměru perturbací, kde váha  $w_i$  pro trajektorii  $i$  je odvozena z nákladů  $J(\mathbf{x}_{k:k+h_p}, V_i)$ :

$$w_i = \frac{\exp\left(-\frac{1}{\lambda}(J(\mathbf{x}_{k:k+h_p}, V_i) - J_{\min})\right)}{\sum_{j=1}^N \exp\left(-\frac{1}{\lambda}(J(\mathbf{x}_{k:k+h_p}, V_j) - J_{\min})\right)},$$

$$J_{\min} = \min_i J(\mathbf{x}_{k:k+h_p}, V_i), \quad (4.78)$$

kde  $\lambda > 0$  je teplotní parametr určující rozptyl vah, a  $J_{\min}$  zajišťuje numerickou stabilitu tím, že normalizuje náklady relativně k nejlepší trajektorii. Teplotní parametr  $\lambda$  hraje klíčovou roli při určování vah  $w_k$  a ovlivňuje chování algoritmu MPPI. Jeho název vychází z analogie s termodynamikou, konkrétně z Boltzmannova rozdělení, kde teplota určuje rozložení pravděpodobností mezi stavy s různou energií. V MPPI parametr  $\lambda$  řídí citlivost vah na rozdíly v nákladech  $J(V_k)$  mezi trajektoriami:

- Vysoké  $\lambda$ :** Pokud  $\lambda \rightarrow \infty$ , exponent  $-\frac{1}{\lambda}(J(V_k) - J_{\min})$  se blíží nule, takže  $\exp(\cdot) \rightarrow 1$  a vahy se přibližují rovnoměrnému rozložení  $w_k \approx \frac{1}{K}$ . To znamená, že algoritmus zohledňuje široké spektrum trajektorií, včetně těch s vyššími náklady, což podporuje *exploraci* různých možností.
- Nízké  $\lambda$ :** Pokud  $\lambda \rightarrow 0$ , exponent se stává velmi negativním pro všechny  $J(V_k) > J_{\min}$ , takže vahy  $w_k$  se soustředí na trajektorii s minimálním nákladem ( $w_k \rightarrow 1$  pro nejlepší trajektorii,  $w_k \rightarrow 0$  pro ostatní). Tento přístup preferuje *exploataci* optimálních řešení.

Matematicky tedy  $\lambda$  ovlivňuje rozptyl vah podle vzorce (4.78). Při velkém  $\lambda$  jsou váhy rozloženy rovnoměrněji, což zvyšuje robustnost algoritmu vůči nejistotám (např. šumu měření nebo nelineárním poruchám), ale může snížit přesnost. Naopak malé  $\lambda$  koncentruje váhy na nejlepší trajektorie, což zvyšuje přesnost řízení, ale může vést k přecitlivělosti na lokální optima nebo neočekávané změny. Pro sledování trajektorie robotem je volba  $\lambda$  kompromisem mezi přesností (dodržení referenční dráhy) a robustností (schopností reagovat na prokluzu kol či nepřesná měření z LIDARu). Optimální hodnota  $\lambda$  je obvykle určena experimentálně podle charakteru systému a prostředí.

Původní formulace zahrnuje komplexnější váhy s penalizací vstupů, ale pro zjednodušení a přehlednost zde používáme základní tvar. Aktualizovaná vstupní sekvence na horizontu řízení  $h_c$  je:

$$\mathbf{u}_{k+m}^{j+1} = \mathbf{u}_{k+m}^i + \sum_{i=1}^I w_i \epsilon_{i,k+m}, \quad t = 0, \dots, h_c - 1, \quad (4.79)$$

kde  $\mathbf{u}_{k+m}^j$  je vstup z předchozí iterace  $j$  (zde  $j$  označuje iteraci algoritmu), a  $h_c \leq h_p$  určuje počet optimalizovaných vstupů aplikovaných v každém kroku.  $m$  je relativní časový posun vzhledem k aktuálnímu časovému kroku  $k$  uvnitř horizontu řízení  $h_c$ .  $k$  je aktuální časový krok, od kterého začíná predikce a optimalizace (např. současný okamžik v RHC).  $m$  určuje, o kolik kroků dopředu od  $k$  aktualizujeme nominální vstup  $u$ . Rozsah  $m = 0, \dots, h_c - 1$  znamená, že aktualizujeme vstupy pro prvních  $h_c$  kroků horizontu řízení. V souladu s principem RHC je v této práci  $h_c = 1$ , takže se aplikuje pouze  $\mathbf{u}_k^{j+1}$ , horizont se posune na  $k + 1$ , a proces se opakuje s novým odhadem  $\hat{\mathbf{x}}_{k+1}$  z MHE.

#### 4.3.4.3 Použití pro sledování trajektorie

MPPI využívá odhady stavů z MHE a predikční schopnosti k robustnímu řízení robota při sledování trajektorie. Pro referenční trajektorii  $\mathbf{r}_{k:k+h_p}$  je nákladová funkce definována jako:

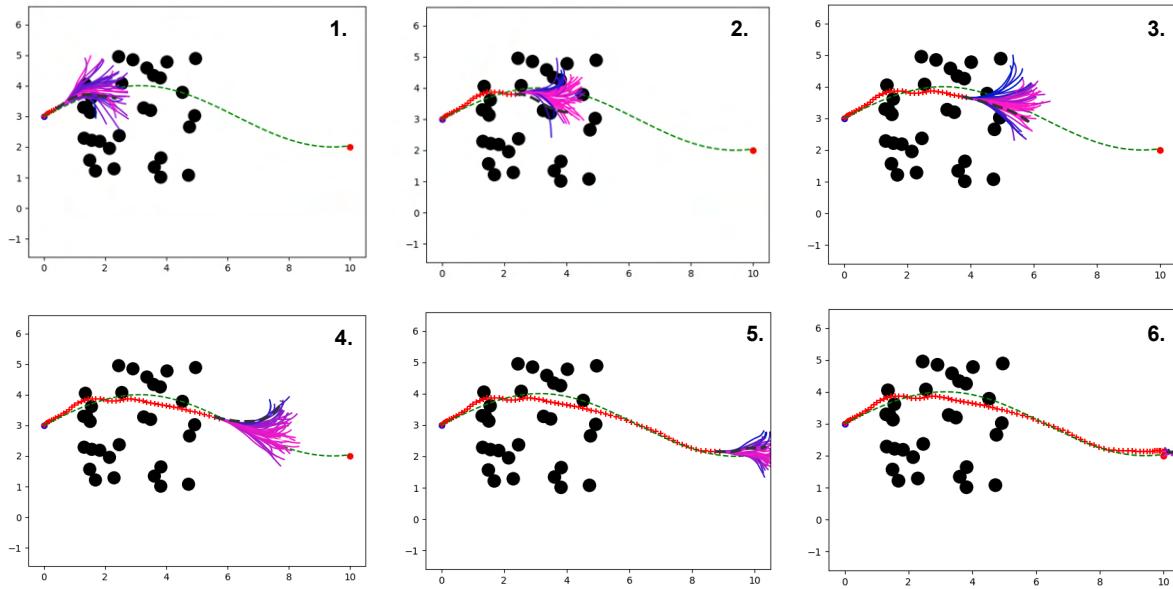
$$J(\mathbf{x}_{k:k+h_p}, V_i) = \left\| \mathbf{x}_{k+h_p} - \mathbf{r}_{k+h_p} \right\|_{Q_{h_p}}^2 + \sum_{l=k}^{k+h_p-1} \left( \left\| \hat{\mathbf{x}}_l - \mathbf{r}_l \right\|_Q^2 + \left\| \mathbf{v}_{i,l} \right\|_R^2 \right), \quad (4.80)$$

kde  $\mathbf{x}_{k:k+h_p}$  je predikovaná trajektorie z modelu (4.73) s počátečním stavem  $\hat{\mathbf{x}}_k$  odhadnutým MHE,  $Q$ ,  $R$ ,  $Q_{h_p}$  jsou váhové matice penalizující odchylku od referenční dráhy a velikost vstupů. Tento přístup umožňuje robotovi reagovat na šum měření (např. z LIDARu) a nelineární dynamiku (např. prokluzu kol), což je výhoda oproti deterministickému MPC. Kombinací MHE pro odhad  $\hat{\mathbf{x}}_k$  a MPPI pro řízení získáváme systém schopný navigace v reálných podmírkách s nejistotami.

#### 4.3.4.4 Simulace a vizualizace

Zkráceně je MPPI pokročilý regulátor, který přesahuje pouhé sledování odchylky od požadované trajektorie. Kromě minimalizace rozdílu mezi aktuální a cílovou polohou optimalizuje dráhu robota s ohledem na různé faktory, jako je vyhýbání se překážkám nebo zkrácení celkové cesty, pokud je to výhodnější. Pro ověření funkčnosti algoritmu byla vytvořena simulace, která modeluje pohyb vozíku v prostředí s náhodně rozmístěnými překážkami a definovanou

trajektorií ve formě sinusové křivky. Simulace využívá model vozidla s dynamickými omezeními (např. maximální rychlostí a řízením) a generuje řadu možných trajektorií, z nichž MPPI vybírá optimální cestu. Vývoj simulace je zachycen na obrázku 4.14, který obsahuje snímky ukazující časový průběh pohybu vozíku: počáteční pozici (modrý bod), cílovou pozici (červený bod), referenční trajektorii (zelená čára), překážky (černé kruhy) a generované trajektorie (barevné čáry), kde nejoptimálnější trajektorie je zvýrazněna černou přerušovanou čarou. Poloha vozíku v jednotlivých časových okamžicích simulace je znázorněna červenými křížky. Simulace je dostupná na GitHubu [35] a její průběh je ukazuje Video 2 [2].



Obrázek 4.14: Znázornění simulace MPPI

[2]↑ <https://youtu.be/m8dPczKWj1Y>

## 4.4 Pure Pursuit algoritmus

Pure Pursuit je geometrický algoritmus pro řízení mobilních robotů, jehož účelem je zajistit hladké a přesné sledování předem definované trajektorie. Tento algoritmus je široce využíván v robotice díky své jednoduchosti a schopnosti generovat kinematicky proveditelné pohybové dráhy. V této práci je aplikován na diferenciální podvozek sledující cyklickou trajektorii  $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$  v globálním souřadném systému  $(X, Y)$ , kde  $(x_{n+1}, y_{n+1}) = (x_1, y_1)$ . Poloha robota je definována jako  $(x_r, y_r, \theta_r)$ , kde  $\theta_r$  je orientace vůči ose  $X$ .

Cílem kapitoly je odvodit matematický základ algoritmu, včetně geometrického principu a výpočtu řídicích vstupů – lineární rychlosti  $v$  a úhlové rychlosti  $\omega$ .

### 4.4.1 Geometrický princip algoritmu

Základním principem Pure Pursuit je, že robot sleduje pohyblivý cílový bod (look-ahead point)  $(x_g, y_g)$  na trajektorii, který je vybrán na základě vzdálenosti  $L$  od aktuální polohy robota  $(x_r, y_r)$ . Robot se k tomuto bodu přibližuje po kruhovém oblouku o poloměru  $R$ , jehož hodnota je určena geometrickými vztahy mezi  $(x_r, y_r)$ ,  $(x_g, y_g)$  a orientací  $\theta_r$ .

Vzdálenost mezi robotem a cílovým bodem, označovaná jako  $l_d$ , je definována jako:

$$l_d = \sqrt{(x_g - x_r)^2 + (y_g - y_r)^2} \quad (4.81)$$

Parametr  $L$ , nazývaný look-ahead distance, je klíčovým faktorem ovlivňujícím chování algoritmu: větší hodnota  $L$  vede k hladšímu, ale méně přesnému sledování, zatímco menší  $L$  zvyšuje přesnost, avšak může způsobit oscilace.

Směrový vektor od robota k cílovému bodu je:

$$\vec{d} = (x_g - x_r, y_g - y_r) \quad (4.82)$$

a směr pohybu robota je dán jednotkovým vektorem  $(\cos \theta_r, \sin \theta_r)$ . Úhel  $\alpha$  mezi těmito vektory určuje zakřivení oblouku a je vypočten pomocí skalárního součinu:

$$\cos \alpha = \frac{\vec{d} \cdot (\cos \theta_r, \sin \theta_r)}{l_d} = \frac{(x_g - x_r) \cos \theta_r + (y_g - y_r) \sin \theta_r}{l_d} \quad (4.83)$$

Pro určení směru otáčení je třeba znát  $\sin \alpha$ , který lze odvodit pomocí vektorového součinu:

$$\sin \alpha = \frac{(x_g - x_r) \sin \theta_r - (y_g - y_r) \cos \theta_r}{l_d} \quad (4.84)$$

Znaménko  $\sin \alpha$  určuje, zda se robot otáčí doprava (kladné) nebo doleva (záporné).

### 4.4.2 Odvození poloměru otáčení

Poloměr otáčení  $R$  je odvozen z geometrie kružnice procházející body  $(x_r, y_r)$  a  $(x_g, y_g)$ , jejíž střed  $O$  leží na přímce kolmé na směr robota a zároveň na střední kolmici tětivy  $l_d$ . V trojúhelníku  $O - (x_r, y_r) - (x_g, y_g)$  je  $l_d$  tětiva subtendující centrální úhel  $2\alpha$ . Z trigonometrického vztahu v kružnici platí:

$$l_d = 2R \sin \alpha \quad (4.85)$$

Odtud:

$$R = \frac{l_d}{2 \sin \alpha} \quad (4.86)$$

Tento vztah je klíčový pro Pure Pursuit, protože  $R$  přímo určuje zakřivení dráhy. Pokud  $\alpha \rightarrow 0^\circ$  (směr robota je zarovnán s  $\vec{d}$ ), pak  $R \rightarrow \infty$ , což odpovídá přímému pohybu. Naopak při  $|\alpha| \rightarrow 90^\circ$  dosahuje  $R$  minimální hodnoty, což znamená ostré zatáčení.

#### 4.4.3 Výpočet řídicích vstupů

Pro diferenciální podvozek jsou řídicími vstupy lineární rychlosť  $v$  a úhlová rychlosť  $\omega$ . Sice bylo  $v$  původně předpokládáno jako konstantní, pro zvýšení stability a přizpůsobení prudkým zatáčkám byla implementována dynamická úprava  $v$  v závislosti na úhlu  $\alpha$ . Tento přístup zajistuje, že při větších hodnotách  $|\alpha|$  (ostřejších zatáčkách) se rychlosť snižuje, čímž se předchází překročení kinematických omezení robota. Je tím tedy docíleno přesnějšího sledování trajektorie. Rovněž toto řešení ve finále umožňuje rychlejší jízdu při minimální ztrátě přesnosti.

Dynamická lineární rychlosť  $v$  je definována jako:

$$v = \begin{cases} v_{\min}, & \text{pokud } |\alpha| \geq \frac{\pi}{2}, \\ v_{\max} \left(1 - \frac{|\alpha|}{\frac{\pi}{2}}\right), & \text{pokud } |\alpha| < \frac{\pi}{2}, \end{cases} \quad (4.87)$$

kde  $v_{\max}$  je maximální povolená rychlosť,  $v_{\min}$  je minimální rychlosť a  $\alpha$  je úhel v radiánech vypočtený dříve (viz rovnice (4.83) a (4.84)). Při  $|\alpha| = 0$  (přímý pohyb) je  $v = v_{\max}$ , zatímco při  $|\alpha| = \frac{\pi}{2}$  ( $90^\circ$ ) klesne na  $v_{\min}$ , s lineární interpolací mezi těmito extrémy.

Úhlová rychlosť  $\omega$  je poté vypočtena s ohledem na dynamické  $v$ :

$$\omega = \frac{2v \sin \alpha}{l_d} \quad (4.88)$$

Aby nedošlo k překročení maximální úhlové rychlosti  $\omega_{\max}$  diferenciálního podvozku, je  $\omega$  omezena:

$$\omega = \begin{cases} \omega_{\max} \cdot \operatorname{sgn}(\omega), & \text{pokud } |\omega| > \omega_{\max}, \\ \omega, & \text{jinak,} \end{cases} \quad (4.89)$$

kde  $\operatorname{sgn}(\omega)$  je znaménková funkce určující směr otáčení (+1 pro doprava, -1 pro doleva).

Rychlosti kol  $v_R$  a  $v_L$  jsou následně odvozeny z kinematického modelu diferenciálního podvozku, stejně tak je možno změnit kinematický výpočet pro podvozek s mecanum koly (viz sekce 4.1):

$$v_R = v + \frac{\omega l}{2}, \quad v_L = v - \frac{\omega l}{2}, \quad (4.90)$$

kde  $l$  je rozchod mezi koly.

Dynamická úprava  $v$  a omezení  $\omega$  zajišťují, že pohyb robota zůstává plynulý a v souladu s jeho fyzickými limity, zejména při náročných manévrech.

#### 4.4.4 Implementace algoritmu

Algoritmus Pure Pursuit pro diferenciální podvozek běží v iterativním cyklu:

1. **Lokalizace:** Určí se aktuální poloha a orientace robota  $(x_r, y_r, \theta_r)$  pomocí odometrie nebo lokalizačního systému.
2. **Výběr cílového bodu:** Na trajektorii  $T$  se vybere bod  $(x_g, y_g)$ , jehož vzdálenost  $l_d$  je nejbližší hodnotě  $L$ , případně první bod za  $L$  od posledního cíle.
3. **Výpočet geometrických parametrů:**

- $l_d = \sqrt{(x_g - x_r)^2 + (y_g - y_r)^2}$ ,
- $\sin \alpha = \frac{(x_g - x_r) \sin \theta_r - (y_g - y_r) \cos \theta_r}{l_d}$ ,
- $R = \frac{l_d}{2 \sin \alpha}$ .

4. **Výpočet řídicích vstupů:**

- Dynamická lineární rychlosť:

$$v = \begin{cases} v_{\min}, & \text{pokud } |\alpha| \geq \frac{\pi}{2}, \\ v_{\max} \left(1 - \frac{|\alpha|}{\frac{\pi}{2}}\right), & \text{jinak,} \end{cases}$$

- Úhlová rychlosť:  $\omega = \frac{2v \sin \alpha}{l_d}$ ,

- Omezení:

$$\omega = \begin{cases} \omega_{\max} \cdot \operatorname{sgn}(\omega), & \text{pokud } |\omega| > \omega_{\max}, \\ \omega, & \text{jinak,} \end{cases}$$

- Rychlosti kol:  $v_R = v + \frac{\omega l}{2}$ ,  $v_L = v - \frac{\omega l}{2}$ .

5. **Aplikace pohybu:** Rychlosti  $v_R$  a  $v_L$  jsou předány motorům podvozku.

6. **Cyklus:** Po aktualizaci polohy robota se proces opakuje s novým cílovým bodem.

Pro cyklickou trajektorii je po dosažení bodu  $(x_n, y_n)$  dalším cílem opět  $(x_1, y_1)$ , což zajišťuje nepřetržité sledování.

#### 4.4.5 Příklad aplikace

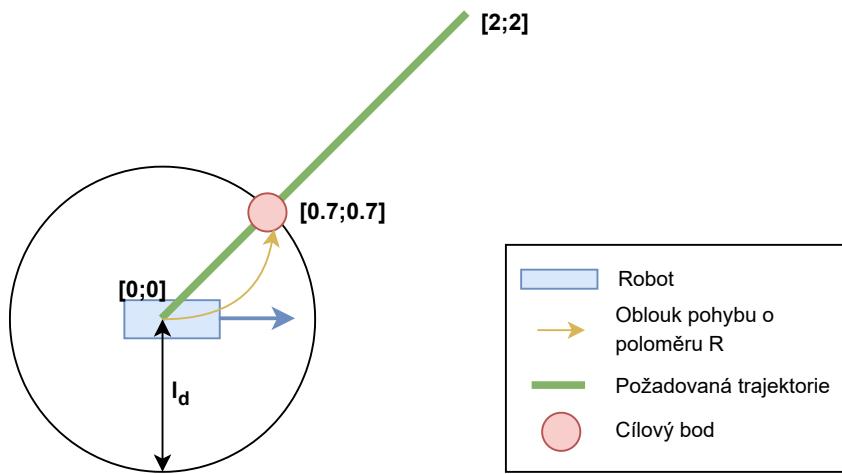
Předpokládejme trajektorii  $T = \{(0, 0), (2, 2), (0, 0)\}$ ,  $v_{\max} = 0.5 \text{ m/s}$ ,  $v_{\min} = 0.1 \text{ m/s}$ ,  $\omega_{\max} = 1.0 \text{ rad/s}$ ,  $L = 1 \text{ m}$  a počáteční polohu robota  $(0, 0, 0^\circ)$ :

- Cílový bod:  $(0.7, 0.7)$ ,  $l_d \approx 1 \text{ m}$ ,
- $\sin \alpha = -0.7$ ,  $|\alpha| = \arcsin(0.7) \approx 0.775 \text{ rad} \approx 44.4^\circ$ ,

- $v = 0.5 \left(1 - \frac{0.775}{\frac{\pi}{2}}\right) \approx 0.5 \cdot (1 - 0.494) \approx 0.253 \text{ m/s}$ ,
- $\omega = \frac{2 \cdot 0.253 \cdot (-0.7)}{1} \approx -0.354 \text{ rad/s}$  (doleva,  $|\omega| < 1.0$ , tedy bez omezení).

Robot tedy jede rychlosť  $v \approx 0.253 \text{ m/s}$  a otáčí se doleva s  $\omega \approx -0.354 \text{ rad/s}$ . V dalším kroku se posune cílový bod díky lookahead vzdálenosti a je vypočten nový vstup. Tento postup je naznačen i na obrázku 4.15

Pure Pursuit vyniká jednoduchostí a hladkostí generovaných drah díky obloukovému pohybu. Jeho efektivita závisí na správné volbě  $L$ : příliš velké  $L$  může vést k „řezání rohů“, zatímco příliš malé  $L$  způsobuje oscilace. Algoritmus je ideální pro statické trajektorie, pro příliš dynamické prostředí může být problematické časté měnění trajektorie. Implementace tohoto algoritmu je přiložena k práci v ROS2 balíčku s připravenou simulací v Gazebo pro testování, podrobněji je tato funkcionality popsána v závěrečných kapitolách.



Obrázek 4.15: Znázornění příkladu funkce Pure Pursuit algoritmu

## 4.5 Detekce malých objektů z 2D LIDARových skenů

Detekce objektů z 2D LIDARových skenů představuje zajímavou výzvu, zejména při zpracování dat při autonomní navigaci a manipulaci. Tato sekce se zaměřuje na teoretické základy metod pro identifikaci malých objektů, jako jsou nohy palety, na základě dat z LIDARu. Ačkoliv praktická aplikace detekce palet bude podrobněji rozpracována v následujících částech (viz sekce 5.5.2), zde je kláden důraz na obecný přístup k analýze dat, včetně jejich reprezentace a shlukování. Sekce porovnává různé techniky, jako je zpracování ztrátových map pomocí OpenCV a shlukovací algoritmy, a hodnotí jejich vhodnost pro daný účel s ohledem na efektivitu a robustnost.

### 4.5.1 Detekce objektů ze ztrátové mapy pomocí OpenCV

Navigační stack v ROS2 využívá tzv. ztrátovou mapu (*costmap*), což je dvourozměrná mřížka s definovaným rozlišením, kde každá buňka je ohodnocena číselnou hodnotou reprezentující bezpečnost průchodu. Tato hodnota je odvozena z dat získaných LIDAREm během mapování, přičemž počátek souřadného systému mapy odpovídá pozici robota při zahájení mapovacího procesu. Dynamická transformace mezi robotem a mapou (viz sekce 3.4) zajišťuje konzistentní propojení těchto dat. Kromě bodů označených jako překážky (detekovaných LIDAREm) zahrnuje ztrátová mapa bezpečnostní zóny: primární zónu, do níž robot nesmí vstoupit kvůli svým rozměrům, a sekundární doporučenou zónu, která optimalizuje trasu robota tak, aby se vyhnul blízkosti překážek.

Ztrátovou mapu lze převést na pole hodnot, ze kterého je možné vyfiltrovat pouze buňky odpovídající skutečným překážkám. Tento výsledek může být reprezentován jako binární obrázek, kde hodnota 1 označuje překážku a 0 volný prostor. Na tento obrázek lze aplikovat techniky počítacového vidění z knihovny OpenCV, například detekci kontur (*contour detection*) k identifikaci oblastí potenciálně uzavřených objektů, jako jsou kruhy nebo obdélníky. Tyto oblasti jsou poté ohraničeny obdélníkovými rámečky (*bounding boxes*) a klasifikovány jako objekty v mapě. Tento přístup se ukázal jako perspektivní pro detekci nohou palety díky své schopnosti rozlišovat izolované struktury.

Přes počáteční úspěch byl tento postup opuštěn z několika důvodů, které omezují jeho použitelnost v reálném čase a průmyslových aplikacích. První nevýhodou je nízká výpočetní efektivita – zpracování celé ztrátové mapy, jejíž velikost roste s pohybem robota, vyžaduje značné množství operací, což vede k prodlevám. Druhým problémem je závislost na úplném zmapování objektu; například detekce palety vyžaduje zachycení všech čtyř nohou, což znamená, že robot musí opakovaně objet paletu z různých úhlů. Tento požadavek navíc komplikuje fakt, že mapování probíhá kontinuálně, což může vést k zanesení palety do globální mapy jako trvalé překážky. Aby byla překážka zahrnuta do mapy, musí být detekována z více pozic, což slouží k eliminaci šumu a dočasných objektů, ale zvyšuje časovou náročnost procesu.

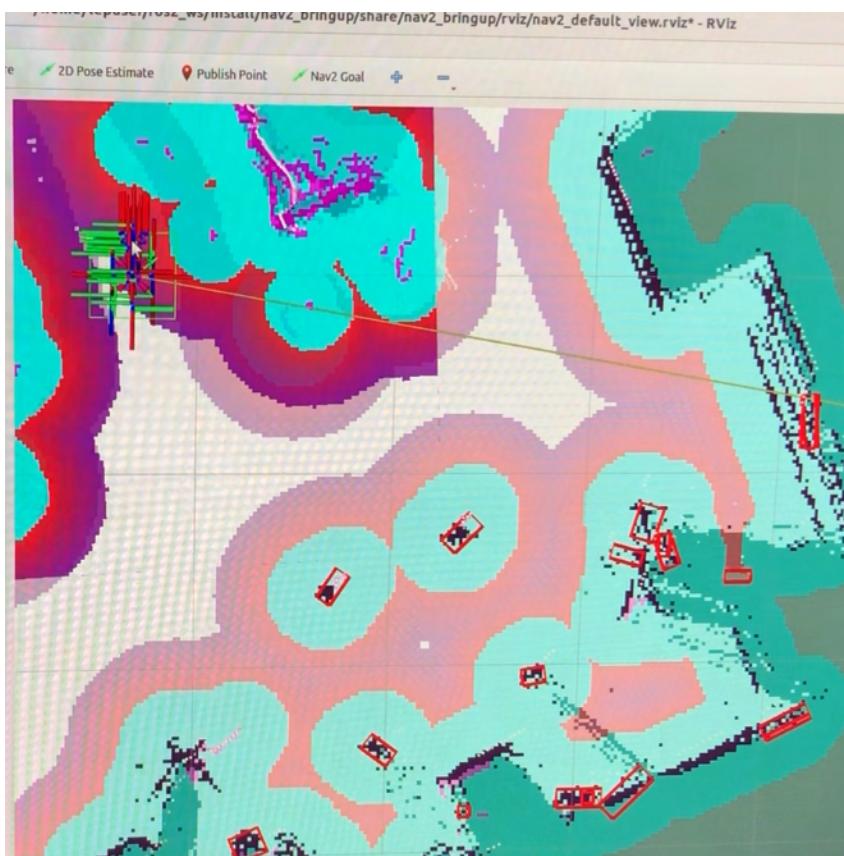
Obrázek 4.16 zachycuje scénu reálného nasazení autonomního vozidla v mapovaném prostředí. Celkové zobrazení představuje 2D ztrátovou mapu, která je vybarvena různými barvami podle bezpečnostních hodnot průchodu.

- **Bílé oblasti:** Tyto oblasti označují volný prostor, který je považován za bezpečný pro pohyb robota. Jsou reprezentovány buňkami ztrátové mapy s nízkými hodnotami, kde robot může bez rizika operovat.
- **Modré / tyrkysové oblasti:** Tyto barvy označují bezpečnostní zóny kolem překážek. Primární bezpečnostní zóna (blíže k překážkám) je vymezena tak, aby zabránila robotovi

narazit do překážek kvůli jeho rozměrům.

- **Červené / růžové oblasti:** Tyto barvy označují bezpečnostní zóny kolem překážek. Sekundární doporučená zóna, která je širší, slouží k optimalizaci trasy robota, aby se vyhnul blízkosti objektů, ale zároveň zůstal efektivní.

Robot je znázorněn zeleným obdélníkem, což je typické pro vizualizaci modelu robota v RViz. Tento obdélník ukazuje aktuální polohu a orientaci (pomocí RGB šipek znázorňujících osy souřadného systému) robota v mapě. Na obrázku jsou rovněž vidět červené obdélníkové rámečky, které ohraničují detekované objekty. Tyto rámečky odpovídají výsledkům detekce potenciálně uzavřených objektů jako jsou nohy palety, které byly identifikovány z binárního obrázku vytvořeného ze ztrátové mapy. Tyto obdélníky slouží k označení oblastí, které by mohly představovat fyzické překážky nebo cíle a jsou výsledkem analýzy pomocí OpenCV. O něco lépe princip zobrazuje [Video 3](#) [3].



Obrázek 4.16: Výsledek detekce z OpenCV - RViz vizualizace

I přes funkčnost tohoto přístupu jeho omezení vedla k hledání alternativy založené na přímém zpracování syrových LIDARových dat. Tento přístup využívá shlukovací metody, jejichž teoretické základy jsou podrobněji popsány v následující sekci.

---

[3] ↑ <https://youtu.be/TdZPEyCgbRs>

## 4.5.2 Shlukování ve strojovém učení a rozpoznávání objektů

Shlukování (*clustering*) je technika strojového učení a rozpoznávání vzorů, která seskupuje datové body do shluků na základě jejich podobnosti vzhledem k definovaným příznakům. V kontextu detekce objektů z LIDARových dat lze shlukování využít k identifikaci skupin bodů odpovídajících fyzickým objektům. Metody shlukování se liší ve způsobu měření podobnosti, zpracování šumu a požadavcích na vstupní parametry. Hlavními výzvami jsou nízká separabilita tříd, přítomnost šumu a odlehlých bodů (*outliers*), což vyžaduje pečlivý výběr vhodného algoritmu.

### 4.5.2.1 Algoritmus K-means

K-means je nehierarchický shlukovací algoritmus, který rozděluje datové body do předem stanoveného počtu shluků  $K$ . Podobnost mezi body je obvykle měřena euklidovskou vzdálostí v  $n$ -dimenzionálním prostoru:

$$d(\mathbf{x}_i, \mathbf{c}_j) = \sqrt{\sum_{k=1}^n (x_{i,k} - c_{j,k})^2},$$

kde  $\mathbf{x}_i$  je pozice  $i$ -tého bodu,  $\mathbf{c}_j$  je centroid  $j$ -tého shluku a  $n$  je počet dimenzí (zde typicky 2 pro 2D LIDARová data). Cílem algoritmu je minimalizovat součet čtverců vzdáleností bodů od jejich přiřazených centroidů:

$$J = \sum_{j=1}^K \sum_{\mathbf{x}_i \in S_j} \|\mathbf{x}_i - \mathbf{c}_j\|^2,$$

kde  $S_j$  je množina bodů přiřazených ke shluku  $j$ .

**Postup algoritmu:**

1. Inicializace: Náhodně umístí  $K$  centroidů do prostoru dat.
2. Přiřazení: Každý bod  $\mathbf{x}_i$  je přiřazen nejbližšímu centroidu  $\mathbf{c}_j$  na základě euklidovské vzdálenosti.
3. Aktualizace: Přepočítá se poloha centroidů jako průměr bodů ve shluku:

$$\mathbf{c}_j = \frac{1}{|S_j|} \sum_{\mathbf{x}_i \in S_j} \mathbf{x}_i.$$

4. Opakování: Kroky 2 a 3 se iterují, dokud se polohy centroidů nemění (nebo je změna menší než zvolená tolerance).

Počet shluků  $K$  musí být určen předem, například odhadem na základě analýzy dat. K-means je efektivní pro velké datasety, ale předpokládá, že shluky mají přibližně sférický tvar a podobnou velikost, což může být limitující.

### 4.5.2.2 Algoritmus DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) je algoritmus založený na hustotě rozložení bodů, který nepotřebuje předem specifikovat počet shluků. Identifikuje shluky jako oblasti vysoké hustoty oddělené oblastmi nízké hustoty, přičemž šum a odlehlé

body jsou explicitně odfiltrovány. Klíčovými parametry jsou poloměr sousedství  $\epsilon$  a minimální počet bodů  $MinPts$  potřebných k vytvoření shluku.

#### Definice typů bodů:

- Středový bod (*core point*): Bod, jehož  $\epsilon$ -sousedství obsahuje alespoň  $MinPts$  bodů (včetně sebe sama).
- Hraniční bod (*border point*): Bod, který není středový, ale leží v  $\epsilon$ -sousedství středového bodu.
- Odlehlý bod (*noise point*): Bod, který není ani středový, ani hraniční.

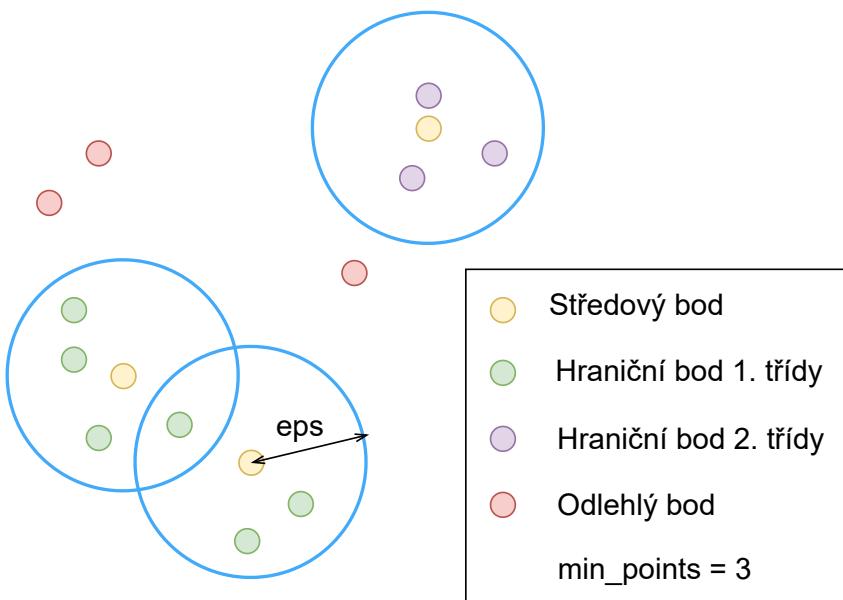
#### Postup algoritmu:

1. Určí se typ každého bodu podle  $\epsilon$  a  $MinPts$ .
2. Odlehlé body jsou označeny jako šum a odstraněny z dalšího zpracování.
3. Středové body, jejichž  $\epsilon$ -sousedství se překrývají, jsou spojeny do jednoho shluku.
4. Hraniční body jsou přiřazeny ke shluku nejbližšího středového bodu.

Hustota je definována počtem bodů v  $\epsilon$ -sousedství: bod  $\mathbf{x}_i$  je středový, pokud:

$$|N_\epsilon(\mathbf{x}_i)| \geq MinPts,$$

kde  $N_\epsilon(\mathbf{x}_i) = \{\mathbf{x}_j \mid d(\mathbf{x}_i, \mathbf{x}_j) \leq \epsilon\}$  a  $d$  je euklidovská vzdálenost. DBSCAN je schopen identifikovat shluky libovolného tvaru a efektivně eliminovat šum, což je výhodné pro analýzu LIDARových dat.



Obrázek 4.17: Schéma principu algoritmu DBSCAN s rozlišením středových, hraničních a odlehlých bodů.

#### 4.5.2.3 Vyhodnocení a rozdíly mezi K-means a DBSCAN

Následující tabulka shrnuje klíčové rozdíly mezi algoritmy K-means a DBSCAN z hlediska jejich vlastností a omezení:

K-means	DBSCAN
Shluky musí být přibližně sférické nebo konvexní a podobné velikosti.	Shluky mohou mít libovolný tvar a různou velikost.
Vyžaduje předem definovaný počet shluků ( $K$ ).	Počet shluků není nutné specifikovat.
Efektivní pro velké datasety s nízkou dimenzí.	Méně efektivní pro vysoce dimenzionální datasety.
Citlivý na šum a odlehlé body, které zkreslují centroidy.	Robustní vůči šumu a odlehlým bodům, které označuje jako šum.
Špatná detekce anomálií; odlehlé body jsou přiřazeny k nejbližšímu shluku.	Schopen identifikovat oblasti vysoké hustoty oddělené nízkou hustotou.
Jeden parametr: $K$ (počet shluků).	Dva parametry: $\epsilon$ (poloměr) a $MinPts$ (minimální počet bodů).
Nezávislý na variacích hustoty bodů.	Méně efektivní u datasetů s proměnlivou hustotou.

Pro detekci malých objektů z 2D LIDARových skenů je DBSCAN výrazně vhodnější než K-means z několika důvodů. Zaprve, LIDARová data často obsahují šum a odlehlé body způsobené odrazy nebo dočasnými překážkami, které K-means nedokáže efektivně odfiltrovat, zatímco DBSCAN je explicitně označuje jako šum. Zadruhé, tvar shluků odpovídajících nohám palet nemusí být sférický ani konvexní, což je omezení K-means, ale nikoli DBSCAN, který zvládá libovolné geometrické struktury. Zatřetí, nutnost předem určit počet shluků ( $K$ ) u K-means je nepraktická v situacích, kde počet objektů není znám, například při průzkumu neznámého prostředí, zatímco DBSCAN tuto informaci nevyžaduje. Ačkoli DBSCAN může být výpočetně náročnější a citlivý na volbu parametrů  $\epsilon$  a  $MinPts$ , jeho robustnost a flexibilita převažují v kontextu detekce palet, kde je klíčová přesnost a schopnost pracovat s reálnými, zašuměnými daty v reálném čase. Proto byl DBSCAN zvolen jako preferovaný algoritmus pro další zkoumání a implementaci v této práci.

# **Kapitola 5**

## **Aplikace**

---

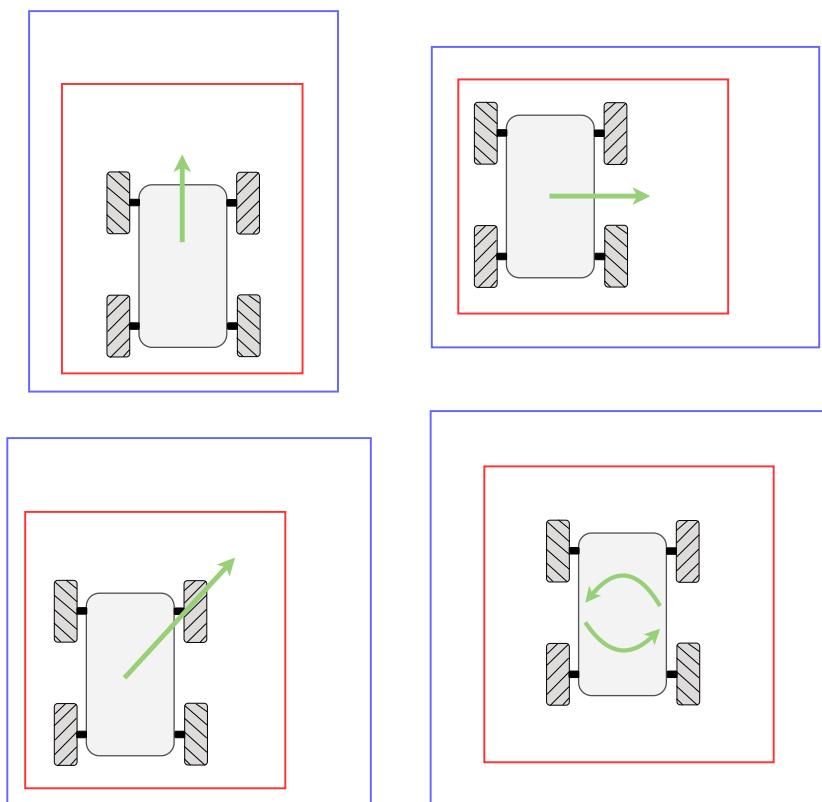
Tato kapitola se zaměřuje na praktickou aplikaci teoretických metod a matematických postupů popsaných v předchozích částech, jejichž cílem je vytvoření funkčního systému pro řízení a navigaci robotů. Integrací těchto poznatků jsou zde prezentovány ucelená navaigační řešení zahrnující autonomní pohyb v prostoru, navigaci podle fyzických čar, sledování virtuálních trajektorií, detekci objektů, jako jsou palety, a definici bezpečnostních zón. Prostřednictvím příkladů testů jednotlivých navaigačních úloh kapitola ilustruje, jak jsou teoretické základy převedeny do praxe, a poskytuje přehled o nezbytných komponentách, vstupech a struktuře řešení daných problémů. Současně slouží jako metodický návod pro sestavení systému schopného efektivně řešit reálné navaigační výzvy.

## 5.1 Bezpečnostní zóny autonomních vozíků

Bezpečnost je stěžejním aspektem průmyslové automatizace, zejména u autonomních vozíků (AGV), které sdílejí pracovní prostor s lidmi. Legislativní rámec a normy, například ISO 13849 nebo ISO 3691-4, stanovují přísné požadavky na prevenci kolizí prostřednictvím včasného zastavení či jiných reakcí. Tato kapitola se věnuje návrhu a implementaci virtuálních bezpečnostních zón kolem AGV, jejichž narušení spouští bezpečnostní mechanismy. Vzhledem k zásadnímu významu tohoto tématu pro praktické nasazení je kapitola zařazena na počátek části zaměřené na aplikaci postupů, metod a algoritmů.

Bezpečnostní zóny jsou koncipovány jako vícevrstvé struktury: vnější vrstvy slouží k omezení rychlosti nebo úpravě trajektorie, zatímco narušení vnitřních vrstev vyvolává okamžité zastavení vozíku (viz obrázek 5.1). Zastavení musí být v souladu s principy bezpečnosti (*safety*) různou měrou definitivní a trvat v nejpřísnějším případě, dokud operátor nepotvrdí odstranění rizika (např. stiskem tlačítka), nebo častěji dokud překážka po stanovenou dobu není detekována. Rovněž je nastavena rekonfigurace pro vymaněvrování od překážky. To zahrnuje zmenšení zóny a opatrný odjezd směrem od nebezpečí. Návrh zón zohledňuje dynamické vlastnosti vozíku, jako jsou maximální rychlosť, brzdná síla, hmotnost včetně nákladu a reakční doba systému. Velikost zón je dynamicky upravována podle aktuálních podmínek, protože brzdná dráha závisí na rychlosti.

U diferenciálních vozíků se zóny obvykle rozšiřují ve směru pohybu (dopředu, dozadu) a při zatáčení kombinací doprava/doleva. U vozíků s mecanum koly, které umožňují pohyb libovolným směrem, je situace komplexnější. Přestože je preferován přední pohyb na rovných úsecích, složité trajektorie vyžadují adaptabilní zóny, jejichž velikost (respektive poměr stran rozšíření) je odvozována z vektoru rychlosti (zelená šipka na obrázku 5.1).

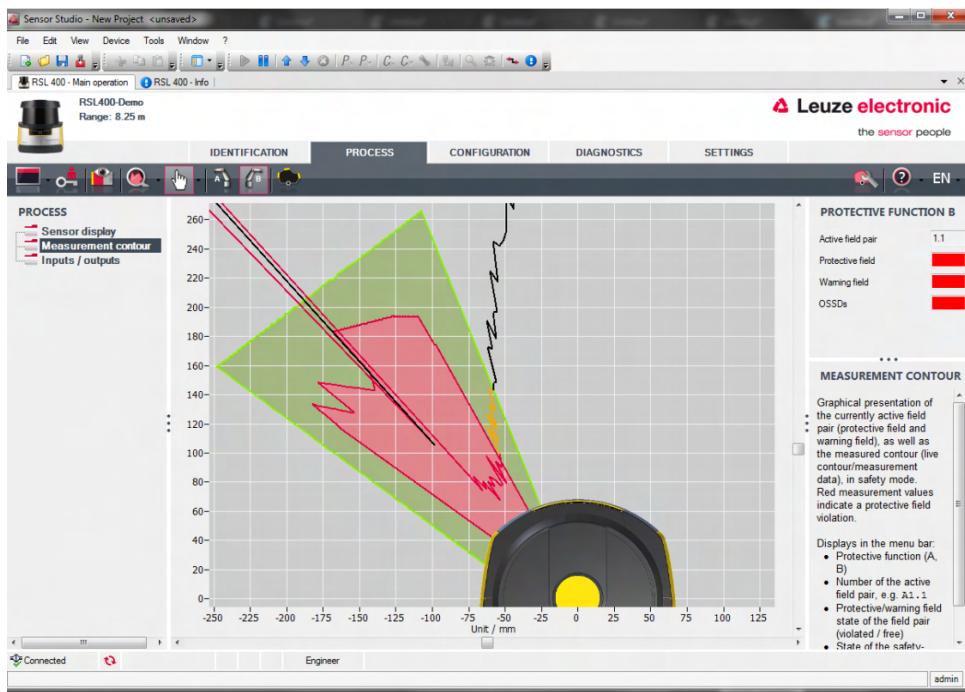


Obrázek 5.1: Schéma vícevrstvých bezpečnostních zón kolem autonomního vozíku.

### 5.1.1 Hardwareové řešení s PLC

V průmyslových aplikacích, kde je nutné splnit přísné bezpečnostní normy, se využívá hardwareový bezpečnostní řadič, například programovatelný logický řadič (PLC). V této práci byl použit Simatic S7-1500 od Siemens, který zajišťuje certifikovanou úroveň bezpečnosti (*Safety Integrity Level, SIL*) díky bezpečnému zpracování signálů v reálném čase a tím pádem zaručenému zastavení motorů při narušení zón. Bezpečnostní logika je zcela oddělena od řídicího systému a běží nezávisle na prioritizované části CPU.

Zásadní roli hrají LIDARové senzory, které jsou bezpečnostně certifikovány a vybaveny výstupy OSSD (*Output Signal Switching Device*). Tyto senzory umožňují interně konfigurovat zóny a dynamicky je zvenčí přepínat podle směru pohybu a rychlosti vozíku. Při detekci narušení (např. přítomnosti člověka) generují OSSD signál, který PLC zpracuje a aktivuje odpovídající opatření – okamžité zastavení nebo snížení rychlosti. Rozšířená varovná zóna umožňuje vyšší rychlosti ve větších prostorách: při jejím narušení dojde ke zpomalení, které zajistí dostatečnou brzdnou dráhu při aktivaci vnitřní bezpečnostní zóny. Tento přístup splňuje průmyslové normy a poskytuje hardwarovou záruku bezpečnosti. Obrázek 5.2 ukazuje konfiguraci zón v softwaru Sensor Studio pro RSL400, kde zelená barva označuje varovnou zónu, červená ochrannou zónu a barevně rozlišený sken znázorňuje narušení jednotlivých zón rukou.



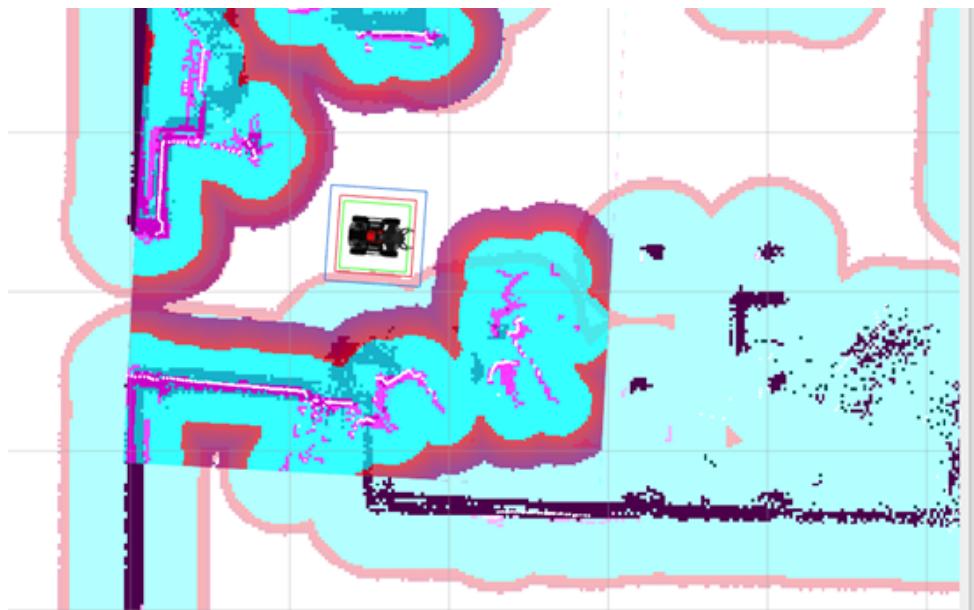
Obrázek 5.2: Konfigurace zón RSL400 a detekce narušení v Sensor Studiu [16].

U vozíků v autonomním režimu jsou zóny dynamicky přepínány podle směru a rychlosti: při nízkých rychlostech jsou menší, při vyšších se rozšiřují v souladu s brzdnou dráhou. V manuálním režimu (např. ovládání joystickem) jsou zóny a rychlosti omezeny, což usnadňuje manévrování v úzkých prostorách, zatímco v autonomním módu jsou zóny rozšířeny.

### 5.1.2 Softwarové řešení v ROS2

Pro testovací model vozíku (viz sekce 2.2.1) bez certifikovaného PLC bylo vyvinuto softwarové řešení v ROS2 využívající nástroj Collision Monitor z navigačního stacku Nav2. Tento uzel monitoruje data z LIDARů a zajišťuje rychlou reakci na potenciální kolize nezávisle na ztrátové mapě (*costmap*) a plánovačích trajektorií, čímž napodobuje funkce bezpečnostních senzorů a hardwaru. Na rozdíl od PLC nezpracovává data na úrovni CPU a podporuje různé senzory (LIDAR, hloubkové kamery), avšak neposkytuje hardwarovou certifikaci ani real-time záruky. Je tak vhodný zejména pro testování a vývoj či dodatečnou ochranu.

Collision Monitor definuje zóny jako polygony vzhledem k základnímu rámci robota (nazývanému *base\_frame*), které mohou být statické nebo dynamicky upravované přes ROS2 témata. Při narušení zón spouští různé reakce: *Stop model* zastaví vozík při překročení prahu bodů, *Slowdown model* sníží rychlosť podle zadaného poměru, *Limit model* omezí maximální rychlosti a *Approach model* zpomalí vozík na základě odhadu času do kolize. Zóny mohou mít tvar kruhu, uživatelského polygonu, stopy robota (*footprint*) nebo *VelocityPolygon*, který se mění podle rychlosti. Tento mechanismus je efektivní pro rychlé zastavení při náhlých překážkách v prostředí s lidmi nebo jinými roboty. Na obrázku 5.3 je kolem modelu robota vidět zeleně jeho stopa (půdorys), o trochu širší je červená ochranná stop zóna a nejšířší je modrá zpomalovací zóna. Jakmile se jakákoli z těchto zón překryje s buňkou nákladové mapy označené jako překážka, dojde k aktivování příslušného pravidla.



Obrázek 5.3: Zóny kolem robota v RVizu vytvořené Collision Monitorem.

### 5.1.3 Kombinované řešení s OSSD a softwarovým monitoringem

U testovacího modelu bez PLC bylo implementováno kombinované řešení integrující OSSD výstupy LIDARů se softwarovým monitoringem v ROS2. OSSD signály, generované při narušení zón nastavených v senzoru, jsou připojeny na GPIO piny řídicího počítače (Raspberry Pi) a doplněny Collision Monitorem pro redundantní detekci. Toto řešení však nenahrazuje hardwarovou bezpečnost PLC, protože Raspberry Pi ani ROS2 nezaručují real-time reakce a jsou náchylné na výpadky komunikace (např. přes Wi-Fi), což může vést ke kolizi při ztrátě signálu. Implementace pouze na lokální řídicí jednotce snižuje riziko, avšak přesnost zůstává závislá

na lokalizaci robota. Dynamické přepínání zón podle směru a rychlosti bylo zajištěno modelem *VelocityPolygon*, což umožnilo flexibilní adaptaci. Tento přístup je praktický pro testovací scénáře, kde je prioritou rychlá reakce při omezené hardwarové bezpečnosti.

#### 5.1.4 Shrnutí

Tato sekce shrnuje návrh a implementaci bezpečnostních zón s ohledem na dynamiku vozíku, průmyslové normy a specifika pohybu. Hardwarové řešení s PLC je preferováno v průmyslu díky souladu s normami a real-time spolehlivosti. Softwarové řešení v ROS2 nabízí flexibilitu a nízké náklady, ale postrádá certifikaci a je citlivé na technické limity, jako jsou výpadky komunikace. Kombinované řešení s OSSD a Collision Monitorem představuje kompromis pro vývojové účely, avšak nenahrazuje hardwarovou bezpečnost. Softwarové mechanismy mohou nicméně doplňovat PLC pro lepší integraci s řídicím systémem.

## 5.2 Autonomní navigace

Pojmem *autonomní navigace* je v kontextu této práce myšlena schopnost průmyslových AGV pohybovat se v předem zmapovaném prostředí mezi aktuální polohou a cílovou pozicí určenou uživatelem. V rámci vývoje byla tato funkcionality implementována na několika robotech s různými druhy podvozků a je proto potřeba, aby byla modulární. Je využit navigační stack Nav2 v ekosystému ROS2. Tato kapitola poskytuje komplexní popis fungování navigačního stacku, který integruje teoretické metody a komponenty popsané v předchozích částech (mapování, lokalizace, plánování trajektorií, řízení, bezpečnost atd.) s praktickou realizací na hardware (senzory, různé podvozky apod.). Cílem je demonstrovat, jak tyto subsystémy spolu pracují pro zajištění bezpečného, přesného a efektivního pohybu vozíku v dynamickém průmyslovém prostředí, přičemž jsou respektovány přísné bezpečnostní normy (např. ISO 13849, ISO 3691-4).

### 5.2.1 Základní koncepty a mechanismy navigace v Nav2

Tato kapitola se věnuje klíčovým principům a mechanismům autonomní navigace mobilních robotů v rámci navigačního stacku Nav2, který je vybudován na middleware ROS2. Cílem je seznámit čtenáře se základními stavebními bloky Nav2, včetně akčních serverů, uzlů s řízením životním cyklem, stromů chování, navigačních serverů, sledování bodů trasy a odhadu stavu, které společně tvoří robustní a modulární systém pro plánování, řízení a správu pohybu robotů v dynamických prostředích. Jednotlivé podkapitoly podrobně popisují tyto koncepty a jejich vzájemné propojení, címž poskytují ucelený přehled nezbytný pro pochopení fungování Nav2 a jeho praktické využití v experimentálních i průmyslových aplikacích.

#### 5.2.1.1 Akční servery

Akční servery představují standardní mechanismus v ROS2 pro správu dlouhotrvajících a asynchronních úkolů. V rámci Nav2 jsou akční servery intenzivně využívány pro řízení komplexních navigačních procesů, přičemž v mnoha případech nahrazují jednodušší komunikační rozhraní založená na topicích, která jsou méně vhodná pro úkoly vyžadující průběžnou zpětnou vazbu a koordinaci.

Akční servery fungují na principu klient-server komunikace, podobně jako klasické servisní servery v ROS2, avšak s důrazem na asynchronní zpracování úkolů. Klient zadá požadavek na úkol, jehož dokončení může trvat delší dobu, například přesun robota na cílovou pozici o 10 metrů doprava. Tento úkol je poté zpracováván v samostatném vlákně na straně serveru, což umožňuje klientovi pokračovat v dalších činnostech. Klient může buď synchronně čekat na dokončení úkolu blokováním svého vlákna, nebo asynchronně kontrolovat stav úkolu a získávat průběžnou zpětnou vazbu. Struktura akce, včetně požadavku, zpětné vazby a výsledku, je definována v souboru s příponou `.action`. V uvedeném příkladu přesunu robota může požadavek (*goal*) specifikovat cílovou pozici, zpětná vazba (*feedback*) poskytovat informace, jako je zbývající vzdálenost k cíli nebo doba navigace, a výsledek (*result*) potvrdit úspěšnost operace logickou hodnotou.

Zpětná vazba a výsledky mohou být získávány dvěma způsoby: synchronně prostřednictvím registrace zpětných volání (*callbacks*) na straně klienta, nebo asynchronně dotazováním na sdílené budoucí objekty (*shared futures*). V obou případech je nutné, aby klientský uzel (*node*) byl aktivní a zpracovával příchozí zprávy, což zajišťuje správnou synchronizaci a komunikaci mezi klientem a serverem.

V rámci Nav2 hrají akční servery klíčovou roli při koordinaci navigačních úkolů prostřednictvím stromů chování (*Behavior Trees*, BT), které tvoří nejvyšší úroveň řízení navigátoru. Hlavní akční server Nav2 komunikuje s klienty pomocí zprávy `NavigateToPose`, která definuje požadavek na navigaci robota na konkrétní pozici a orientaci v prostoru. Tento server dále propojuje BT navigátor s nižšími akčními servery, které zpracovávají specifické úkoly, jako je plánování globální a lokální trasy, řízení pohybu robota nebo obnovovací akce (*recovery behaviors*), například při zaseknutí robota. Každý z těchto akčních serverů má vlastní definici akce v balíčku `nav2_msgs`, což zajišťuje standardizovanou a efektivní interakci mezi jednotlivými komponentami navigačního stacku Nav2.

### 5.2.1.2 Uzly s řízeným životním cyklem

Uzly s řízeným životním cyklem (*Lifecycle Nodes*, někdy označované jako *Managed Nodes*) představují standardní mechanismus v ROS2 pro zajištění předvídatelného a strukturovaného chování uzlů během jejich životního cyklu. Podrobný popis jejich fungování nabízí oficiální dokumentace ROS2 [36]. Tyto uzly jsou založeny na stavovém automatu, který definuje přechody mezi jednotlivými stavami, jako je spuštění, konfigurace, aktivace, deaktivace a ukončení. Tento přístup zaručuje kontrolované a předvídatelné chování systému při jeho inicializaci, běhu a vypínání.

Životní cyklus uzlu začíná v nezkonfigurovaném stavu (*Unconfigured*), kdy uzel pouze inicializuje svůj konstruktor. V této fázi by konstruktor neměl provádět operace, jako je nastavování síťových rozhraní nebo čtení parametrů, aby se předešlo nepředvídatelnému chování. Spouštěcí systém nebo správce životního cyklu (*Lifecycle Manager*) poté převádí uzel do neaktivního stavu (*Inactive*) prostřednictvím fáze konfigurace, která je vyvolána metodou `on_configure()`. Během konfigurace se nastavují parametry, inicializují síťová rozhraní ROS (například publishery, subscribers nebo klientské služby) a v případě bezpečnostních systémů se alokuje dynamická paměť. Následně je uzel převeden do aktivního stavu (*Active*) metodou `on_activate()`, která aktivuje síťová rozhraní a připraví uzel na zpracování dat, například příjem zpráv nebo odesílání povelů.

Při ukončení uzel prochází opačným procesem: nejprve je deaktivován metodou `on_deactivate()`, čímž se deaktivují síťová rozhraní a uzel přestane zpracovávat data. Následně je vyčištěn metodou `on_cleanup()`, během níž se uvolňuje alokovaná paměť a zdroje, a nakonec je uzel převeden do finálního stavu (*Finalized*) metodou `on_shutdown()`, což zajišťuje čisté ukončení procesu. Tento strukturovaný přístup minimalizuje riziko neočekávaných chyb při startu nebo vypínání systému.

V rámci projektu jsou uzly s řízeným životním cyklem široce využívány a aplikovány na všechny servery, které tvoří navigační stack. ROS2 doporučuje tento přístup jako preferovanou metodu pro správu uzlů, pokud je to technicky proveditelné, protože zvyšuje přehlednost a robustnost systému. V Nav2 je využívána obalová třída `nav2_util::LifecycleNode`, která zjednodušuje implementaci těchto uzlů pro běžné navigační aplikace.

Tato třída také poskytuje propojení typu *bond* se správcem životního cyklu, což zajišťuje, že server zůstane funkční po přechodu do aktivního stavu. V případě havárie serveru je správce životního cyklu okamžitě informován prostřednictvím *bond* mechanismu a systém je automaticky převeden do nižšího stavu, například do neaktivního nebo finálního, aby se předešlo kritickému selhání a zajistila bezpečnost celého navigačního procesu.

### 5.2.1.3 Stromy chování

Stromy chování (*Behavior Trees*, BT) představují stále populárnější přístup k řešení složitých robotických úloh díky své přehlednosti a škálovatelnosti. Na rozdíl od tradičních konečných stavových automatů (*Finite State Machines*, FSM), které mohou při větším počtu stavů a přechodů (desítky stavů a stovek přechodů) vést k nepřehledným a chybám náchylným implementacím, nabízejí BT stromovou strukturu úkolů, která usnadňuje definování vícekrokových aplikací. Například u robota hrajícího fotbal by implementace herní logiky pomocí FSM byla značně komplikovaná kvůli nutnosti modelovat množství stavů a pravidel, jako jsou rozhodnutí o střelbě na branku z různých pozic (zleva, zprava, ze středu). BT však umožňuje definovat základní akce, jako *kopnout*, *jít* nebo *jít k míči*, a hierarchicky je kombinovat do komplexních chování, což zjednoduší návrh a údržbu [37]. Tento formální rámec navíc usnadňuje analýzu a ověřování správnosti systému pomocí pokročilých nástrojů díky centralizaci logiky ve stromu a nezávislým serverům úkolů, které komunikují pouze prostřednictvím stromu.

V rámci Nav2 je pro implementaci stromů chování využita knihovna BehaviorTree V4 [38], která poskytuje prostředí pro tvorbu BT. Nav2 definuje sadu pluginů uzlů, které lze v BT Navigatoru sestavit do stromu chování. Tyto pluginy jsou načítány z XML souboru, kde jsou propojeny s registrovanými názvy, což umožňuje definovat navigační logiku prostřednictvím stromové struktury. Důvodem volby této knihovny je její podpora načítání podstromů, díky čemuž lze strom Nav2 integrovat jako podstrom do vyššího BT v rámci klientské aplikace. Klíčovým prvkem je plugin `NavigateToPoseAction`, který propojuje BT s akčním rozhraním Nav2, což umožňuje klientským aplikacím volat navigační stack standardním způsobem přes zprávu `NavigateToPose`.

Konkrétním příkladem využití BT v této práci je strom *Navigate To Pose With Consistent Replanning And If Path Becomes Invalid*, implementovaný v rámci Nav2. Tento strom zajistí uje robustní navigaci robota z výchozí pozice do cílové polohy ve volném prostoru a představuje pokročilejší řešení oproti základním BT v Nav2 díky integraci kontextově specifických a systémových obnovovacích mechanismů (*recovery behaviors*). Jeho hlavním cílem je zajistit spolehlivou navigaci s možností až šesti opakování pokusů před ohlášením selhání, což poskytuje dostatečný prostor pro překonání přechodných poruch, jako jsou dočasné výpadky senzorů nebo přítomnost dynamických překážek, například shluků lidí. Hlavní navigační část stromu zahrnuje uzly `ComputePathToPose` pro výpočet globální trajektorie a `FollowPath` pro její sledování, přičemž tyto uzly jsou navrženy obecně, aby abstrahovaly technické detaily a umožnily uživatelům definovat vlastní algoritmy plánování a řízení prostřednictvím pluginů, což bylo v této práci hojně využito.

Během nominálního provozu je přeplánování trajektorie iniciováno v případech, kdy se předchozí trasa stane neplatnou (je zadán nový cíl), nebo pokud nová trajektorie není k dispozici po dobu 10 sekund. Při selhání plánovače nebo regulátoru se aktivují kontextově specifické obnovovací podstromy: selhání plánovače (`ComputePathToPose`) vede k vymazání globální nákladové mapy, zatímco selhání regulátoru (`FollowPath`) spouští vymazání lokální nákladové mapy. Tyto akce řeší běžné problémy, jako jsou chyby ve vnímání prostředí, a mohou být rozšířeny o další strategie podle specifických potřeb aplikace. Pokud lokální obnovy selžou, strom přejde do globálního obnovovacího podstromu zaměřeného na systémové poruchy, například zaseknutí robota nebo jeho nevhodnou pozici. Tento podstrom zahrnuje uzel `GoalUpdated`, který je pravidelně aktualizován pro okamžitou reakci na nové cíle, a sekvenčně zkouší obnovovací akce, jako je vymazání nákladových map, rotace na místě, čekání, nebo couvání. Po každé akci je znova spuštěn hlavní navigační podstrom, a pokud selhání přetravá, pokračuje další obnovou v sekvenci, címž se maximalizuje pravděpodobnost úspěšného

dokončení úkoly.

Ačkoli tento strom využívá pevně definované algoritmy, jeho flexibilita může být rozšířena pomocí uzlů PlannerSelector, ControllerSelector a GoalCheckerSelector, které umožňují dynamickou změnu algoritmů plánování, řízení a kontroly cíle prostřednictvím ROS topiců. Tento přístup poskytuje uživatelům možnost přizpůsobit navigaci bez nutnosti úpravy samotného stromu, což je výhodné pro externí aplikace vyžadující rychlé změny nastavení. Alternativně lze vytvořit specializované podstromy s podmínkovými uzly optimalizovanými pro konkrétní scénáře, jako jsou úzké prostory nebo otevřené plochy. I když je pro větší přehlednost doporučováno implementovat změny přímo v logice BT, podpora dynamické konfigurace z externích systémů je praktická pro zkušené uživatele, kteří potřebují rychle upravovat chování navigátoru.

#### 5.2.1.4 Navigační servery

Navigační servery tvoří základní stavební bloky navigačního stacku Nav2 a zahrnují plánovače, regulátory, vyhlazovače a servery chování. Plánovače a regulátory zajišťují výpočet a sledování tras, vyhlazovače vylepšují kvalitu plánovaných cest a obnovovací akce zvyšují odolnost systému vůči poruchám. Níže jsou popsány jejich obecné principy a konkrétní využití v rámci projektu.

##### Servery plánovače, regulátoru, vyhlazovače a chování:

Projekt využívá čtyři hlavní akční servery: server plánovače (*Planner Server*), server regulátoru (*Controller Server*), server vyhlazovače (*Smoothen Server*) a server chování (*Behavior Server*). Tyto servery spravují mapu pluginů algoritmů pro různé navigační úkoly a hostují sdílenou reprezentaci prostředí, jako jsou globální a lokální nákladové mapy, které algoritmy využívají pro své výpočty.

Servery plánovače, regulátoru a vyhlazovače jsou během provozu konfigurovány pomocí názvů (aliasů) a typů algoritmů, které jsou registrovány v knihovně `pluginlib`. Názvy slouží jako aliasy pro jednotlivé úkoly, zatímco typy definují konkrétní implementace algoritmů. Uživatelé mohou vytvářet vlastní pluginy, pokud splňují požadovanou strukturu a parametry. Například v této práci je použit regulátor MPPI (*Model Predictive Path Integral*), který je v konfiguraci registrován pod standardním názvem `FollowPath` pro sledování referenční trasy. Jeho parametry jsou uloženy v příslušném jmenném prostoru, například `FollowPath.<param>`, což umožňuje snadnou konfiguraci a přizpůsobení.

Každý z těchto serverů poskytuje akční rozhraní odpovídající jeho úkolu. Když strom chování (*Behavior Tree*, BT) aktivuje příslušný uzel, například `FollowPath`, zavolá odpovídající akční server, jehož zpětné volání (*callback*) vybere algoritmus podle zadaného názvu. Tento přístup umožňuje abstrahovat algoritmy v BT do obecných tříd, což podporuje modularitu – například lze definovat N pluginů regulátorů pro různé úkoly, jako je sledování tras, dokování k nabíječce, vyhýbání se překážkám nebo interakce s prostředím. Sdílení pluginů v rámci jednoho serveru optimalizuje využití reprezentace prostředí, protože duplikace nákladových map a dalších zdrojů by byla výpočetně náročná.

Server chování (*Behavior Server*) přiřazuje každému chování specifický název a každý plugin má svůj vlastní akční server, protože rozmanitost akcí (například obnovovací chování, čekání nebo rotace) nelze sjednotit pod jedno univerzální rozhraní. Tento server také přijímá aktualizace lokální nákladové mapy od regulátoru v reálném čase, což eliminuje potřebu duplikace nákladných instancí mapy a zvyšuje efektivitu.

Díky tomu, že uzly BT jsou jednoduché pluginy volající akce, lze snadno vytvořit nové

uzly pro jiné akční servery s různými typy akcí. Přesto se doporučuje využívat stávající servery, pokud to jejich rozhraní umožnuje. V případě potřeby nového serveru jej lze implementovat s novým typem a rozhraním pluginu, což vyžaduje vytvoření nového pluginu uzlu BT. Tento modulární přístup umožňuje rozšíření funkcionality bez nutnosti úprav základního repozitáře Nav2. V rámci této práce byl tento přístup využit a potřebné úpravy byly implementovány jako samostatné pluginy pod existujícím rozhraním příslušného serveru, což zajistilo snadnou kompatibilitu. Pro všechny následující funkcionality existuje v Nav2 rozhraní pluginů (viz [39]). Je možné tedy různě kombinovat již existující, případně vytvářet své vlastní.

### **Plánovače:**

Plánovač (*Planner*) je odpovědný za výpočet globální trasy podle zadané cílové pozice a účelové funkce. Mezi typické úkoly plánovače patří výpočet nejkratší trasy z aktuální pozice do cílové polohy, výpočet tras pro úplné pokrytí volného prostoru (například při průzkumu) nebo výpočet tras podle předem určených cest. Plánovač využívá globální reprezentaci prostředí a data ze senzorů, například LiDARů nebo kamer. V Nav2 má plánovač za úkol vytvořit platnou, ideálně optimální trasu z aktuální do cílové polohy, přičemž podporuje různé typy plánů prostřednictvím pluginů, jako jsou třeba NavFn, Theta\* nebo SMAC Planner, které lze konfigurovat podle potřeb aplikace.

### **Regulátory:**

Regulátory (*Controllers*), označované také jako lokální plánovače, zajišťují sledování globální trasy a řešení lokálních problémů v reálném čase. Využívají lokální reprezentaci prostředí, jako je lokální nákladová mapa, k výpočtu proveditelných akčních vstupů, například rychlosti a směru pohybu robota. V této práci použitý regulátor MPPI predikuje budoucí polohu robota na základě aktuálních senzorových dat a při každé aktualizaci vypočítá lokální trasu, která zohledňuje dynamické překážky a další omezení. Mohou být navrženy pro různé úkoly, jako je sledování globální trasy, dokování k nabíječce s využitím detektorů, nebo interakce s objekty v prostředí. V Nav2 regulátor generuje akční zásahy potřebné pro sledování globálního plánu, přičemž projekt podporuje různé třídy regulátorů ve formě pluginů, například DWB (Dynamic Window Approach), MPP I nebo TEB (Timed Elastic Band), které lze dynamicky měnit podle požadavků.

### **Server chování:**

Server chování (*Behavior Server*) hraje klíčovou roli při zajišťování odolnosti systému vůči poruchám prostřednictvím obnovovacích chování (*recovery behaviors*). Tato chování řeší neznámé nebo chybové stavů, například když percepční systém detekuje falešné překážky, což vede k vymazání nákladové mapy, nebo když se robot zaseknut kvůli dynamickým překážkám, což spustí akce jako couvání, rotace na místě nebo přesun do volného prostoru. V případě úplného selhání může obnova zahrnovat upozornění operátora, například prostřednictvím e-mailu nebo SMS. Server chování však není omezen pouze na obnovovací akce – může hostovat různé typy chování s vlastními API, které sdílejí společné zdroje, jako jsou nákladové mapy nebo buffery transformací (*TF buffers*), což zvyšuje efektivitu a modularitu systému.

### **Vyhlažovače:**

Vyhlažovače (*Smoothers*) slouží k úpravě globální trasy vygenerované plánovačem, protože kritéria optimality plánovače jsou často omezená a nemusí plně zohledňovat reálné podmínky prostředí. Vyhlažovače snižují hrubost tras, vyhlazují zatáčky a zvyšují vzdálenost od překážek díky přístupu ke globální reprezentaci prostředí. Oddělený vyhlazovač je výhodný, protože

umožňuje kombinovat různé plánovače a vyhlazovače nebo provádět cílené úpravy trasy podle specifických požadavků. V Nav2 vyhlazovač přijímá trasu od plánovače a vrací její vylepšenou verzi, přičemž podporuje různé metody vyhlazování, jako jsou Simple Smoother nebo Constrained Smoother, které lze konfigurovat podle potřeb aplikace.

### 5.2.1.5 Sledování bodů trasy

Sledování bodů trasy (*waypoints*) představuje základní funkci autonomní navigace v Nav2, která umožňuje robotovi dosáhnout jednoho nebo více cílových bodů v prostředí. Základním scénářem je navigace k jednomu cílovému bodu, kdy robot využívá navační stack k výpočtu a sledování tras z aktuální pozice do zadaného místa. Nav2 však podporuje i pokročilejší scénáře, jako je zadání více bodů trasy, což umožňuje definovat sekvenci cílů, které robot postupně projede. Tyto body mohou být buď proloženy zhruba a vytvořena souvislá trasa, nebo může být vyžadováno přesné projetí každého zvoleného bodu s možností definování specifického chování v jednotlivých průjezdních bodech.

Balíček `nav2_waypoint_follower` poskytuje rámec pro správu a sledování bodů trasy prostřednictvím pluginů, které umožňují definovat specifické akce nebo chování v jednotlivých cílových bodech. Uživatelé mohou například nastavit, aby robot v daném bodě provedl konkrétní úkol, jako je pořízení fotografie, zvednutí palety, čekání určitou dobu, nebo jiné interakce s prostředím - například otevření dveří. Tento přístup umožňuje přizpůsobit navigaci specifickým požadavkům aplikace, přičemž chování v průjezdních bodech lze volit buď přímo v konfiguraci trasy, nebo dynamicky během provozu prostřednictvím ROS topiců.

Nav2 podporuje několik režimů pro zpracování bodů trasy. V základním režimu robot projíždí body v pořadí, přičemž každý bod je považován za samostatný cíl a navigace probíhá s využitím standardního BT stromu, jako je `NavigateToPose`. V pokročilejším režimu lze trasové body proložit a vytvořit souvislou globální trasu, kterou robot sleduje jako jednu kontinuální cestu, což je efektivní pro scénáře, kde není nutné zastavovat v každém bodě. V podstatě se jedná o přesnější definování požadované cesty, než když je zadán jen cílový bod. Alternativně může být vyžadováno přesné projetí jednotlivých bodů, což je užitečné v aplikacích, kde je potřeba zajistit, že robot fyzicky dosáhne přesné pozice, například při manipulaci s nákladem nebo při synchronizaci s jinými systémy.

### 5.2.1.6 Odhad stavu a transformace v navačním stacku

Pro správné fungování navačního stacku Nav2 je nezbytné zajistit odhad stavu robota, což zahrnuje jeho lokalizaci a odometrii, a definovat povinné transformace mezi jednotlivými referenčními rámcemi (souřadnými systémy). Tyto transformace jsou nezbytné pro výpočet polohy robota v prostředí, plánování tras a správu nákladových map. V této části jsou popsány požadované transformace, standardy, mechanismy lokalizace, odometrie a správa nákladových map.

#### Standardy a povinné transformace:

Navigace v Nav2 vyžaduje definici dvou klíčových dynamických transformací: `map → odom` a `odom → base_link`. Transformace `map → odom` je poskytována pozičním systémem, jako je lokalizace nebo SLAM, a popisuje vztah mezi globální mapou (*map frame*) a odometrickým referenčním rámcem (*odom frame*). Transformace `odom → base_link` je zajišťována odometrií a popisuje polohu robota (*base\_link frame*) vzhledem k odometrickému rámcovi. Kromě těchto dynamických transformací jsou potřeba statické transformace mezi `base_link` a rámcem senzorů (například *lidar\_frame* nebo *camera\_frame*), které jsou obvykle definovány v URDF modelu robota.

Standardy pro tyto transformace jsou definovány v dokumentaci Nav2 [37], který specifikuje referenční rámce pro navigaci v ROS. Požadovaný strom transformací (*TF tree*) má strukturu `map → odom → base_link → [senzory]`. Knihovna TF2 zajišťuje synchronizované transformace mezi těmito rámcemi v reálném čase, což umožňuje konzistentní výpočet polohy robota a převod dat mezi jednotlivými rámcemi. Poziční systém (například AMCL nebo SLAM) poskytuje transformaci `map → odom`, zatímco odometrie zajišťuje transformaci `odom → base_link`. Statické transformace, jako je vztah mezi `base_link` a senzory, jsou obvykle pevně dané a definované v URDF modelu robota, což zajišťuje konzistentní geometrickou reprezentaci.

### Lokalizace a SLAM:

Transformace `map → odom` je zajišťována pozičním systémem, což může být bud' lokalizace, nebo SLAM. Nav2 nabízí několik nástrojů pro tyto účely. Pro lokalizaci na základě předem známé statické mapy je k dispozici balíček `amcl` (*Adaptive Monte Carlo Localization*), který využívá částicový filtr k odhadu polohy robota v mapě na základě senzorových dat, jako jsou LiDARová měření nebo vizuální informace. Pro dynamické mapování prostředí je k dispozici `SLAM Toolbox`, který umožňuje současnou lokalizaci a tvorbu mapy v neznámém prostředí. Tyto nástroje nejsou vázány na konkrétní typ senzoru – i když LiDAR je často preferován pro svou přesnost, lze použít i jiné senzory, jako jsou kamery nebo ultrazvukové senzory, za předpokladu, že jsou splněny standardy REP-105 a transformace jsou správně definovány.

### Odometrie:

Odometrie zajišťuje transformaci `odom → base_link` a je klíčová pro krátkodobý odhad polohy robota. Tato transformace je obvykle vypočítána na základě dat z různých zdrojů, jako jsou enkodéry kol, LiDAR, nebo IMU (*Inertial Measurement Unit*). Odometrie poskytuje kontinuální odhad polohy a orientace robota v odometrickém rámcu, avšak její přesnost je omezena kumulací chyb v důsledku šumu a nepřesnosti senzorů, například skluzu kol nebo driftu IMU. V Nav2 je odometrie nezbytnou součástí navigačního stacku, protože slouží jako základ pro lokalizaci – poziční systém (například AMCL) využívá odometrii k počátečnímu odhadu pohybu a koriguje její chyby na základě globální mapy a senzorových dat. Uživatel je odpovědný za implementaci a zajištění spolehlivého zdroje odometrie, který odpovídá požadavkům konkrétní aplikace, například pomocí fúze dat z více senzorů.

### Nákladové mapy a vrstvy:

Nákladové mapy (*costmaps*) jsou v Nav2 reprezentovány jako matice, kde každá buňka obsahuje informaci o nákladech v rozmezí od 0 do 255, což odpovídá stavu prostoru: neznámý (*unknown*), volný (*free*), obsazený (*occupied*), nebo různé stupně překážek a zón. Nav2 využívá dvě hlavní nákladové mapy – globální (*global costmap*) pro plánování dlouhodobých tras a lokální (*local costmap*) pro lokální plánování a vyhýbání se překážkám. Tyto mapy jsou dynamicky aktualizovány pomocí vrstev (*layers*), které jsou implementovány jako pluginy v knihovně `pluginlib`. Vrstvy přidávají data ze senzorů, jako jsou LiDAR, sonar nebo hloubkové kamery, a umožňují plánovači a kontroléru zohlednit překážky a další prostorové informace při výpočtu tras.

Kromě základních vrstev, jako jsou `static layer` (statická mapa), `obstacle layer` (detekce překážek) a `inflation layer` (zvýšení nákladů kolem překážek pro bezpečnostní zóny), Nav2 podporuje také pokročilé mechanismy, jako jsou filtry nákladových map. Tyto filtry umožňují anotovat mapu pomocí masky (*costmap filter mask*), což mění chování robota v určitých oblastech prostředí. Například lze definovat zóny vyloučení (*keep-out zones*), kde je

pohyb zakázán, zóny s omezením rychlosti pro bezpečnou navigaci, nebo preferované pruhy pro jízdu, což je užitečné ve výrobních halách nebo skladech. Plugin filtru transformuje data masky a aktualizuje nákladovou mapu v závislosti na poloze robota, což umožňuje dynamické přizpůsobení navigace specifickým požadavkům prostředí.

Nav2 tedy poskytuje robustní rámec pro odhad stavu a správu transformací, což je nezbytné pro efektivní a bezpečnou autonomní navigaci. Kombinace přesné lokalizace, spolehlivé odometrie a dynamických nákladových map zajišťuje, že navigační stack může být použit v širokém spektru aplikací, od experimentálních až po průmyslové.

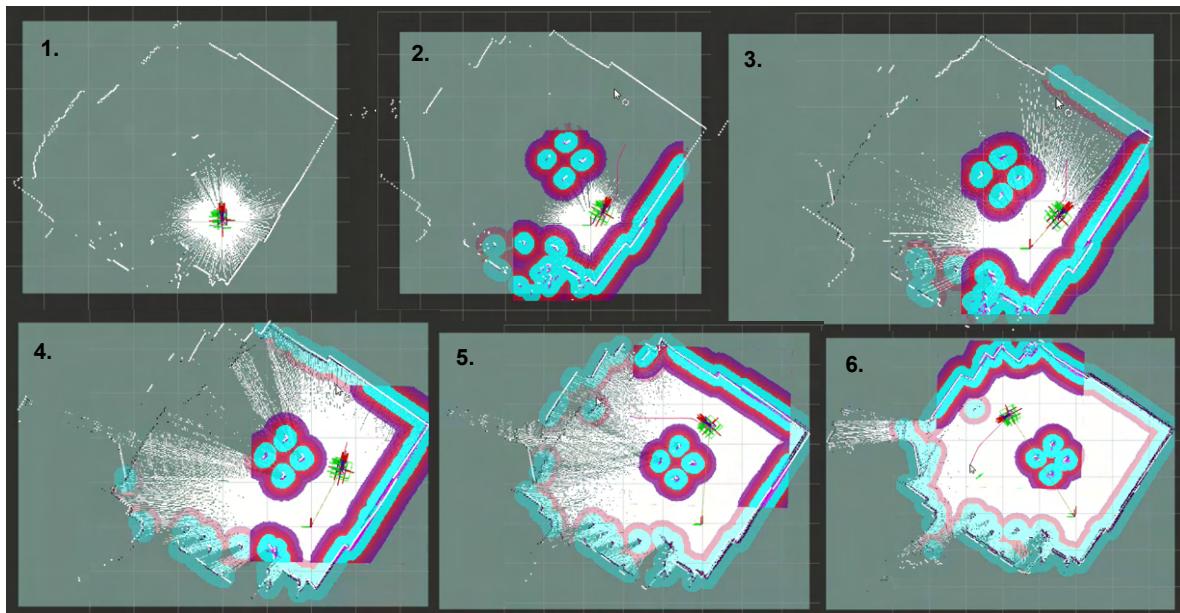
## 5.2.2 Struktura propojení subsystémů navigace

Tato kapitola se věnuje způsobu, jakým jsou jednotlivé subsystémy navigace propojeny do funkčního celku, který umožňuje autonomní navigaci. Popisuje interakce mezi základními komponentami, jako jsou plánovače, regulátory a lokalizační algoritmy apod., a objasňuje, jak jejich spolupráce zajišťuje efektivní a robustní navační systém. Celý systém je postavený na výše popsané knihovně Nav2 a jejích akčních servrech, do kterých jsem implementoval vlastní pluginy podle daného rozhraní.

### 5.2.2.1 Vytvoření mapy prostředí pomocí SLAM Toolbox

Přesná a robustní mapa prostředí je základním předpokladem pro autonomní navigaci průmyslových AGV, protože slouží jako referenční rámec pro lokalizaci a plánování trajektorií. V rámci této práce je k vytvoření 2D mapy prostředí využit SLAM Toolbox, open-source nástroj integrovaný do ROS2, který umožňuje simultánní lokalizaci a mapování [40]. SLAM Toolbox je aplikovatelný na libovolného robota vybaveného LiDARy a odometrickými senzory, jako jsou enkodéry kol a IMU (*Inertial Measurement Unit*), což zajišťuje spolehlivý sběr dat pro mapování.

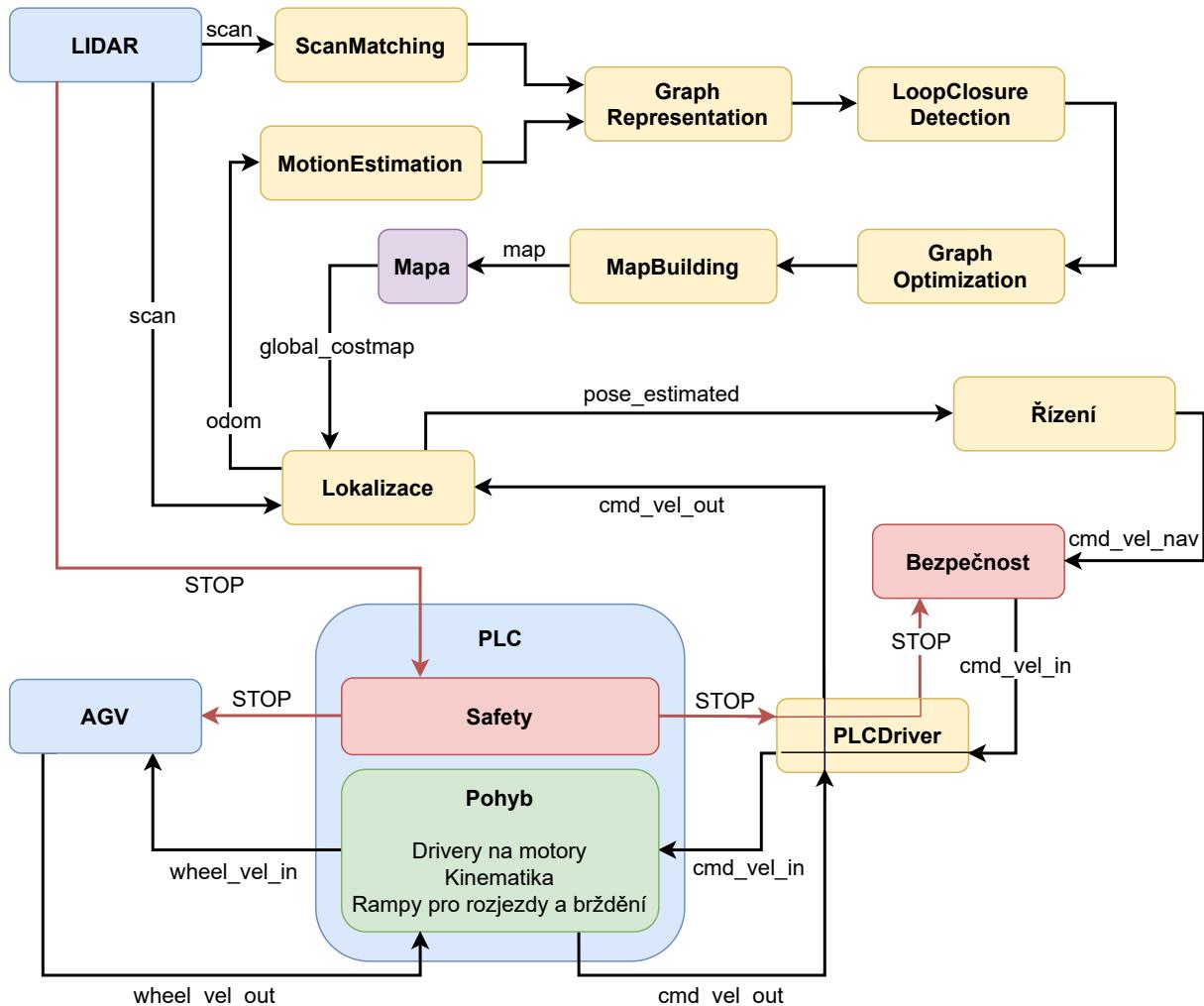
SLAM Toolbox je založen na principu grafové optimalizace, kde jsou data z LiDARů a odometrické informace modelovány jako uzly a hrany v grafu. Proces mapování probíhá v několika fázích. Nejprve jsou data z LiDARů, konkrétně bodová mračna (*PointCloud*), převedená do formátu LaserScan a odometrické informace do formátu (*Odometry*). Tato data jsou získaná z enkodérů a IMU a přijímaná přes ROS2 témata, jako /scan a /odom. Tyto údaje jsou poté integrovány do grafu, kde uzly reprezentují odhady polohy robota v jednotlivých časových okamžicích a hrany zachycují vztahy mezi těmito polohami na základě odometrických měření a korespondence LiDARových skenů. Algoritmus iterativně optimalizuje tento graf pomocí metody nejmenších čtverců, čímž minimalizuje chyby způsobené kumulací šumu v odometrii (např. skluz kol) a nepřesnostmi v LiDARových měření (např. odrazy na hranách). Výsledkem je konzistentní odhad polohy robota a současně vytvořená 2D mapa typu *OccupancyGrid* (viz obrázek 5.4).



Obrázek 5.4: Ukázká průběhu mapování pomocí SLAMToolbox.

Při uvažování obrázku 5.4 vytvořená mapa kategorizuje prostor do tří typů oblastí: obsazené buňky (překážky, např. stěny, regály černou barvou), volné buňky (bílou barvou) a neznámé buňky (neprozkoumané oblasti zelenošedou barvou). Mapa přesně reprezentuje statické prvky prostředí, jako jsou stěny, sloupy nebo regály ve skladových a výrobních prostorech, a je připravena pro použití v navigačním stacku Nav2. Rozlišení mapy je škálovatelné, typicky v řádu centimetrů (typicky 5 cm na buňku), což zajišťuje vysokou přesnost při následné lokalizaci a plánování tras. SLAM Toolbox podporuje také detekci smyček (*loop closure*), kdy algoritmus rozpozná, že robot prošel již známou oblastí, a koriguje mapu i odhad polohy, což výrazně zvyšuje konzistenci mapy při mapování větších prostor.

SLAM Toolbox je implementován jako ROS2 uzel, který efektivně komunikuje s ostatními komponentami navigačního stacku prostřednictvím ROS2 grafu (viz sekce 3.3). Výsledná mapa ve formátu *OccupancyGrid* je uložena a slouží jako vstup pro další fáze navigace, konkrétně pro lokalizaci pomocí AMCL (viz sekce 3.5) a plánování trajektorií pomocí algoritmů, jako jsou NavFn nebo MPPI (viz kapitoly 3.6 a 4.3.4). Tato mapa umožňuje autonomní pohyb AGV s vysokou přesností a bezpečností, což je důležité pro nasazení v průmyslových prostředích, kde jsou požadovány minimální odchylky polohy.



Obrázek 5.5: Schéma propojení jednotlivých komponent SLAM Toolboxu.

Schéma vnitřní struktury balíčku SLAM Toolbox, znázorněné na obrázku 5.5, ilustruje základní funkční bloky a jejich propojení při zpracování dat. Vstupní rozhraní balíčku přijímá

data z topiců `/scan` a `/odom`, která jsou předána modulu pro předzpracování dat. Tento modul filtruje a synchronizuje vstupní data, která jsou poté zpracována moduly *registrace skenů* a *odhadu pohybu*. Modul registrace skenů porovnává LiDARové skeny k určení relativního posunu, zatímco modul odhadu pohybu využívá odometrii k počátečnímu odhadu pohybu robota. Výsledky těchto modulů jsou integrovány do grafové reprezentace, kde uzly představují polohy robota a hrany zachycují relativní posuny. Modul detekce smyček identifikuje již navštívené oblasti a přidává odpovídající hrany do grafu, čímž zvyšuje konzistenci mapy. Graf je následně optimalizován modulem grafové optimalizace pomocí metody nejmenších čtverců, což poskytuje přesný odhad polohy robota. Na základě optimalizovaných poloh modul tvorby mapy generuje 2D mapu, která je spolu s transformací `map → odom` publikována výstupním rozhraním na topicy `/map`, `/map_metadata` a `/tf`. Tento tok dat zajišťuje efektivní vytvoření mapy a odhad polohy pro použití v navigačním stacku Nav2.

SLAM Toolbox tak představuje robustní a flexibilní řešení pro mapování prostředí, které je díky své open-source povaze snadno přizpůsobitelné specifickým požadavkům, jako jsou omezení hardwaru nebo kinematické vlastnosti vozíku. Jeho schopnost efektivně integrovat data z více senzorů a vytvářet přesné 2D mapy z nich činí ideální nástroj pro počáteční fázi autonomní navigace v Nav2. Kompletní průběh mapování lépe zobrazuje [Video 4](#)<sup>[1]</sup>. Na videu je mapování prováděno pomocí autonomní navigace se spuštěným navigačním stackem. To není nutné a mnohdy je výhodnější pro mapování použít pouze manuální ovládání a s robotem projet požadované prostory.

### 5.2.2.2 Lokalizace v mapě

Po vytvoření mapy je nezbytná přesná lokalizace vozíku v rámci této mapy, což zahrnuje kombinaci odometrie a adaptivního částicového filtru AMCL (viz sekce 3.5). Proces začíná výpočtem odometrie na základě dat z enkodérů a IMU, což vyžaduje využití kinematického modelu vozíku (viz sekce 4.1). Odometrie je vypočítána pomocí dopředné kinematiky:

$$\mathbf{v} = \mathbf{J} \cdot \boldsymbol{\omega},$$

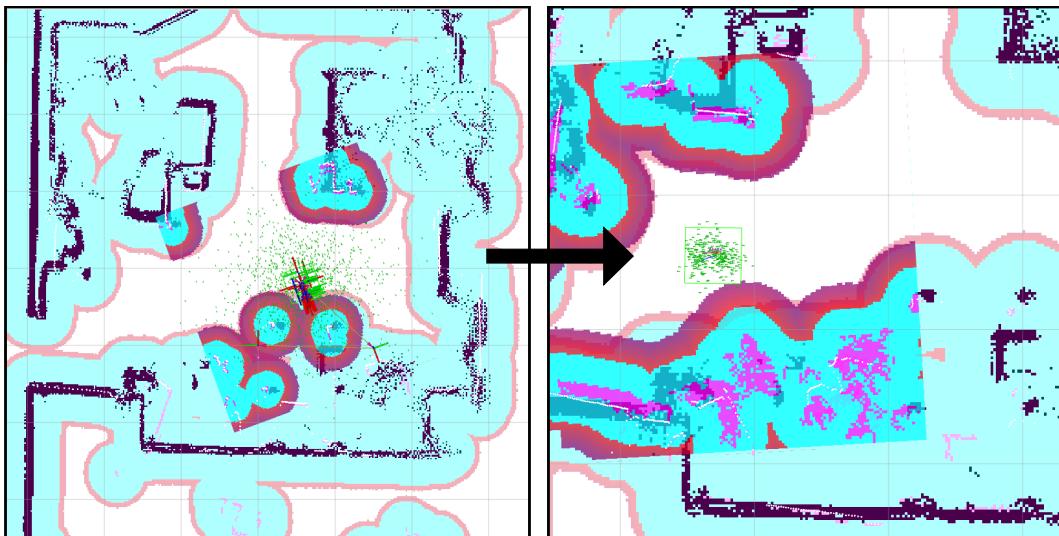
kde  $\mathbf{v}$  je vektor lineární a úhlové rychlosti vozíku,  $\boldsymbol{\omega}$  je vektor rychlostí kol, a  $\mathbf{J}$  je Jakobiho matice, která transformuje rychlosti kol do globálních souřadnic (viz sekce 4.1). Tento odhad je však náchylný na kumulativní chyby (např. šum v enkodérech), proto je zpřesňován AMCL algoritmem (viz sekce 3.5.3).

AMCL využívá odometrii jako počáteční odhad polohy a orientace vozíku a porovnává aktuální LiDARové skeny s mapou prostředí. Algoritmus využívá částicový filtr, který reprezentuje pravděpodobnostní distribuci polohy vozíku pomocí množiny částic, kde každá částice představuje možnou pozici a váhu podle shody skenu s mapou. AMCL je parametricky nastaven podle důvěryhodnosti senzorů (např. přesnosti LiDARů a enkodérů), což umožňuje dosáhnout potřebné přesnosti lokalizace a dobře kompenzovat nedostatky odometrie.

Na začátku uživatel inicializuje lokalizaci zadáním hrubého odhadu polohy v mapě (např. pomocí nástroje *2D Pose Estimate* v RViz, viz obrázek 5.6, sekce 3.9), což spouští AMCL a umožňuje poskytnout první data namísto dat z odometrie, než se robot rozjede. Výsledná lokalizace je transformována do společného rámce vozíku pomocí *tf2* knihovny (viz sekce 3.4), což zajišťuje konzistentní interakci mezi lokalizací, plánováním a řízením v reálném čase.

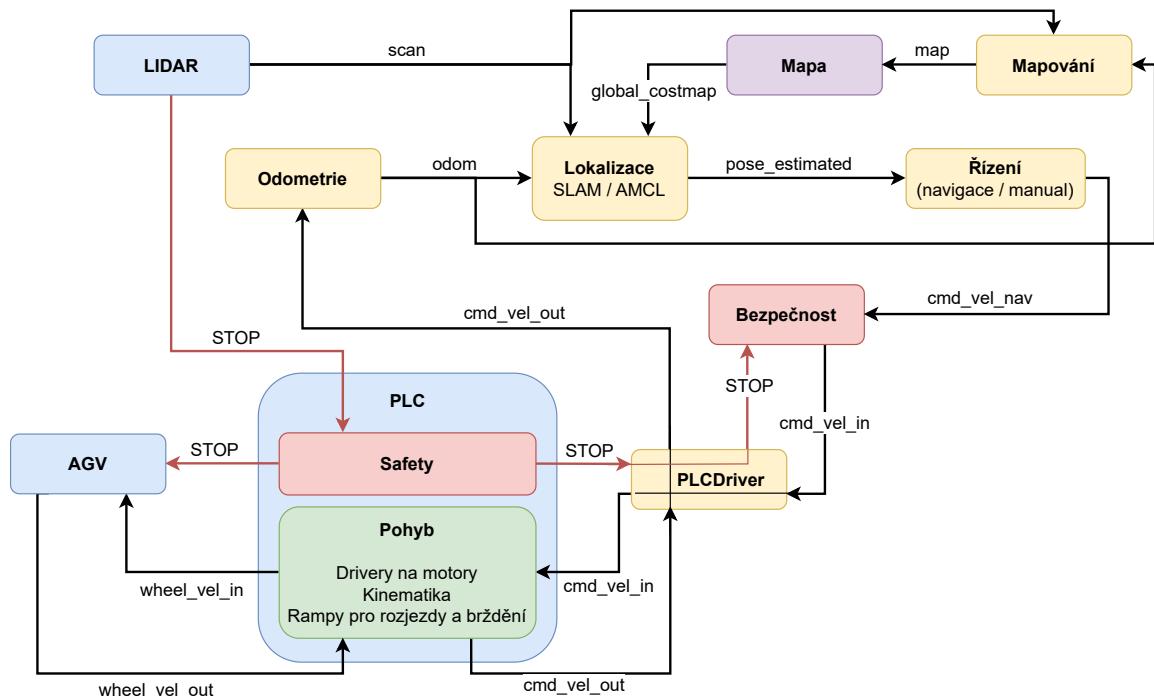
---

<sup>[1]</sup>↑ <https://youtu.be/ln4ohmL6BnM>



Obrázek 5.6: Ukázka vizualizace lokalizace v mapě pomocí nástroje RViz.

Proces lokalizace vozíku názorně demonstruje doprovodné [Video 5](#) [2], které ilustruje fungování AMCL v praxi na příkladu virtuálních drah [5.4.3](#). Video (cca od času 0:40) zachycuje inicializaci lokalizace prostřednictvím hrubého odhadu polohy vozíku v mapě, který nemusí být přesný (lze vidět velkou varianci vizualizovanou zelenými šipkami). Jakmile AMCL začne zpracovávat další data – konkrétně odhad pohybu z odometrie a aktuální LiDARové skeny, které porovnává s předem vytvořenou mapou – rychle zpřesní odhad polohy a orientace vozíku (lze vidět zmenšení variance). Tento proces ukazuje, že počáteční odhad polohy nemusí být přesný, avšak je nezbytné uvést vozík do pohybu, aby algoritmus získal dostatek relevantních dat pro korekci a dosažení přesné lokalizace v reálném čase.



Obrázek 5.7: Schéma propojení jednotlivých komponent lokalizace.

[2] ↑ <https://youtu.be/Ek70PZHnybw>

Schéma procesu lokalizace, znázorněné na obrázku 5.7, nezahrnuje uživatelský vstup prostřednictvím nástroje *2D Pose Estimate*. Tento nástroj není nezbytný, protože AMCL může teoreticky odhadnout polohu vozíku i bez počáteční inicializace, pokud je robot uveden do pohybu. Algoritmus přitom využívá historii pohybu z odometrie a odpovídající LiDARové skeny (*LaserScan*), které porovnává s mapou typu *OccupancyGrid*, aby určil polohu vozíku v rámci celé mapy. Tento přístup však s rostoucí velikostí mapy zvyšuje výpočetní náročnost a zpomaluje proces lokalizace. Navíc v prostředích s opakujícími se vzorci, jako jsou kanceláře nebo výrobní haly, může AMCL nesprávně lokalizovat vozík na jiné místo, než kde se skutečně nachází. Z těchto důvodů je doporučeno zadat přibližnou počáteční polohu, což výrazně zvyšuje rychlosť a spolehlivost lokalizace. Alternativně lze místo AMCL použít SLAM algoritmus, který rovněž poskytuje přesný odhad polohy. Většina aplikací však nevyžaduje současné mapování, protože cílem je obvykle jednorázově vytvořit mapu a poté se v ní pouze lokalizovat a navigovat. SLAM je navíc výpočetně náročnější a mohl by do mapy zahrnovat dočasné překážky, jako jsou odložené palety, osoby, nebo stojící AGV, což by narušilo konzistenci mapy pro dlouhodobou navigaci.

### 5.2.2.3 Navigace a plánování trajektorie

Navigace jako taková zahrnuje globální a lokální plánování trajektorií, které zajišťují efektivní a bezpečný pohyb vozíku. Globální plánování je realizováno pomocí NavFn algoritmu (viz sekce 3.6) v rámci Nav2, který generuje optimální trajektorii přes celou mapu na základě nákladové mapy (*costmap*) a cílové pozice určené uživatelem. Trajektorie je ohodnocena podle nákladů (např. vzdálenosti) a respektuje statické překážky zanesené v mapě. Tato trajektorie se přeplánuje pouze v případě zjištění neprůjezdnosti trasy, například zavřených dveří, které byly při mapování otevřené, nebo nově objevených překážek.

Lokální plánování, realizované pomocí MPPI (viz sekce 4.3.4), neustále monitoruje okolí vozíku (např. oblast 5x5 m) na základě aktuálních LiDARových dat a odometrie. Tento modul dynamicky přizpůsobuje trajektorii neočekávaným překážkám, jako jsou pohyblivé objekty (např. lidé) nebo nově zjištěné překážky, které nejsou zaneseny v mapě. MPPI generuje řadu možných trajektorií, hodnotí je podle nákladové funkce (zahrnující vzdálenost od aktuálního cíle, překážky a dynamiku vozíku) a vybírá tu nejlepší. Pokud lokální plánovač nenalezně možnost objetí překážky, signalizuje globálnímu plánovači potřebu přeplánování. V případě, že ani globální plánovač nenalezně alternativní trasu, vozík zastaví a čeká, dokud překážka nezmizí. Po určité době navigace selže a je ukončena.

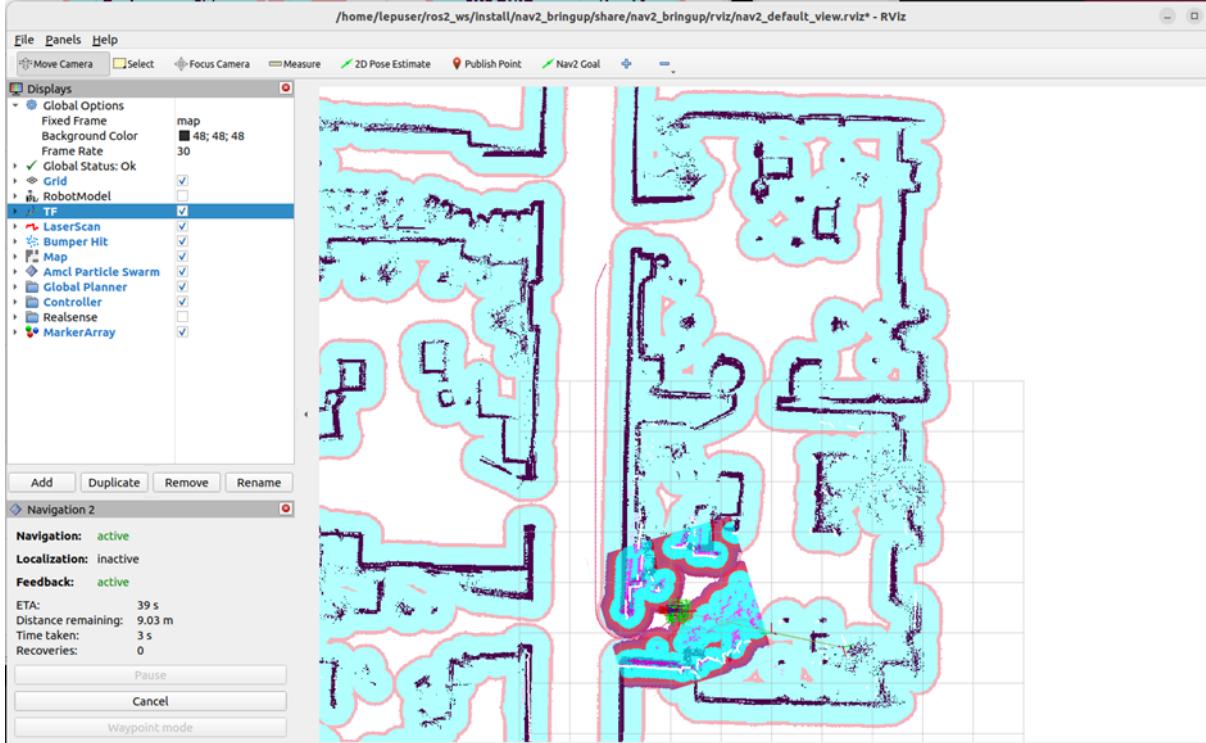
Řízení pohybu je nastaveno tak, aby vozík dodržoval rozjezdové a brzdné rampy s ohledem na možný náklad a různé povrchy, což minimalizuje nárazy a zajišťuje plynulý pohyb v souladu s normou ISO 3691-4, která vyžaduje klidný pohyb i z toho důvodu, aby se zabránilo obavám lidí v blízkosti vozíku. Řízení v podstatě odpovídá lokálnímu plánovači a je tedy zajišťováno pomocí MPPI algoritmu, který generuje řídící vstupy (rychlosti pro jednotlivá kola) na základě cílové trajektorie z plánovače. Tyto vstupy jsou transformovány pomocí inverzní kinematiky (viz sekce 4.1):

$$\omega = \mathbf{J}^{-1} \cdot \mathbf{v},$$

kde  $\omega$  je vektor rychlostí kol,  $\mathbf{v}$  je požadovaná lineární a úhlová rychlosť vozíku, a  $\mathbf{J}^{-1}$  je inverzní Jakobiho matice. Rychlosti kol jsou zpětně měřeny enkodéry a použity pro výpočet odometrie během jízdy, což aktualizuje pozici vozíku v mapě a zajišťuje konzistentní transformaci v rámci tf2.

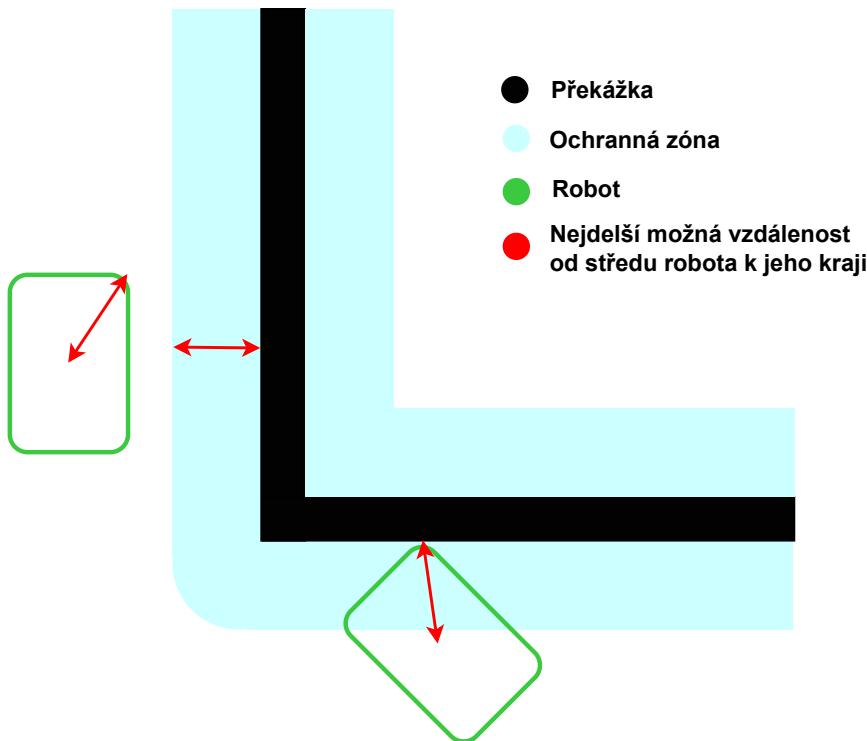
Trajektorie generovaná globálním plánovačem NavFn je navržena tak, aby byla co nejkratší, energeticky úsporná a minimalizovala počet zatáček. Tento přístup je zřejmý například na ob-

rázku 5.8, kde je znázorněn průběh navigace úzkou chodbou s preferencí přímého směru. Trajektorie je optimalizována s ohledem na váhy parametrů generátoru, jako je délka trasy, vzdálenost od překážek a energetická náročnost, což zajišťuje, že robot neprochází zbytečně středem volného prostoru a minimalizuje potenciální kolize s jinými subjekty v prostředí.



Obrázek 5.8: Příklad přímé trajektorie generované algoritmem NavFn v úzké chodbě.

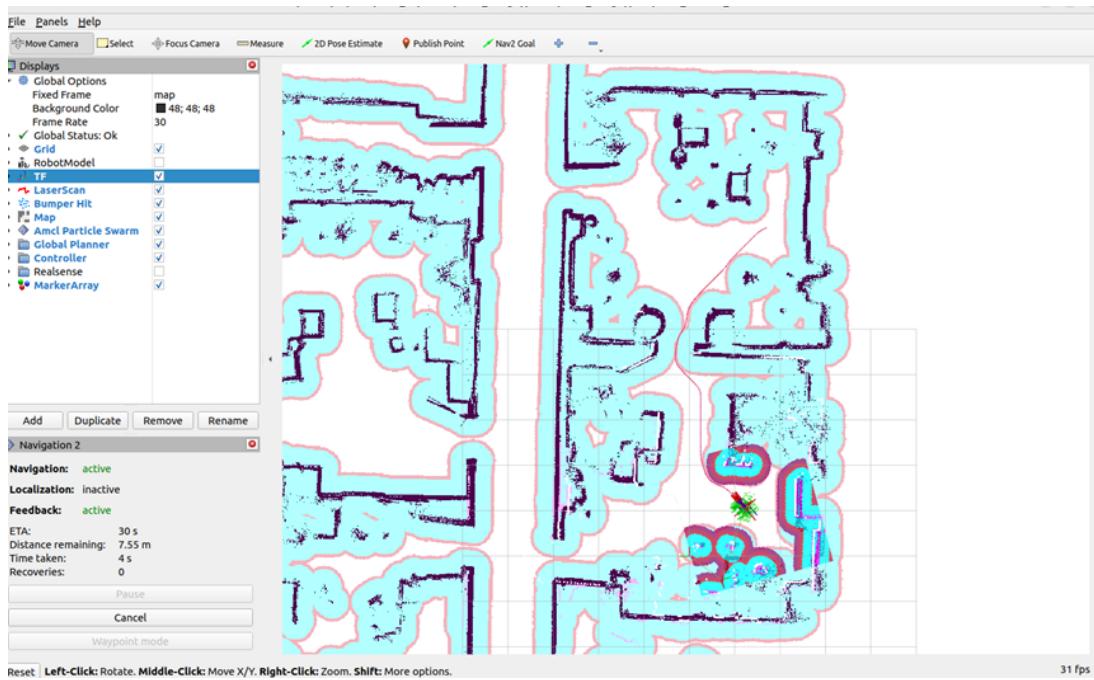
Bezpečnostní zóny, definované jako rozšíření překážek o poloměr robota (viz obrázek 5.9), hrají klíčovou roli při určování průjezdnosti trasy. Jsou zde proto, že robot je reprezentován pouze hmotným bodem. Tyto zóny efektivně zvětšují zakázané oblasti kolem překážek, čímž je zajištěno, že střed robota sleduje trajektorii bez rizika kolize jeho okrajů s okolními objekty. Na obrázku 5.8 je patrná širší růžová zóna kolem překážek, která představuje doporučenou bezpečnostní rezervu pro plánovač. Tato zóna indikuje prostor, do kterého by plánovač ideálně neměl vést trajektorii, avšak v případě, že neexistuje jiná možnost, je vjetí do této oblasti povoleno. V praxi to znamená, že v otevřených prostorech si robot udržuje větší odstup od překážek, zatímco v úzkých prostorech může projet blíže k nim. V takových situacích však mohou zasáhnout bezpečnostní mechanismy, jako jsou bezpečnostní zóny nebo modul *Collision Monitor* (viz sekce 5.1), které omezí rychlosť robota, aby byla zajištěna bezpečná navigace.



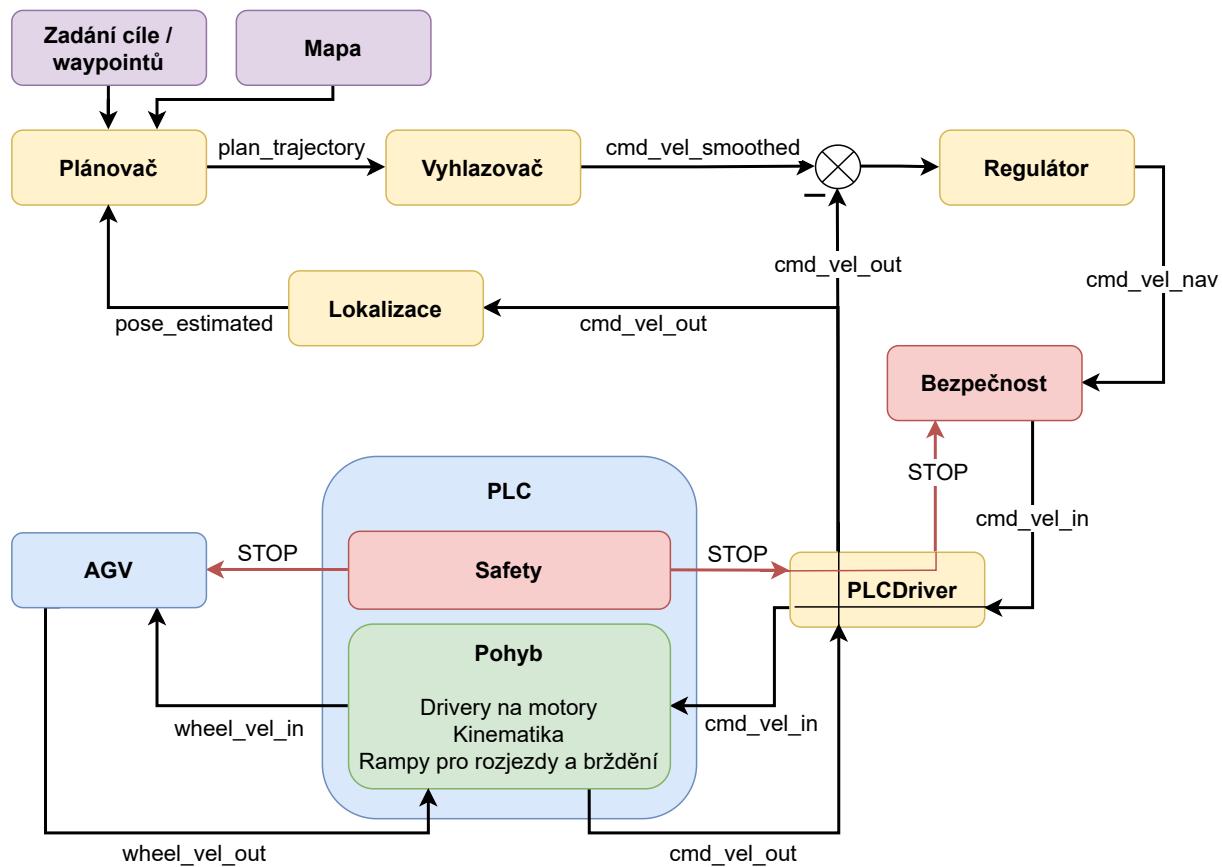
Obrázek 5.9: Rozšíření překážek o bezpečnostní zónu odpovídající poloměru robota, zajišťující bezkolizní navigaci.

Po naplánování je trajektorie vyhlazována, aby byla zajištěna plynulost pohybu a minimální poloměr zatáček v souladu s kinematickými možnostmi robota. Podvozky použité v této práci umožňují otáčení na místě díky diferenciálnímu nebo omnidirekčnímu pohonu, což eliminuje potřebu složitých manévrů na konci trasy – například finální natočení do cílové orientace probíhá efektivně bez zbytečných oblouků. Minimální poloměr zatáček je určen fyzickými vlastnostmi robota, avšak generátor trajektorie je nastaven tak, aby upřednostňoval co nejpřímější trasy a vyhýbal se ostrým změnám směru (např. pravým úhlům), které by vyžadovaly zastavení, otočení a opětovný rozjezd. Takové situace by byly neefektivní z hlediska času i energie a nesplňovaly by požadavek na optimální trasu. Obrázek 5.10 ilustruje, jak jsou zatáčky řešeny s ohledem na bezpečnostní zóny, které rozšiřují překážky a omezují průjezdné prostory.

Je třeba zdůraznit, že trajektorie v Nav2 není definována jako dlouhý seznam bodů, nýbrž jako sled vektorů okamžitých rychlostí, které zahrnují lineární a úhlovou složku. Vyhlažovač proto neoptimalizuje pouze geometrickou hladkosť trajektorie, ale také plynulost změn rychlosti, tedy zrychlení, aby byla zajištěna plynulá jízda robota bez náhlých trhavých pohybů. Výsledná trajektorie tak obsahuje ideální posloupnost rychlostí, které jsou postupně předávány regulátoru robota k vykonání. V praxi však řízení není zcela přesné, a to i přes zahrnutí definice parametrů robota jako je maximální přípustné zrychlení nebo omezení rychlosti. K zajištění přesnosti je proto využita zpětná vazba, která zahrnuje kontrolu skutečně naměřené rychlosti robota a jeho polohy určené lokalizací (viz sekce 3.5). Na základě odchylky mezi požadovanou a skutečnou trajektorií plánovač přepočítává nové hodnoty rychlostí, čímž zajišťuje, že robot sleduje globální trajektorii co nejlépe. Toto propojení je znázorněno na obrázku 5.11.

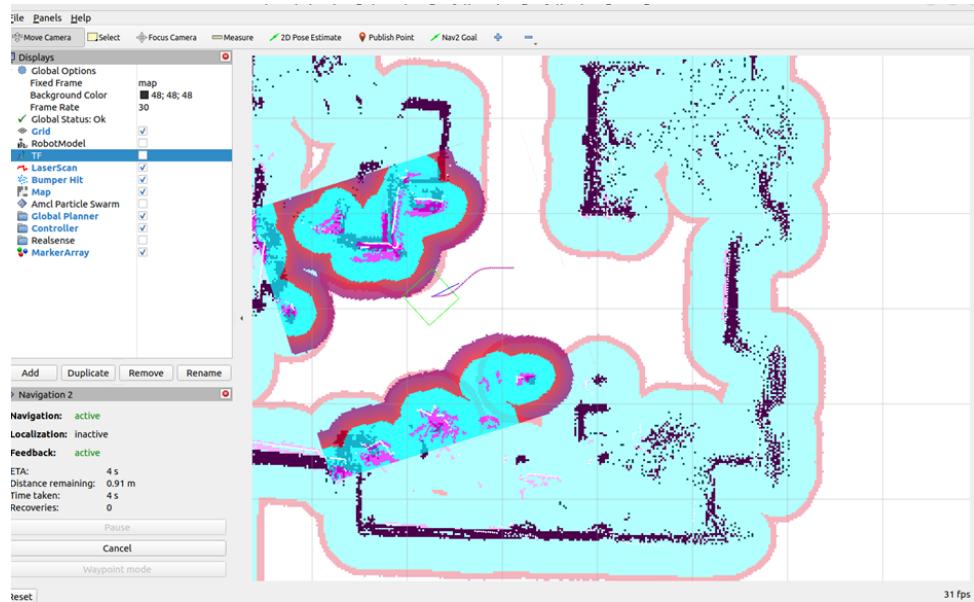


Obrázek 5.10: Vyhlašení zatáček s ohledem na minimální poloměr otáčení a bezpečnostní zóny.



Obrázek 5.11: Schéma propojení jednotlivých komponent pro plánování a sledování trajektorie.

Lokální plánování pomocí MPPI dále zohledňuje prediktivní řízení, jehož příklad je znázorněn na obrázku 5.12. MPPI vyhledává optimální bod na trajektorii v rámci plovoucího horizontu (tzv. *look-ahead distance*) a generuje k němu optimální cestu s ohledem na aktuální překážky a dynamiku vozíku. Velikost predikčního horizontu je kritickým parametrem. Příliš velký horizont může v ostrých zatáčkách vést k nevhodným trajektoriím, které by procházely přes překážky, čemuž je sice zabráněno bezpečnostními omezeními, ale může to způsobit oscilace v řízení v důsledku častých změn strategie. Naopak příliš malý horizont redukuje prediktivní schopnosti algoritmu a může vést k nestabilnímu chování, jako jsou nepředvídatelné výkyvy v pohybu. Optimální nastavení horizontu proto vyžaduje pečlivou kalibraci, aby byla zajištěna plynulost a bezpečnost navigace.



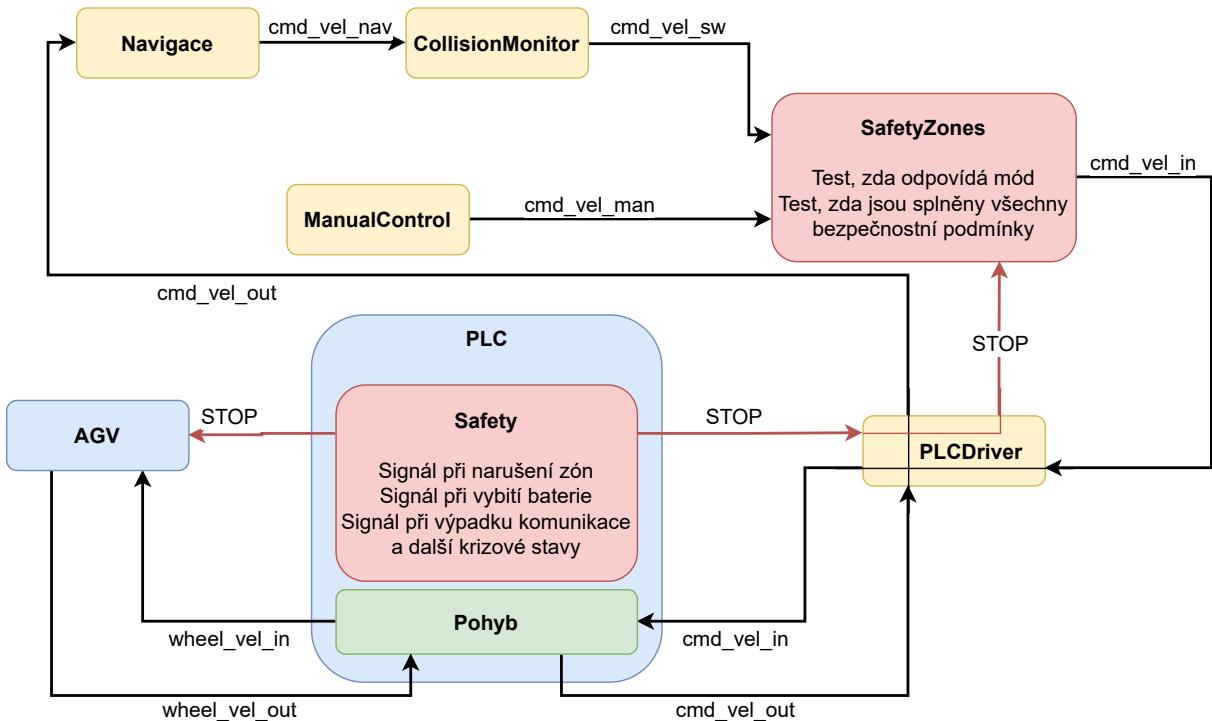
Obrázek 5.12: Příklad prediktivního řízení s MPPI v prostředí s překážkami, ukazující výběr optimální cesty k bodu na trajektorii.

Tento kombinovaný přístup globálního a lokálního plánování zajišťuje, že trajektorie je nejen efektivní a bezpečná, ale také přizpůsobivá reálným podmínkám, což je zásadní pro robustní autonomní navigaci vozíku v dynamickém prostředí.

#### 5.2.2.4 Bezpečnostní mechanismy

Bezpečnost autonomní navigace je zajištěna implementací bezpečnostních zón (viz sekce 5.1), které monitorují okolí vozíku a reagují na potenciální kolize. Hardwarové řešení využívá certifikované LIDARy (RSL200, RSL400) s OSSD výstupy, které signalizují narušení zón přímo do PLC (např. Simatic S7-1500, viz sekce 5.1.1). Tento signál vede k okamžitému zastavení motorů, čímž se zabrání kolizi, přičemž informace o situaci je následně předána navigačnímu stacku v ROS2.

Softwarové řešení zahrnuje Collision Monitor z Nav2 (viz sekce 5.1.2), který neustále analyzuje data z LIDARů a odometrie pro detekci překážek v definovaných zónách. Kombinace hardwarového (PLC) a softwarového (Collision Monitor) přístupu poskytuje redundanci a maximalizuje bezpečnost, přičemž minimalizuje riziko kolize i v dynamických prostředích, jako jsou skladové haly s pohyblivými lidmi nebo překážkami. Schéma propojení a vzájemného působení těchto komponent je znázorněno na obrázku 5.13.



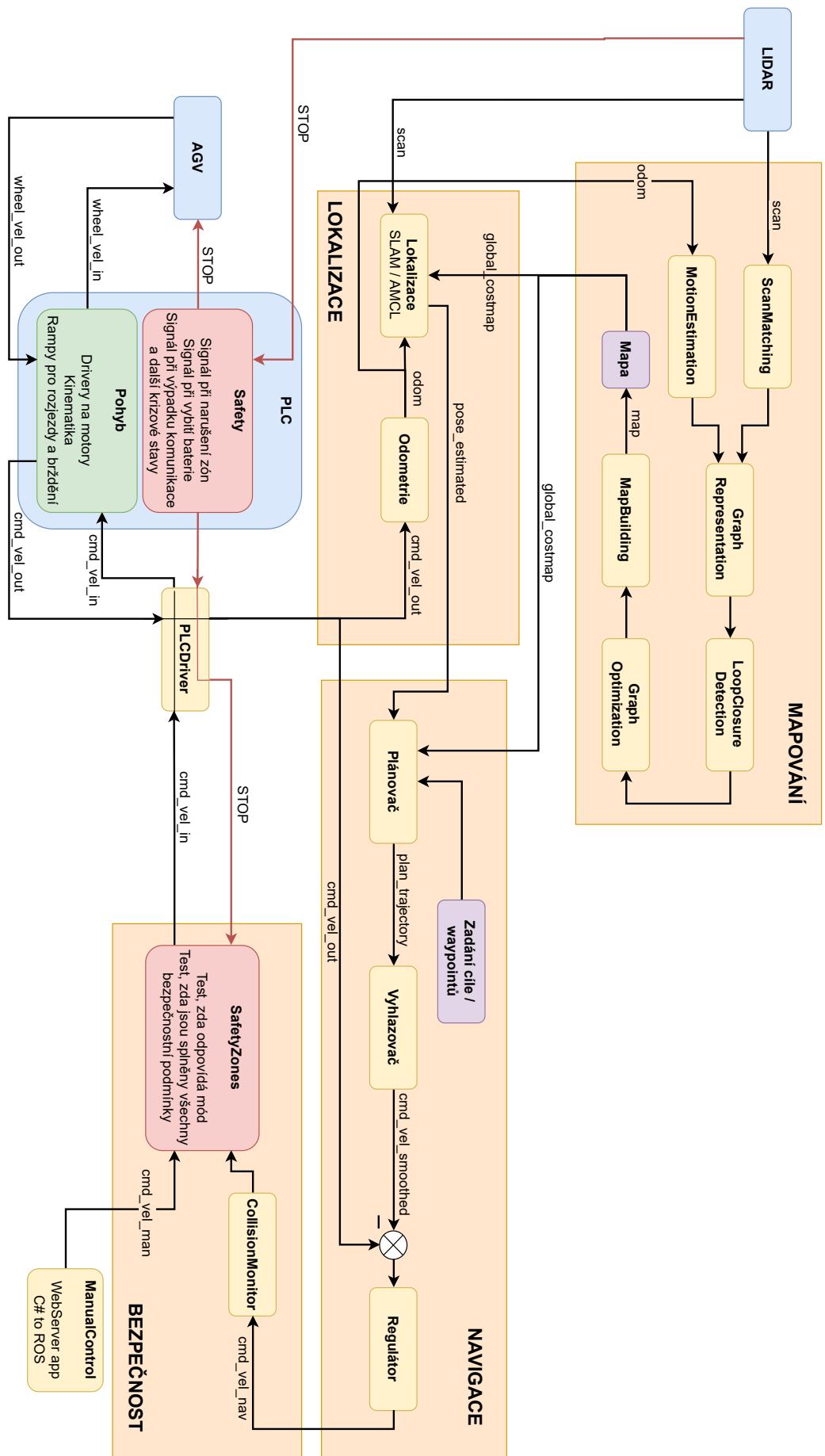
Obrázek 5.13: Schéma propojení jednotlivých SW komponent bezpečnostních prvků.

### 5.2.2.5 Integrovaný proces navigace

Schéma procesu autonomní navigace v Nav2, znázorněné na obrázku 5.11, ilustruje propojení základních komponent pro mapování, lokalizaci, plánování, řízení a zajištění bezpečnosti AGV. Proces začíná mapováním prostředí pomocí SLAM Toolbox, který využívá LiDARová data a odometrii k vytvoření 2D map. Podrobná vnitřní struktura SLAM Toolbox je znázorněna v horní části schématu a zahrnuje moduly pro registraci skenů, odhad pohybu, grafovou reprezentaci, detekci smyček a tak dále, jak je popsáno v kapitole 5.2.2.1. Výsledná mapa je poté využita pro lokalizaci a plánování.

Lokalizace vozíku kombinuje odometrická data a LiDARové skeny k přesnému odhadu polohy vozíku v mapě, jak je popsáno v kapitole 3.5. Odometrický modul poskytuje nepřetržitý odhad pohybu na základě dat z pohonů vozíku pomocí kinematického modelu. Globální trajektorie je plánována na základě mapy a zadaných cílových bodů, přičemž výsledná trajektorie je optimalizována vyhlazovačem z hlediska hladkosti křivky a plynulosti rychlosti. Regulátor poté převádí tuto trajektorii na příkazy rychlosti, které jsou předány AGV prostřednictvím PLC driveru.

Bezpečnost navigace je zajištěna moduly *SafetyZones* a *CollisionMonitor*, které nakládají s informacemi o krizových stavech. PLC dále zpracovává signály, jako je narušení bezpečnostní zóny, výpadek komunikace atd., a předává odpovídající příkazy pohonům. AGV může být alternativně řízeno manuálně přes pomocí webové aplikace. Celkový systém byl validován v simulačním prostředí Gazebo (viz sekce 3.8) a při reálném nasazení na několika robotech, kde byla potvrzena jeho spolehlivost, plynulost pohybu i soulad s normami ISO 13849 a ISO 3691-4.



Obrázek 5.14: Kompletní schéma propojení všech komponent autonomní navigace.

## 5.2.3 Aplikace na různých platformách

Tato kapitola se zaměřuje na popis implementace a testování autonomní navigace na různých hardwarových platformách, které slouží k ověření navaigačních algoritmů v reálných podmínkách. Cílem je využít specifika jednotlivých podvozků, jejich senzorickou výbavu a integraci s ROS2, aby bylo možné posoudit jejich vhodnost pro různé scénáře autonomního pohybu. Tyto informace poskytují ukázku toho, jak bylo postupováno při vývoji.

### 5.2.3.1 Podvozek DJI

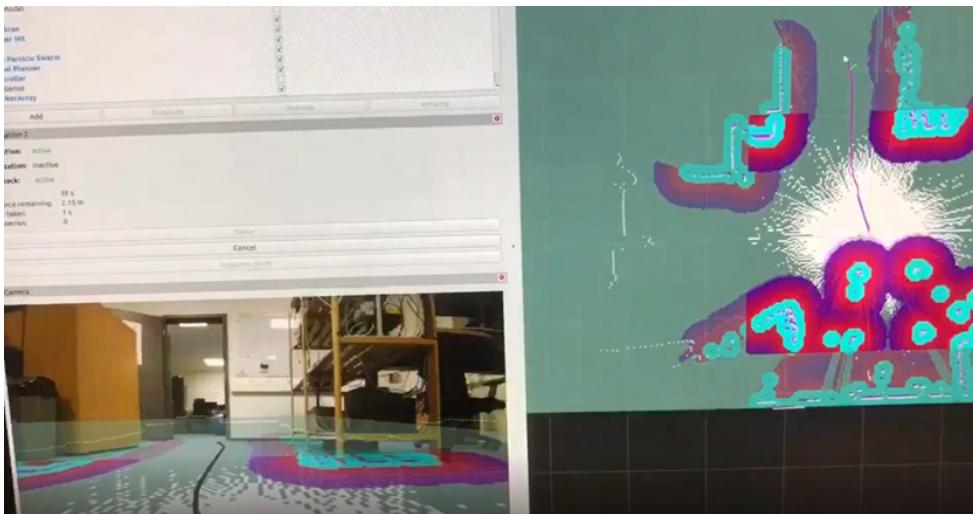
Podvozek DJI (viz sekce 2.2.1) slouží jako první testovací platforma pro ověření navaigačních algoritmů na reálném zařízení mimo simulační prostředí. Navzdory své počáteční roli v ladění algoritmů disponuje tento podvozek kvalitní senzorickou výbavou, zejména enkodéry kombinovanými s inerciální měřicí jednotkou (IMU), což zajišťuje poměrně přesnou odometrii pro výpočet polohy a orientace. Tato vlastnost umožňuje efektivní testování lokalizačních algoritmů v reálných podmínkách. Naopak omezení se projevuje při přesném pozicování, protože motory podvozku vyžadují minimální prahovou rychlosť pro aktivaci pohybu. Tato charakteristika znemožňuje jemné posuny o malé vzdálenosti nebo provoz při velmi nízkých rychlostech, což ovlivňuje přesnost dosažení cílových bodů v některých scénářích.

Řídící systém podvozku je integrován do ROS2 prostřednictvím výpočetní jednotky Raspberry Pi 4, která primárně zpracovává senzorická data a komunikuje s hardwarem. Na této jednotce probíhá pouze základní předzpracování signálů, zatímco kompletní výpočty spojené s navigací, plánováním trajektorií a zpracováním dat deleguje na výkonnější linuxový počítač připojený přes Wi-Fi síť. Díky standardizované komunikaci v ROS2 je propojení mezi zařízeními bezproblémové, avšak latence způsobená přenosem dat přes Wi-Fi mírně ovlivňuje reakční dobu systému, což se projevuje zejména při dynamických manévrech na zpožděných transformacích naměřených dat. Pro komunikaci s podvozkem se využívá existující ROS2 driver [41], což eliminuje potřebu jeho vývoje a umožňuje zaměřit se výhradně na testování navaigačních algoritmů. Kromě toho je pro podporu LIDARů vytvořen doplňkový driver v ROS2, který sbírá UDP data z těchto senzorů a publikuje je ve standardizovaném formátu do příslušných topiců. Tento driver umožňuje integraci LIDARů do navaigačního systému. Kromě autonomní navigace byl na tomto podvozku ověřen i modul manuálního ovládání vyvinutý kolegou Janem Holubem [27]. Z tohoto důvodu byl připravem i mechanismus pro rozlišení autonomního a manuálního režimu.

Pro zajímavost je na obrázku 5.15 znázorněna fúze lidarových dat, která slouží k vytvoření mapy prostředí a definici bezpečnostních zón kolem detekovaných překážek, s obrazem z kamery, do něhož se tato data promítají. Tato vizualizace demonstruje integraci senzorických informací pro zlepšení situačního povědomí a robustnosti navigace. Kromě toho jsou k dispozici rovněž videoukázky dokumentující širokou škálu experimentů provedených na tomto podvozku. Ty zahrnují autonomní navigaci na základě jednoho cílového bodu, navigaci podle posloupnosti waypointů propojených jednou spojitou trajektorií, navigaci s waypointy vyžadujícími povinný průjezd specifickým bodem, dále dynamické objíždění překážek, které se v prostředí neočekávaně objeví, vyhýbání se překážkám v reálném čase a aktualizaci mapy prostředí včetně odstranění překážek, které byly fyzicky odstraněny. Tyto testy dokazují adaptabilitu a funkčnost navaigačních algoritmů v různých scénářích. To vše podrobněji znázorňuje Video 6 [3], kde je vše popsáno přímo u konkrétních ukázk.

---

[3] ↑ <https://youtu.be/FsqOBiu9Me4>



Obrázek 5.15: Ukázka fúze LIDARu a kamery v ROS2.

### 5.2.3.2 Diferenciální podvozek

Diferenciální podvozek (viz sekce 4.1.1) slouží k ověření navigačních algoritmů v situaci, kdy senzorická data a výstupy na motory procházejí skrze PLC, což přináší specifické výzvy. Integrace PLC do řídicího řetězce způsobuje mírná zpoždění v přenosu dat kvůli přidané vrstvě zpracování, ale zároveň umožňuje přímé zpracování signálů ze senzorů (např. LIDARu a enkodérů) přímo na úrovni PLC. Tento přístup se využívá k testování bezpečnostních funkcí (safety), zejména schopnosti vozíku včas zastavit při detekci překážky v bezpečnostních zónách. Bezpečnostní algoritmy respektují dynamická omezení podvozku a zajišťují soulad s normami pro autonomní vozidla, jako je ISO 3691-4.

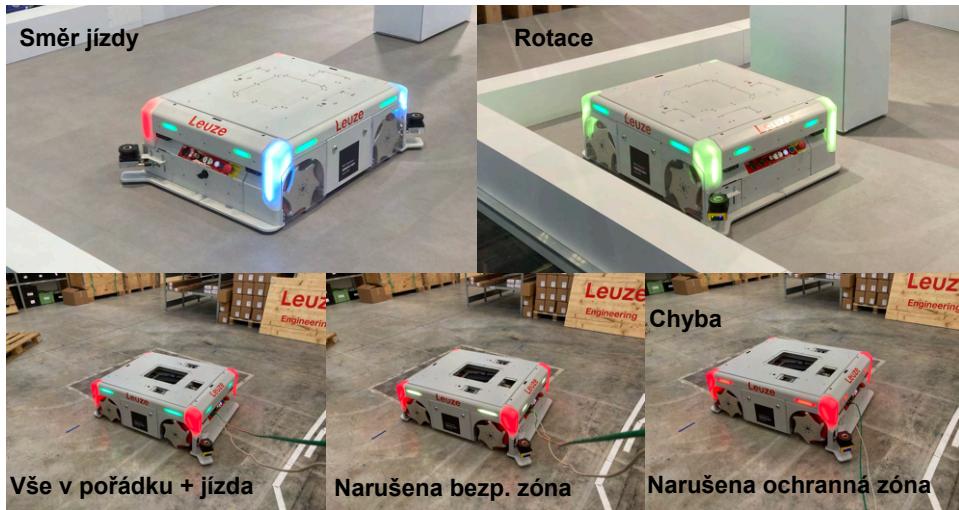
Na tomto podvozku se vyvíjí a testuje vlastní ROS2 driver, který zahrnuje kompletní obsluhu hardwaru – od zpracování odometrických dat až po řízení motorů. Tento vývoj představuje významný posun oproti přístupu na podvozku DJI, kde se využívá hotový driver, a umožňuje detailní optimalizaci komunikace mezi ROS2 a PLC. Po počátečním nasazení s Raspberry Pi 4 a externím výpočetním počítacem se testuje alternativa s Raspberry Pi 5 jako samostatnou jednotkou pro kompletní zpracování navigace. Výsledky ukazují, že Raspberry Pi 5 poskytuje dostatečný výkon pro reálné nasazení, včetně podpory vzdálené vizualizace přes RViz, a to při výrazně nižších nákladech ve srovnání s průmyslovým počítacem (IPC). Pro další zlepšení se plánuje implementace prediktivního filtrování dat z enkodérů na úrovni ROS2, což může kompenzovat latenci PLC a zvýšit přesnost odometrie.

Pro zajištění komunikace mezi ROS2 a PLC se využívá knihovna Snap7 [42], která funguje na bázi protokolu TCP. Tato knihovna umožňuje obousměrný přenos dat mezi řídicím systémem PLC a výpočetní jednotkou hostující ROS2 prostřednictvím standardního síťového spojení. Snap7 implementuje rozhraní pro čtení a zápis dat z/do paměti databloků PLC, přičemž komunikace probíhá formou strukturovaných zpráv obsahujících například vektor rychlostí ( $v_x, v_y, \omega$ ), stavové informace nebo kontrolní hodnoty. Získané zprávy se v ROS2 převádějí do standardizovaného formátu zpráv (např. typu *geometry\_msgs/Twist* pro rychlosti) a publikují do příslušných topiců, jako je */cmd\_vel* pro řízení pohybu. Tento přístup zajišťuje bezproblémovou integraci průmyslového řízení s ROS2 ekosystémem, přičemž TCP protokol poskytuje spolehlivý a rychlý přenos dat i přes potenciální latenci síťové komunikace.

### 5.2.3.3 AGV s velkým všesměrovým podvozkem

Tento podvozek všesměrového typu představuje finální testovací platformu, která splňuje veškeré požadavky pro plnohodnotné nasazení autonomní navigace v reálných podmínkách. Disponuje vysokou kvalitou motory s podporou ProfiSafe komunikace přes PLC, což zajišťuje přesný pohyb i při nízkých rychlostech a eliminuje omezení pozorovaná u podvozku DJI. Kodovány integrované do motorů poskytují precizní odometrická data, díky čemuž je lokalizace v mapě vysoko přesná, což je nezbytné pro robustní navigaci v dynamických prostředích. Celý řídicí systém běžící v ROS2 je úspěšně nasazen na Raspberry Pi 5, což představuje ekonomicky výhodné řešení při zachování dostatečného výkonu pro zpracování dat v reálném čase. Tato jednotka zvládá nejen výpočty navigace a komunikaci s PLC, ale i vizualizaci přes RViz, která je dostupná přes síť pomocí technologie *noVNC* pro vzdálenou plochu. Dále na Raspberry Pi 5 běží webová aplikace umožňující manuální ovládání, přepínání mezi autonomním a manuálním režimem a zadávání povelů robotovi, což zvyšuje uživatelskou flexibilitu.

PLC řídí hardwarové funkce, jako jsou dynamické změny bezpečnostních zón a světelná signalizace (viz obrázek 5.16 a Video 7 [4]), navržená v souladu s průmyslovými normami. Signalizace zřetelně indikuje stav vozítka (připravenost, pohyb, chyba) a směr pohybu – například blikáním světel v rozích podvozku při rotaci nebo trvalým svícením ve směru pohybu, kde se rozšiřují bezpečnostní zóny. Dopředná a inverzní kinematika je implementována přímo v PLC, což odděluje hardwarovou parametrizaci (např. rozměry podvozku, poloměry kol) od softwarového řízení v ROS2. Tento přístup zajišťuje vysokou přenositelnost řídicího systému mezi různými podvozky, protože ROS2 komunikuje s PLC pouze prostřednictvím vektoru rychlostí ( $v_x, v_y, \omega$ ) a zpětné vazby o aktuálním pohybu. Kromě rychlostí se předávají doplňující informace, jako je inkrementující se hodnota pro kontrolu kontinuity komunikace a identifikace režimu (autonomní/manuální), což ovlivňuje nastavení bezpečnostních zón a povolených rychlostí – například manuální režim využívá užší zóny a nižší rychlostní limity pro zvýšenou bezpečnost obsluhy.



Obrázek 5.16: Ukázka funkce světlené signalizace.

Softwarové řešení je navrženo s důrazem na modularitu a univerzálnost, což umožňuje jeho snadné nasazení na různé typy AGV bez nutnosti zásadních úprav. Kombinace výkonného hardwaru, efektivního PLC a kompaktního ROS2 systému na Raspberry Pi 5 představuje optimální

[4] ↑ <https://youtu.be/N6Oc99miTk4>

kompromis mezi výkonem, cenou a funkčností vhodný jak pro experimentální testování, tak pro průmyslové aplikace. Následující videoukázky shrnují testy provedené na AGV s velkým všesměrovým podvozkem a snaží se ukázat, jak spolu kooperují jednotlivé subsystémy jako je aplikace, RViz a samotné AGV. Nejprve základ, kterým je manuální ovládání. To je u takto velkého stroje naprostou nezbytností, zároveň je velice vhodné pro mapování a tak dále. Průběh manuálního ovládání tedy ukazuje [Video 8](#)<sup>[5]</sup>. Následně byl připraven scénář, kde bezpečnostní zóny byly příliš velké ve srovnání s parametry navigace. Z toho důvodu AGV naplánuje objetí překážky podle špatných parametrů a dojde k narušení bezpečnostních zón. Jak ukazuje [Video 9](#)<sup>[6]</sup>, AGV skutečně zastaví a brzy po odstranění překážky se opět rozjede, aby dokončilo misi. Poslední [Video 10](#)<sup>[7]</sup> ukazuje druhý případ, a to že je vše nakonfigurováno správně. Při nalezení překážky, která v mapě není, AGV přeplánuje trasu a překážku bezpečně objede tak, aby se dostalo na požadovanou pozici s požadovanou orientací. Videá znázorňují reálné použití povelování z aplikace i RVizu pro ovládání a zadávání pokynů.

---

[5] ↑ <https://youtu.be/BV9gnHkLGfw>

[6] ↑ <https://youtu.be/Bhdc4WxN1-g>

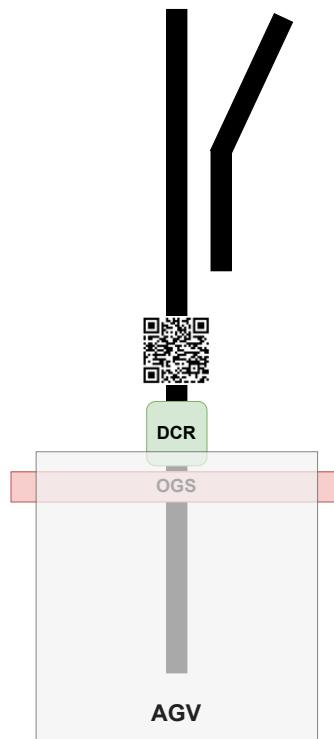
[7] ↑ [https://youtu.be/5y\\_YiSTZgvs](https://youtu.be/5y_YiSTZgvs)

## 5.3 Jízda po čáře

Tato kapitola se zabývá implementací a testováním navaigacního systému pro AGV založeného na sledování čáry pomocí optického senzoru OGS (*Optical Guidance Sensor*) a čtení QR kódů senzorem DCR (*Data Code Reader*), doplněného o bezpečnostní prvky využívající LiDAR. Cílem je analyzovat možnosti tohoto přístupu v kontextu průmyslových požadavků, zejména s ohledem na možnosti použití výše zmíněných senzorů, bezpečnost, efektivitu a zjednodušení vývoje softwaru. I když tento přístup neodpovídá nejmodernějším trendům plně autonomní navigace, zůstává v praxi široce využíván díky své jednoduchosti, spolehlivosti a nižším nákladům.

### 5.3.1 Motivace a výchozí požadavky

Na základě požadavků byl navržen způsob, kterým lze řešit rozhodování na křižovatkách a v situacích, kdy je více možností volby (viz obrázek 5.17). Navigační systém byl založený na tradičním principu sledování čáry, který je v průmyslových provozech stále běžný díky své jednoduchosti a předvídatelnosti. Systém využívá senzor OGS pro detekci vyznačené trasy a senzor DCR pro čtení QR kódů umístěných na dráze, které obsahují informace o trase, například změny směru na křižovatkách nebo úpravy rychlosti. QR kódy mohou sloužit i ke komunikaci s nadřazeným systémem, který vozítku zadává povely na základě načtených informací. Shodného výsledku je možné docílit pomocí RFID tagů a adekvatního senzoru na jejich čtení. Tím se systém zbaví problémy se světlem, ale je o něco držší. Cílem vývoje bylo otestovat vhodnost této kombinace senzorů pro daný scénář a zohlednit rostoucí požadavky na bezpečnost.



Obrázek 5.17: Prostředí definující požadavky na AGV.

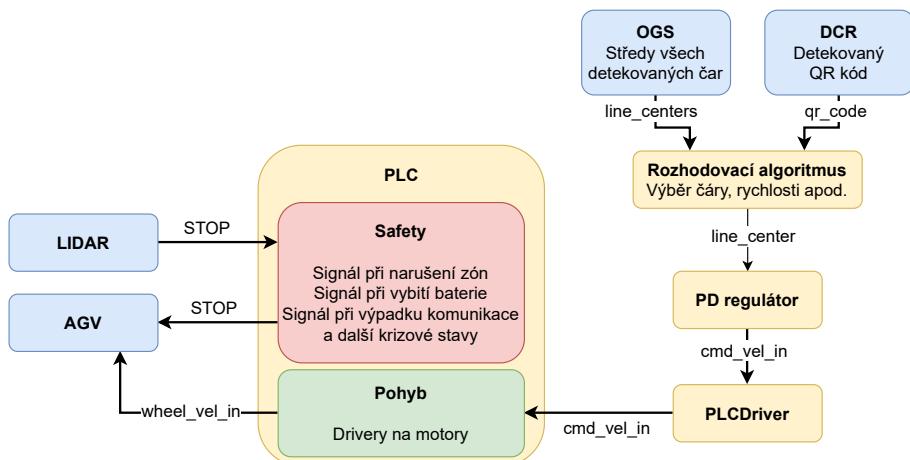
Moderní bezpečnostní normy vyžadují pokročilejší řešení než tradiční mechanické koncové spínače, které zastavují AGV až při fyzickém kontaktu s překážkou, jak to ještě u mnohých starých AGV v současných provozech funguje. Proto byl do systému integrován LiDAR, který

s vysokou frekvencí skenuje okolí a umožňuje definovat bezpečnostní zóny. Při narušení těchto zón LiDAR generuje signál, například prostřednictvím protokolu ProfiSafe, který je předán do PLC. PLC následně zastaví motory a přeruší provoz, přičemž obnovení je možné až po uplynutí stanovené doby od pominutí nebezpečí (viz sekce 5.1). Tento přístup zůstává ekonomicky výhodný pro aplikace, kde AGV sleduje pevně definované trasy podle jízdního řádu, a zároveň umožňuje rychlejší vývoj softwaru a nižší hardwarové náklady, například použitím jednoho LiDARu s omezeným zorným polem 180° v případech, kdy AGV necouvá.

### 5.3.2 Technická implementace a konfigurace

Pro testování byly stanoveny požadavky na maximální rychlosť vozítka s ohledem na jeho hmotnost a náklad:  $0.7 \text{ m s}^{-1}$  na rovných úsecích a  $0.3 \text{ m s}^{-1}$  v zatáčkách. Senzor OGS byl na konfigurován pro sledování čáry na podlaze, zatímco senzor DCR zajišťoval čtení QR kódů s informacemi o trase. Testy probíhaly v reálných podmínkách tovární hal, kde podklad vykazoval nerovnoměrné zbarvení, místy nízký kontrast čáry a proměnlivé osvětlení (část trasy byla osvětlena okny nebo zářivkami, část byla ve stínu). Po optimalizaci nastavení senzoru OGS, zejména jeho filtrů a citlivosti na kontrast, bylo dosaženo spolehlivého sledování čáry i v těchto náročných podmínkách.

Poloha čáry detekovaná senzorem OGS slouží jako vstup pro regulaci pohybu pomocí navrženého PD regulátoru, který upravuje úhlovou rychlosť vozítka, zatímco lineární rychlosť zůstává fixně nastavena podle načtených informací z QR kódů. Té se však musí regulátor dynamicky přizpůsobovat. Vozítko dokázalo při rychlosći  $0.7 \text{ m s}^{-1}$  přesně detektovat a interpretovat QR kódy, a to i v situacích, kdy byly umístěny tři kódy v těsné blízkosti za sebou. Na základě načtených informací byla dynamicky upravována rychlosť (např. zpomalení před zatáčkou a zrychlení po jejím projetí) a aktivována barevná LED signalizace indikující aktuální stav – například červená LED při zpomalení a modrá při návratu na maximální rychlosť (viz Video 11 [8]). Komunikace mezi jednotlivými komponentami systému je znázorněna na obrázku 5.18.



Obrázek 5.18: Schéma propojení komponent při navigaci po čáře.

Pro integraci senzorů do ROS2 byl vyvinut driver, který zpracovává data ze senzorů OGS a DCR přes sériovou linku. Senzor OGS poskytuje široké spektrum informací, včetně aktuálního kontrastu, nastavení filtrů a počtu detekovaných čar, avšak pro regulaci byla využita pouze poloha středu čáry. Aby byla zachována reakční frekvence senzoru 10 ms, byla data filtrovaná a

[8] ↑ <https://youtu.be/Ent6ZXx1k0U>

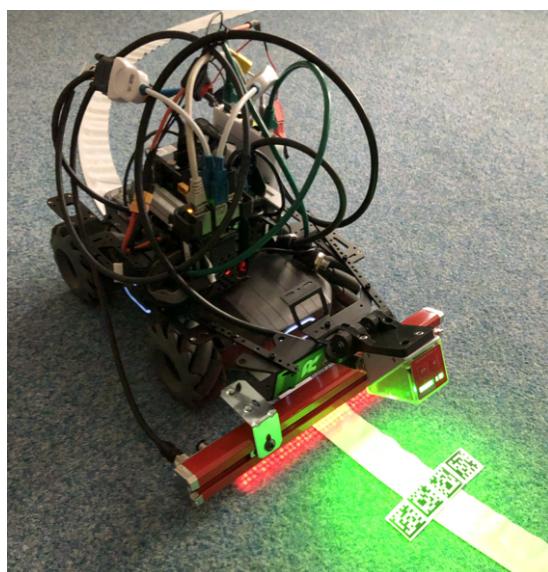
přeposílána pouze nezbytná pole. Senzor DCR byl nakonfigurován tak, aby přes sériovou linku odesílal pouze načtené označení QR kódu, což zjednodušilo jeho zpracování. Driver publikuje tato data jako zprávy v ROS2 – polohu středu čáry a načtený QR kód – na jejichž základě jsou generovány povely pro regulaci pohybu a ovládání LED signalizace. Tento mechanismus zajišťuje efektivní koordinaci mezi senzory a řídicím systémem vozítka v reálném čase.

### 5.3.3 Aplikace na různých platformách

Tato sekce popisuje implementaci a testování navaigacního systému založeného na jízdě po čáře na dvou různých platformách – podvozku DJI a diferenciálním podvozku. Na podvozku DJI byly provedeny počáteční testy zaměřené na ověření funkčnosti senzorů OGS a DCR, jejich integrace do ROS2 a regulace pohybu, čímž byl položen základ pro další vývoj. Diferenciální podvozek pak sloužil jako hlavní testovací platforma pro komplexní scénáře v reálných sklado-vých a výrobních prostředích, kde byly optimalizovány rozhodovací algoritmy a bezpečnostní funkce. Obě platformy demonstруjí možnosti navrženého systému a jeho přizpůsobivost různým podmínkám, přičemž výsledky naznačují potenciál pro další rozvoj a přechod k pokročilejším navaigacním přístupům.

#### 5.3.3.1 Podvozek DJI

Podvozek DJI sloužil jako počáteční testovací platforma pro ověření funkčnosti technologie jízdy po čáře v reálných podmínkách (viz obrázek 5.19). Hlavním cílem bylo posoudit vhodnost senzorů OGS a DCR pro detekci čáry a QR kódů a zajistit jejich integraci do prostředí ROS2. Pro tento účel byly vyvinuty ovladače (drivery), které umožňují přenos dat přes sériovou linku RS232 do formátu ROS2. Zpracovaná data jsou publikována do příslušných topiců, zatímco bezpečnostní funkce jsou zajištěny samostatnými safety scannery. Senzor OGS se ukázal jako vysoce konfigurovatelný pro různé typy podkladů, čar a křížovatek, a navíc poskytuje informace o vlnových délkách detekovaného světla. Tato vlastnost umožňuje rozlišovat barvy čar, což otevírá možnost vytvořit roboty sledující specifické barvy – například jeden robot může sledovat červené čáry a druhý modré, což může být výhodné v rozlehlých provozech pro organizaci pohybu více vozítek.



Obrázek 5.19: Fotografie podvozku DJI sledujícího čáru a čtoucího QR kódy.

Pro regulaci pohybu byl navržen jednoduchý PD regulátor, který byl zvolen kvůli své vhodnosti pro systémy s kmitavým chováním způsobeným oscilacemi vozítka kolem čáry. Integrační složka nebyla zahrnuta (většinou je systému s tímto charakterem spíše na škodu), protože cílem není dosažení přesné referenční hodnoty, nýbrž udržení stability systému s optimálními dynamickými vlastnostmi. Ladění regulátoru probíhalo ve dvou fázích: nejprve byl implementován P regulátor pro základní stabilitu, což umožnilo vozítku přibližně kopírovat čáru a sbírat data. Následně byla provedena experimentální identifikace v uzavřené regulační smyčce, při níž byla naměřena data z jízdy na rovných úsecích i v zatáčkách. Tato data byla approximována modelem kmitavého systému druhého řádu pomocí nástroje PIDlab [43]. Výsledný přenos uzavřené smyčky (viz vztah 5.1) byl přepočítán na přenos samotného systému (viz vztah 5.2):

$$F_c = \frac{C \cdot P}{1 + C \cdot P} \quad (5.1)$$

$$P = \frac{F_c}{C - F_c \cdot C}, \quad (5.2)$$

což umožnilo přesné naladění PD regulátoru specificky pro daný typ AGV. Tento proces je však nutné opakovat pro každý exemplář zvlášť kvůli rozdílům v mechanických vlastnostech.

Testy zahrnovaly analýzu vlivu proměnných světelných podmínek, odrazivosti materiálu a rychlosti zpracování dat na přesnost regulace. Senzor OGS umožnil stabilní sledování čáry i v ostrých zatáčkách při maximální testované rychlosti  $1 \text{ m s}^{-1}$ , přičemž rychlosť čtení dat nebyla limitujícím faktorem díky efektivnímu zpracování. Senzor DCR byl testován pro čtení QR kódů za pohybu při rychlosti  $0.7 \text{ m s}^{-1}$ , což představovalo výzvu kvůli jeho kamerové technologii. Výkon senzoru byl ovlivněn osvětlením, stíny a rozmažáním obrazu, ale po optimalizaci iluminace byla zajištěna spolehlivá detekce až tří po sobě jdoucích kódů. Testy potvrdily funkčnost konfigurace pro základní scénáře jízdy po čáre, avšak komplexnější prvky, jako jsou křížovatky, nebyly zkoumány. Videoukázky demonstруjí detekci QR kódu senzorem DCR s vizuální signalizací přes změnu barvy LED světel (viz Video 12 [9]), kde je vidět zpomalený záběr přečtení kódu a reakce na něj) a stabilní sledování čáry senzorem OGS (viz Video 13 [10]). Bezpečnostní zastavování nebylo v této fázi řešeno, protože primárním cílem bylo posouzení technické pravděpodobnosti.

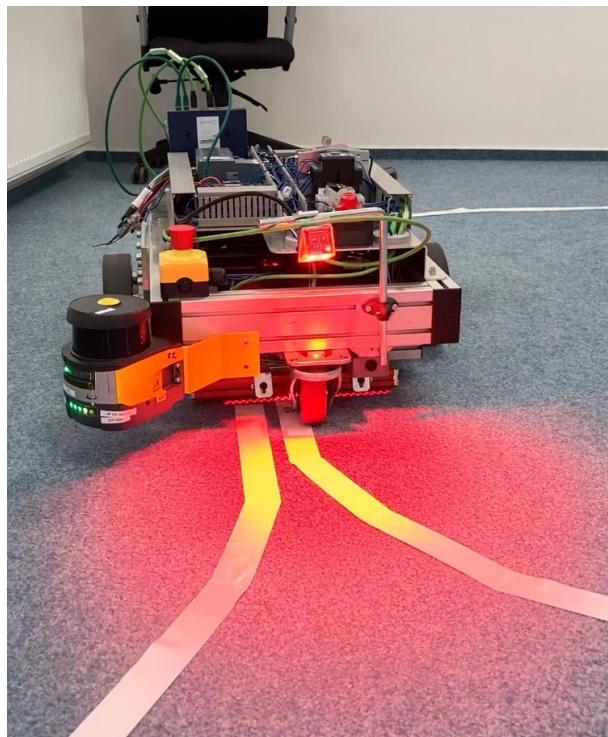
### 5.3.3.2 Diferenciální podvozek

Diferenciální podvozek představoval hlavní testovací platformu pro aplikaci jízdy po čáre v reálných skladowých a výrobních prostředích. Tento prototyp byl navržen pro komplexní scénáře zahrnující dlouhé úseky tratí s výhybkami, křížovatkami, křížením drah a zatáčkami, kde rozhodování o trase probíhalo na základě informací z QR kódů přečtených senzorem DCR (viz obrázek 5.20). Po úspěšném otestování základních funkcí na podvozku DJI byl na této platformě optimalizován rozhodovací algoritmus pro odbočování, zastavování na stanovištích a přizpůsobení rychlosti podle načtených kódů. Současně byl laděn PD regulátor, aby zajistil stabilitu při různých rychlostech, s důrazem na minimální poloměr zatáček, plynulé přepínání rychlostních profilů a robustnost při křížení nebo přerušení čáry. Senzor OGS se prokázal jako spolehlivý i při detekci sedřené čáry díky své odolnosti vůči šumu a schopnosti rozlišovat čáru při minimální viditelnosti.

---

[9] ↑ <https://youtu.be/e3kIIHGrRsQ>

[10] ↑ <https://youtu.be/Ent6ZXx1k0U>



Obrázek 5.20: Fotografie finálního prototypu pro testy jízdy po čáře.

Integrace PLC a bezpečnostních LiDARů zajistila soulad s požadovanou úrovní bezpečnosti (Safety Integrity Level, SIL), přičemž bezpečnostní zóny byly nakonfigurovány tak, aby umožňovaly plynulý provoz a zároveň garantovaly okamžité zastavení při narušení. Testování v reálných podmínkách prokázalo funkčnost systému. Videoukázky demonstrují odbočování na základě QR kódů, sjíždění na spojující se trasy, regulaci v prudkých zatáčkách a reakci na narušení bezpečnostních zón (viz [Video 14](#) [11]) včetně ignorování křížících se drah (viz [Video 15](#) [12]). Druhá ukázka zachycuje jízdu po trase s prudkými odbočkami, kde byla rychlosť omezena kvůli budoucímu využití pro tahání vlečných vozíků, přičemž požadavek na hladkosť pohybu byl splněn díky optimalizovanému regulátoru a přesné detekci senzoru OGS.

### 5.3.3.3 Shrnutí

Testování ukázalo, že kombinace senzorů OGS a DCR s doplňkovým LiDARem splňuje požadavky na rychlosť, přesnosť a bezpečnosť. AGV dosahovalo stabilního výkonu při rychlosći  $0.7 \text{ m s}^{-1}$  na rovných úsecích a  $0.3 \text{ m s}^{-1}$  v zatáčkách, přičemž QR kódy byly spolehlivě detektovány i v náročných světelných podmínkách. Implementovaný driver v ROS2 zajistil efektivní zpracování dat a jejich převod na řídicí příkazy pro dynamickou regulaci pohybu a signalizaci. Tento přístup umožňuje splnit přísné bezpečnostní normy bez nutnosti měnit základní navigační logiku založenou na čáře. Hlavní výhodou je jednoduchost a nižší náklady ve srovnání s plně autonomními systémy, díky méně komplexnímu vývoji softwaru a možnosti použít LiDAR s omezeným zorným polem. Na druhé straně systém zůstává omezený svou závislostí na fyzické čáře a postrádá flexibilitu moderních navigačních přístupů. Zároveň však instalace LiDARu, primárně pro bezpečnost, otevírá potenciál pro budoucí rozšíření o pokročilejší funkce, což inspirovalo vývoj virtuálních drah popsaných v následujících částech.

[11]↑ <https://youtu.be/l445HYQfIKg>

[12]↑ [https://youtu.be/8kM1mF827\\_4](https://youtu.be/8kM1mF827_4)

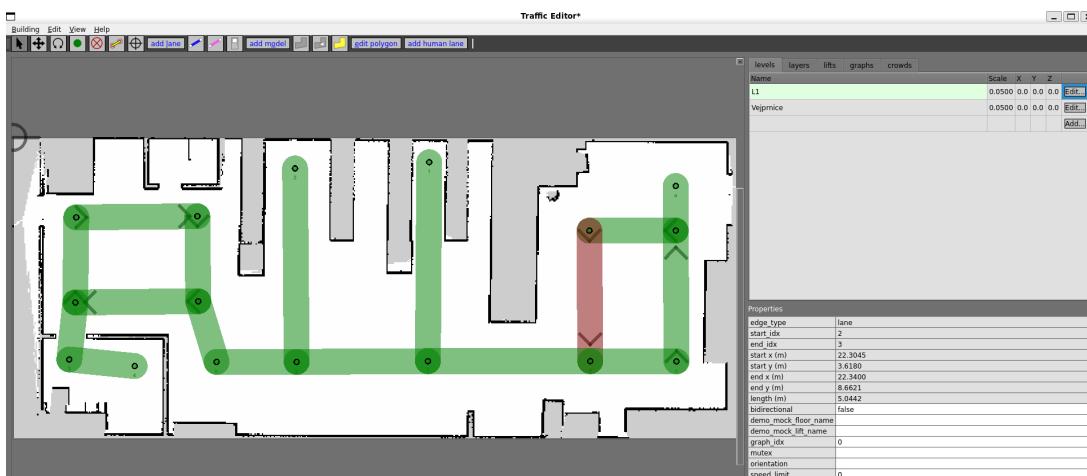
## 5.4 Automatická navigace

Termínem automatická navigace je v této práci označeno chování robota, které není autonomní (nepřizpůsobuje se dynamickým změnám prostředí, neplánuje si sám trasu atd.), ale zároveň využívá určité prvky jako je mapování, lokalizace v mapě apod. Tyto komponenty tvoří dohromady základ, který zajišťuje přenos informace o poloze robota v prostředí. S tím je možné dále pracovat velmi rozmanitým způsobem. Tato kapitola se věnuje popisu aplikací, které byly na tomto základu postaveny, a dalším možnostem využití připravených funkčních celků.

### 5.4.1 Jízda po virtuální čáře

Plně autonomní pohyb robota v prostoru je sice atraktivní vlastností, avšak v některých scénářích není taková míra samostatnosti žádoucí ani nutná. V současných průmyslových provozech lze často vidět vozítka, která se pohybují po vyznačených čarách na podlaze a na křížovatkách se rozhodují podle předem definovaných pravidel. Tato vozítka operují v uzavřeném systému tras, což na první pohled představuje cenově dostupné řešení – postačí senzor pro sledování čáry a případně čtečka QR kódů pro orientaci. Nicméně tento předpoklad nemusí být zcela správný. Jak bylo řečeno v předchozí kapitole 5.3, moderní požadavky na bezpečnost vyžadují robustní *safety* řešení, která přesahují možnosti jednoduchých senzorů. Aby systém splňoval bezpečnostní standardy, je třeba zajistit komunikaci přes *safety* protokoly, použít *safety* PLC moduly a vyvinout sofistikovaný *safety* program, jenž překračuje základní příkazy typu „jed“ nebo „stůj“. K tomu jsou zapotřebí pokročilé bezpečnostní senzory, typicky používané jsou LIDARy. Tyto senzory s krátkou periodou detekují překážky, umožňují nastavení různých bezpečnostních zón a jejich dynamické přepínání, přičemž přímo generují varování. Tato informace může být dále zpracována například v PLC.

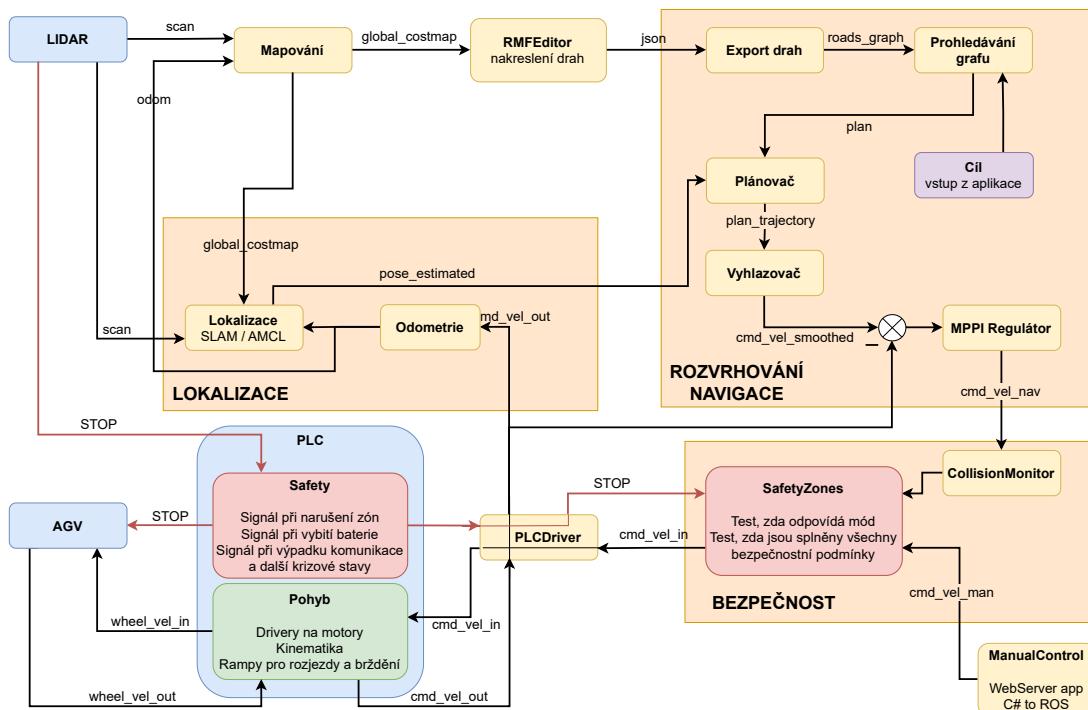
V praxi tak mnohá vozítka disponují LIDAREm z bezpečnostních důvodů, aniž by jej využívala k navigaci. Tato nevyužitá kapacita inspirovala myšlenku integrovat LIDAR i do navigačního procesu, konkrétně pro pohyb robotů po virtuálních drahách v mapě. Tento přístup přináší několik výhod: eliminuje potřebu dalších senzorů, zjednoduší konfiguraci tras (změny lze provést úpravou mapy bez fyzického přemalování čar) a umožňuje zahrnout přímo do mapy specifická pravidla pohybu.



Obrázek 5.21: Ukázka kreslení drah do mapy pomocí *rmf-traffic-editor*.

Základním principem navigace v tomto přístupu je definice sítě virtuálních cest a pojmenovaných bodů v mapě prostředí. K tomuto účelu je využit nástroj *rmf-traffic-editor* (viz obrázek

5.21), který umožňuje zakreslit orientované dráhy přímo do mapy a vygenerovat orientovaný graf ve formátu JSON. Tento graf je následně načten a převeden na sadu drah, po kterých se robot pohybuje. Robot sleduje pouze tyto předem definované dráhy a přesouvá se mezi body na základě zadaných pokynů, přičemž nejkratší cesta mezi dvěma body je určena prohledáváním grafu (viz sekce 3.7). Naplánovaná trasa je předána plánovači, který generuje konkrétní plán pohybu ve formě vektorů rychlostí pro řízení robota. Plánovač zohledňuje aktuální polohu robota určenou lokalizací. Pokud se robot nachází příliš daleko od naplánované trasy, je schopen se k ní přiblížit a přitom se vyhnout překážkám. Jakmile se dostane na virtuální dráhu, přísně ji sleduje a nevybočuje z ní. V případě detekce překážky na trase se robot zastaví a čeká, dokud překážka nezmizí, což zajišťuje bezpečnost a předvídatelnost jeho pohybu. Každá trasa je před předáním regulátoru zpracována vyhlazovačem, který zajišťuje plynulé změny směru a rychlosti. Tento přístup využívá kompletní autonomní navigaci s tím, že trajektorie jsou pevně dány předem, což eliminuje problémy spojené s dynamickým plánováním, jako jsou nečekané překážky, slepé cesty nebo zacyklení v prostoru. Robot tak operuje ve vyhrazeném prostoru, kde plánovač slouží pouze ke korekci drobných odchylek mezi reálnou a požadovanou polohou způsobených například nepřesnostmi lokalizace. Tento mechanismus, znázorněný na obrázku 5.22, nahrazuje fyzické čáry, které vyžadují pravidelnou údržbu, čištění, jsou obtížně přizpůsobitelné a jejich detekce je závislá na světelných podmínkách.



Obrázek 5.22: Propojení jednotlivých komponent pro jízdu po virtuálních silnicích.

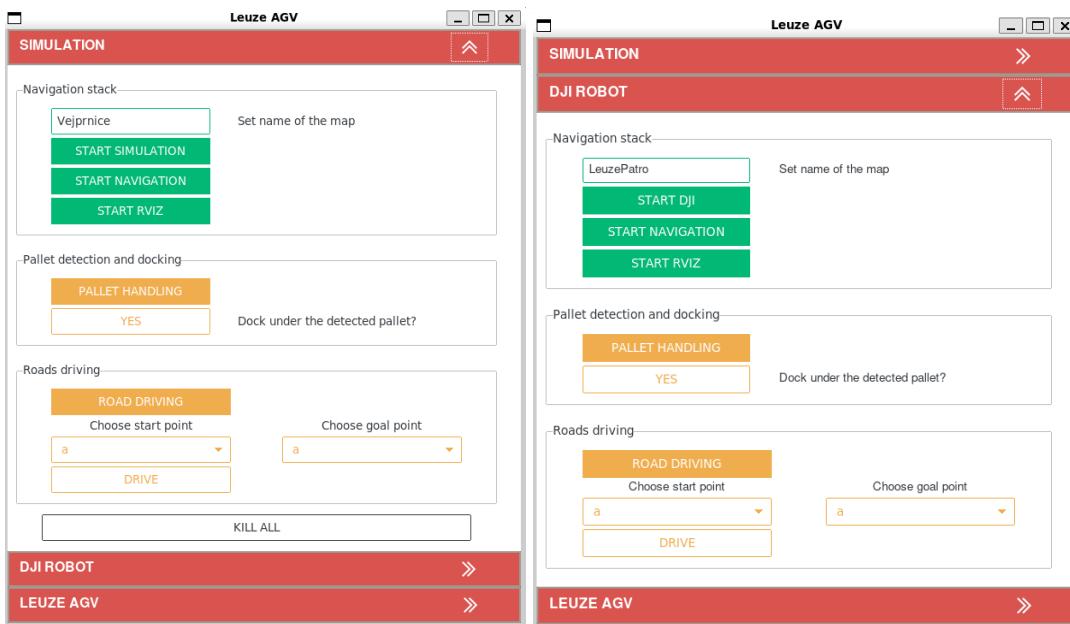
#### 5.4.1.1 Jednoduché grafické uživatelské rozhraní (GUI)

Pro praktické využití tohoto navigačního systému je nezbytné poskytnout nástroj, který umožňuje snadné zadávání pokynů pro přesun robota mezi pojmenovanými body v mapě. Z tohoto důvodu bylo navrženo jednoduché grafické uživatelské rozhraní (GUI), které usnadňuje interakci uživatele s robotem. Rozhraní integruje spouštění příkazů v linuxových terminálech prostřednictvím nástroje xTerm a vychází ze standardního způsobu spouštění uzlů v ROS2, jak ilustruje následující příklad:

```
ros2 run package_name node_name ros2_params
ros2 launch package_name launch_file_name.py ros2_params
```

Pro komplexní úlohy se v ROS2 často využívají *launch* soubory, které umožňují současné spuštění více uzlů a nastavení jejich parametrů, včetně přemapování topiců. Tyto soubory jsou vhodné pro úkoly zahrnující vzájemně provázané dílčí procesy, například lokalizaci, mapování, plánování a sledování trajektorie. Z důvodu přehlednosti a efektivity však není praktické zařnovat všechny funkce do jednoho *launch* souboru. GUI proto rozlišuje jednotlivé operace, přičemž každé tlačítko spouští specifický *launch* soubor, který provede příslušnou sadu operací.

Výsledné rozhraní, znázorněné na obrázku 5.23, umožňuje uživateli rychle vybrat typ robota nebo simulace, připojit se k zařízení, spustit navigaci, mapování, načíst předem vytvořenou mapu, aktivovat virtuální silnice z uloženého souboru nebo zadat přesun k pojmenovanému bodu. Uživatel tak může jednoduše určit počáteční a cílový bod, přičemž systém automaticky vypočítá a sleduje nejkratší cestu podle definované sítě virtuálních drah. Toto GUI je vytvořeno jako dočasné řešení pro testovací a vývojové účely, aby usnadnilo spuštění a ovládání všech funkcionalit. Přesto se stalo základem pro vývoj komplexní webové aplikace, která splňuje požadavky pro průmyslové nasazení. Vývoj této aplikace zajišťuje kolega Jan Holub [27], a její stručný popis je uveden v závěru práce.



Obrázek 5.23: Ukázka jednoduchého ovládacího GUI.

#### 5.4.1.2 Shrnutí

Tento přístup představuje zjednodušenou alternativu, která kombinuje výhody plně autonomní navigace s principy tradičního sledování čáry. Využitím LiDARu (původně instalovaného pro bezpečnostní účely) k navigaci po virtuálních drahách dochází k úspoře zdrojů a zvýšení modifikovatelnosti prostředí bez nutnosti manuálního překreslování fyzických čar. Prohledávání grafů zajišťuje efektivní plánování cest, zatímco jednoduché GUI usnadňuje ovládání a zadávání pokynů. Příprava vyhrazených virtuálních silnic a pojmenovaných bodů, na které lze robota jednoduše poslat, může vzdáleně připomínat základy systémů pro správu flotily (*fleet management*). Stručnému představení této koncepce se věnuje následující kapitola.

## 5.4.2 Fleet management

Tato sekce se zaměřuje na koncept správy flotily robotů (*fleet management*) a jeho integraci s autonomní navigací v rámci navaigаčního stacku Nav2. Cílem je objasnit podstatu správy flotily, její výhody při využití existujících navaigаčních technologií a přínosy pro průmyslové a komerční aplikace. Ačkoliv se tato práce zabývá pouze vývojem jednoho robota s danými funkcemi, koncept správy flotily představuje důležitý základ pro budoucí rozvoj kooperace více robotů, což je nezbytné pro reálné průmyslové nasazení.

### 5.4.2.1 Definice a podstata správy flotily

Správa flotily robotů představuje systém pro koordinaci a řízení více autonomních robotů působících v rámci jednoho prostředí nebo organizace. Tento systém zahrnuje plánování úkolů, alokaci zdrojů, monitorování stavu robotů – například jejich polohy, stavu baterie nebo zatížení – a optimalizaci jejich činnosti s cílem zajistit efektivitu celého systému. V kontextu této práce správa flotily rozšiřuje existující navaigаční schopnosti jednotlivých robotů o centralizované nebo distribuované řízení na úrovni celé skupiny.

Základním předpokladem je existence autonomní navigace, která umožňuje robotům samostatně se pohybovat, vyhýbat se překážkám a plnit zadáne úkoly. Správa flotily tyto individuální schopnosti povyšuje na kolektivní úroveň, kde koordinace mezi roboty zvyšuje celkovou efektivitu systému. Využití vyhrazených virtuálních drah, jak je popsáno v předchozí kapitole, je v tomto případě vhodné, protože jasně definuje pravidla pohybu pro všechny roboty. Například na každé křižovatce (uzlu) se robot hlásí nadřazenému systému, který tak sleduje obsazenost jednotlivých drah a zajišťuje plynulý provoz.

Přístup založený na ROS2 a navaigаčním stacku Nav2 umožňuje zaměřit se na optimalizaci koordinace, aniž by bylo nutné vyvíjet navaigаční stack od základu. Nav2 poskytuje robustní nástroje, jako jsou akční servery a stromy chování, které usnadňují implementaci správy flotily. Dále systém podporuje sdílenou mapu, kdy mapování může probíhat současně na více robotech a výsledná mapa je skládána jako jeden celek. Pokud jeden robot detekuje nezmapovanou překážku, tato informace je sdílena s ostatními roboty, kteří ji zohledňují při plánování svých tras.

### 5.4.2.2 Výhody správy flotily nad autonomní navigací

Integrace správy flotily s autonomní navigací přináší několik výhod, které zvyšují efektivitu a použitelnost robotických systémů v průmyslových prostředích. Správa flotily umožňuje dynamicky přidělovat úkoly na základě dostupnosti robotů, jejich polohy, stavu baterie nebo aktuálního zatížení. Například robot s nízkým stavem baterie je odeslán k nabíjení, zatímco jiný robot přebírá jeho úkol. Koordinace více robotů zkracuje čas potřebný k dokončení komplexních úkolů, jako je mapování prostoru, doprava materiálu nebo bezpečnostní hlídka, díky paralelnímu plnění těchto činností. SLAM Toolbox navíc podporuje mapování prostřednictvím více autonomních jednotek, což dále zefektivňuje proces. Systém je snadno rozšířitelný o další roboty bez nutnosti zásadních změn v navaigаční infrastruktuře, což je ideální pro rostoucí průmyslové aplikace. Centralizované nebo distribuované řízení reaguje na selhání jednotlivých robotů přerozdělením úkolů, čímž minimalizuje dopad poruch na celkový výkon. Vyšší míra abstrakce umožňuje definovat jednotky práce (například vyzvednutí zboží ve skladu) a předávat je robotům bez složité navaigаční logiky.

Správa flotily robotů nad autonomní navigací představuje logický krok k rozšíření schopností jednotlivých robotů na kolektivní úroveň. Využitím robustního základu Nav2, včetně jeho

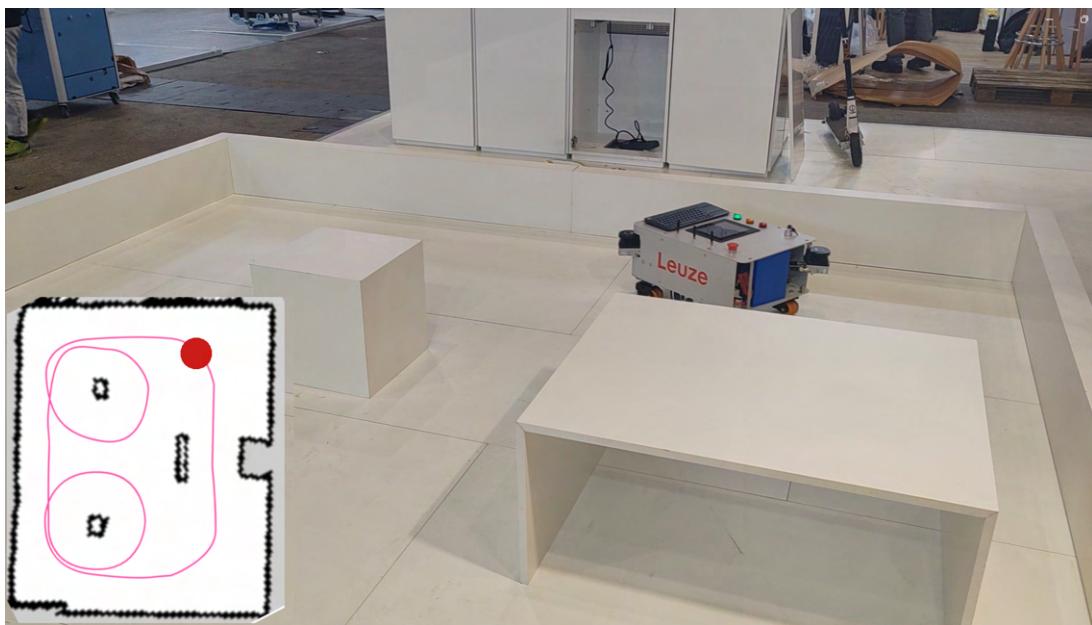
nástrojů jako akčních serverů, stromů chování a sledování bodů trasy, lze vytvořit efektivní systém pro koordinaci více robotů. Tento přístup optimalizuje využití zdrojů, zvyšuje produktivitu a otevřá cestu ke škálovatelným a odolným řešením pro moderní robotické aplikace.

### 5.4.3 Virtuální dráhy

Tato sekce se zaměřuje na implementaci cyklických virtuálních drah. Místo robustního MPPI regulátoru je využit algoritmus Pure Pursuit. Principy tohoto algoritmu jsou podrobně popsány v kapitole 4.4. Pure Pursuit je založen na geometrickém výpočtu řídicích zásahů, kdy robot sleduje plovoucí bod na trajektorii, označovaný jako *look-ahead point*, ve vzdálenosti předem definované od robota. Tento přístup zajišťuje plynulé a robustní řízení po zadané dráze a je zvláště vhodný pro scénáře s předem známou a neměnnou trajektorií, což odpovídá navrženému konceptu virtuálních drah. Níže je popsána implementace virtuálních drah, jejich praktické využití a přínosy v kontextu testování lokalizace, senzorů a průmyslových aplikací.

#### 5.4.3.1 Základní princip a využití

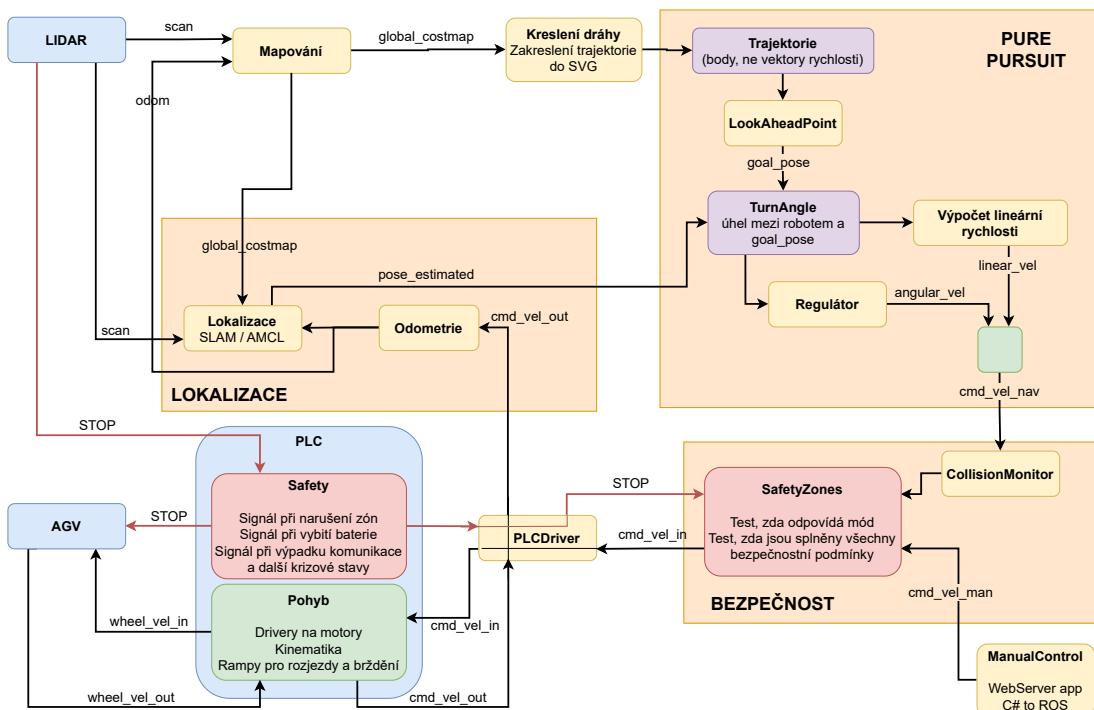
Implementace virtuálních drah vychází z následujícího principu. Nejprve je vytvořena mapa prostředí, například pomocí SLAM Toolbox nebo ručním zakreslením, která slouží jako základ pro definici trajektorie. Následně je aktivována lokalizace robota v mapě, využívající kombinaci LiDARových dat, enkodérů nebo vizuální odometrie prostřednictvím algoritmu AMCL. Po zadání fixní trajektorie, definované jako posloupnost bodů v prostoru, robot automaticky sleduje tuto dráhu v cyklickém režimu za použití algoritmu Pure Pursuit. Pure Pursuit zajišťuje plynulé přizpůsobení směru a rychlosti pohybu vzhledem k aktuální poloze a orientaci robota. Tento princip je znázorněn na obrázku 5.24, který ukazuje prostředí pohybu vozíku spolu s odpovídající mapou a nakreslenou trajektorií.



Obrázek 5.24: Znázornění principu a funkce virtuálních drah.

Virtuální dráhy představují efektivní nástroj pro testování lokalizačních algoritmů. Cyklický pohyb robota po stejné trase umožňuje dlouhodobé monitorování přesnosti lokalizace v reálném čase. Lze tak sledovat potenciální drift polohy, výpadky LiDARových skenů nebo anomálie, jako je nesprávná registrace bodových mračen způsobená šumem měření. Opakovaný pohyb navíc usnadňuje sběr velkého množství dat bez nutnosti manuálního zásahu, což zjednoduší analýzu robustnosti lokalizačních metod za různých podmínek, například při změně osvětlení nebo v přítomnosti dynamických překážek.

Fungování systému je založeno na přesné lokalizaci robota v mapě. Trajektorie je definována jako uzavřená křivka nakreslená do mapy a exportována s ohledem na její vlastnosti, jako je rozlišení a reálná velikost. Tato trajektorie je poté předána řídicímu systému. Algoritmus Pure Pursuit vyhledá nejbližší bod na trajektorii ve vzdálenosti *look-ahead* od robota, přičemž zohledňuje index bodu vzhledem k předchozímu indexu, aby se zabránilo přeskakování při křížení trajektorie. Následně je vypočítán úhel mezi tečnou k trajektorii v daném bodě a aktuálním natočením robota. Tento úhel slouží k regulaci úhlové rychlosti, tedy rychlosti otáčení robota kolem vlastní osy. Současně robot udržuje nenulovou lineární rychlosť, takže jeho pohyb opisuje oblouk. Aby robot zvládl prudké zatáčky, lineární rychlosť je dynamicky upravována na základě velikosti požadovaného úhlu – při větším úhlu se rychlosť snižuje, při menším zvyšuje. Regulace je zajištěna jednoduchým proporcionálním regulátorem, přičemž plynulosť pohybu je zajištěna díky regulaci na bod před robotem. Tento mechanismus je podrobněji znázorněn na obrázku 5.25.



Obrázek 5.25: Propojení jednotlivých komponent pro jízdu po virtuální dráze.

Tento přístup našel uplatnění při testování senzorů a jejich vhodnosti pro aplikace AGV. Cyklický pohyb umožňuje ověřovat funkčnost bezpečnostních skenerů, například přepínání mezi varovnými a bezpečnostními zónami při detekci blízkých překážek. Takto připravený scénář simuluje reálné provozní podmínky, kde AGV musí reagovat na neocenívané změny v prostředí, a zároveň poskytuje kontrolované prostředí pro sběr dat o výkonu senzorů. Lze tak testovat citlivost laserových skenerů na různé povrchy, jejich reakční dobu nebo odolnost vůči rušení, což je cenné při výběru vhodného senzorického vybavení. I když jsou vlastnosti senzorů obvykle testovány při jejich uvedení na trh, tento přístup umožňuje analyzovat dopad jejich nedokonalostí na přesnost lokalizace, která je zásadní pro správné manévrování. Dále lze zkoumat vliv vzájemného ozařování skenerů při pohybu více robotů v jednom prostředí.

Virtuální dráhy přinášejí výhody i z hlediska průmyslového nasazení a demonstračních účelů. V průmyslovém prostředí slouží jako základ pro automatizaci opakujících se přepravních úloh s pevně danou trajektorií, například při dopravě materiálu mezi stanovišti ve výrobní

hale. Opakovatelnost a přesnost pohybu zajištěná algoritmem Pure Pursuit umožňuje testování dlouhodobé spolehlivosti AGV bez potřeby složitých adaptivních algoritmů, jako jsou MPC nebo MPPI, které jsou vhodnější pro dynamická prostředí. V rámci veletrhů a předváděcích akcí virtuální dráhy umožňují atraktivní prezentaci, kdy robot automaticky projíždí předem definovanou trasu, čímž demonstruje své schopnosti, spolehlivost a vlastnosti senzorů před potenciálními zákazníky či partnery. Tento přístup je nenáročný na přípravu a vizuálně působivý, což zvyšuje jeho praktickou hodnotu.

#### 5.4.3.2 Shrnutí

Implementace virtuálních drah byla rozšířena o uživatelsky přívětivou funkcionalitu, kterou navrhl kolega Holub. Tato funkcionalita umožňuje definovat trajektorii přímo v grafickém rozhraní zakreslením dráhy do SVG obrázku mapy. Uživatel nakreslí požadovanou cestu, která je převedena na posloupnost bodů a odeslána do řídicí jednotky robota prostřednictvím webové aplikace. Tento přístup zjednodušuje zadávání trajektorie, eliminuje potřebu manuálního programování a umožňuje rychlé přizpůsobení dráhy podle aktuálních potřeb. Po odeslání robot interahuje s řídicí aplikací a spouští či zastavuje sledování trajektorie pomocí algoritmu Pure Pursuit.

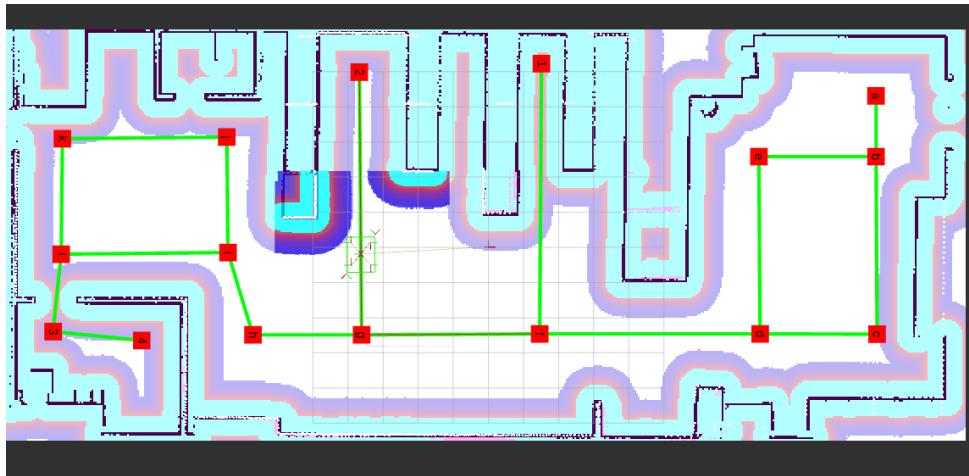
Virtuální dráhy přinášejí výhody v testovacím, demonstračním i průmyslovém prostředí. Možnost rychlého definování a opakování trajektorií usnadňuje validaci senzorických systémů před jejich nasazením do reálného provozu, což zkracuje vývojový cyklus a snižuje náklady na testování. V logistických centrech lze virtuální dráhy efektivně využít k navigaci v úzkých prostorech, zatímco v automobilovém průmyslu nacházejí uplatnění při dopravě komponentů prostřednictvím opakovaných rutin. Představují jednoduchý, ale efektivní nástroj pro testování lokalizace a senzorů v kontrolovaném prostředí. Jejich hodnota spočívá v kombinaci robustnosti a nízkých nároků na implementaci, díky čemuž jsou vhodné jak pro akademické experimenty, tak pro průmyslové a prezentační účely. Díky přístupu kolegy Holuba je systém navíc přístupný i uživatelům bez hlubších technických znalostí, což rozšiřuje jeho potenciální využití.

## 5.4.4 Aplikace na různých platformách

Tato kapitola se zaměřuje na praktickou implementaci a testování dvou odlišných přístupů k automatické navigaci na reálných platformách – podvozku DJI a podvozku TinyAGV. První přístup, označený jako *Jízda po virtuální čáre*, byl vyvinut na podvozku DJI a kombinuje výhody autonomní navigace s pevně definovanými trasami, což umožňuje efektivní organizaci pohybu v prostředí s předem stanovenými pravidly. Druhý přístup, označený jako *Virtuální dráhy*, byl implementován na podvozku TinyAGV a skoro jako jediný nebyl testován na modelu DJI. Využívá algoritmus Pure Pursuit k cyklickému následování předem nakreslených tras, což slouží nejen pro demonstrační účely, ale i pro dlouhodobé testování lokalizace a senzorů. Obě řešení byla testována v simulačním i reálném prostředí, přičemž jejich výsledky ukazují potenciál pro průmyslové nasazení.

### 5.4.4.1 Podvozek DJI

Na podvozku DJI byl vyvinut a testován první koncept automatické navigace označený jako *Jízda po virtuální čáre*. Tento přístup vychází z požadavku omezit pohyb robota na specifické oblasti, například preferovat pravou stranu chodby při navigaci. Toho lze dosáhnout manuální úpravou nákladové mapy přidáním zón s nižšími náklady, které jsou prioritizovány oproti ostatním volným prostorům. V rámci této práce však bylo navrženo a implementováno alternativní řešení založené na definici virtuálních silnic – pevně stanovených tras, které zjednoduší plánování pohybu a zadávání cílových bodů prostřednictvím přiřazení pojmenovaných lokací. Tento koncept kombinuje výhody autonomní navigace a jízdy po čáre, díky čemuž je perspektivní pro průmyslové aplikace, jako je organizace pohybu ve skladech nebo výrobních halách. Navíc poskytuje základ pro budoucí integraci do systémů správy flotily (*fleet management*). Dosavadní vývoj byl však omezen na experimentální testování v kancelářském a simulačním prostředí, jak je znázorněno na obrázku 5.26.



Obrázek 5.26: Ukázka virtuální čáry definované na mapě pro podvozek DJI.

Implementace virtuální čáry spočívala v úpravě plánovače trajektorií, který byl přepracován z dynamického výběru optimální trasy na statické následování předem definovaných cest. Nad tímto plánovačem byla vytvořena ovládací nadstavba, která umožňuje zadání virtuálních silnic a jejich směrovosti. Systém v této fázi nezahrnuje dynamické objíždění překážek - při dlouhodobé neprůjezdnosti trasy se robot pokusí přeplánovat cestu po alternativních virtuálních silnicích, pokud jsou k dispozici. Výzvou zůstala jednosměrná orientace silnic, která znemožňuje návrat

po stejně trase k nejbližší křižovatce v případě blokace. Na druhou stranu definování směrovosti silnic může být v mnoha aplikacích žádoucí, například pro zajištění plynulého toku pohybu. Pro průmyslové nasazení by bylo nutné tyto nedostatky řešit jasnou definicí strategií přeplánování, aby systém odpovídal požadavkům dynamického prostředí.

Úprava plánovače trajektorií umožnila efektivní kombinaci autonomní navigace s navigací po virtuálních čarách, jak demonstruje [Video 16](#) [13] ze simulace. Robot může být nejprve vyslán do libovolného cíle v prostoru pomocí autonomní navigace a poté přepnuto do režimu jízdy po virtuálních čarách. Při inicializaci tohoto režimu není vyžadována počáteční poloha přímo na trase virtuální čáry. Robot autonomně vyhledá nejbližší bod definované trasy za využití standardních principů autonomní navigace, včetně dynamického vyhýbání se překážkám a optimalizace trasy. Po dosažení virtuální čáry systém přejde do automatického režimu následování předem stanovené cesty. Během tohoto režimu není z bezpečnostních důvodů možné zadat nový cíl v rámci autonomní navigace - takový požadavek je blokován, dokud automatický režim neskončí nebo není manuálně ukončen. Kombinace obou přístupů byla ověřena v simulačním prostředí, jak zachycuje zmíněné video, které demonstruje přechod mezi režimy. Druhé [Video 17](#) [14] ilustruje reálné nasazení na podvozku DJI, kde robot sledoval virtuální čáru těsně podél zdi chodby, a to i přes tendenci autonomní navigace preferovat střed volného prostoru, což potvrzuje funkčnost navrženého řešení vzhledem k původnímu požadavku.

Celý koncept prozatím zůstal v této fázi vývoje, ale má potenciál pro další rozvoj a testování na reálné průmyslové platformě. Perspektivní je také rozšíření o nadstavbu řídicí logiky pro koordinaci více robotů a v budoucnu integrace do systémů správy flotily (*fleet managementu*).

#### 5.4.4.2 TinyAGV

Podvozek TinyAGV, původně navržený pro prezentační účely (viz sekce [2.2.4](#), obrázek [5.24](#)), byl přestavěn pro implementaci druhého typu automatické navigace označeného jako *Virtuální dráhy*. Výzvou při jeho nasazení byla nepřesná odometrie způsobená absencí enkodérů a využitím pouze Hallových sond pro odhad rychlosti kol, což negativně ovlivnilo lokalizaci v mapě generované pomocí SLAM. Tento nedostatek však vedl k detailnímu prostudování a optimalizaci algoritmu AMCL (viz sekce [3.11](#)). Klíčem k úspěchu byla správná parametrizace generování částic, která zohledňovala nízkou důvěru k odometrickým datům a definovala rozptyl částic vzhledem k odhadované poloze a orientaci. Tím byl kompenzován šum v odometrii a dosažena dostatečná přesnost lokalizace, což dokazuje význam pečlivé kalibrace algoritmů v podmírkách omezené senzorické výbavy. Ukázku kompenzace chybné odometrie ukazuje v simulaci [Video 18](#) [15].

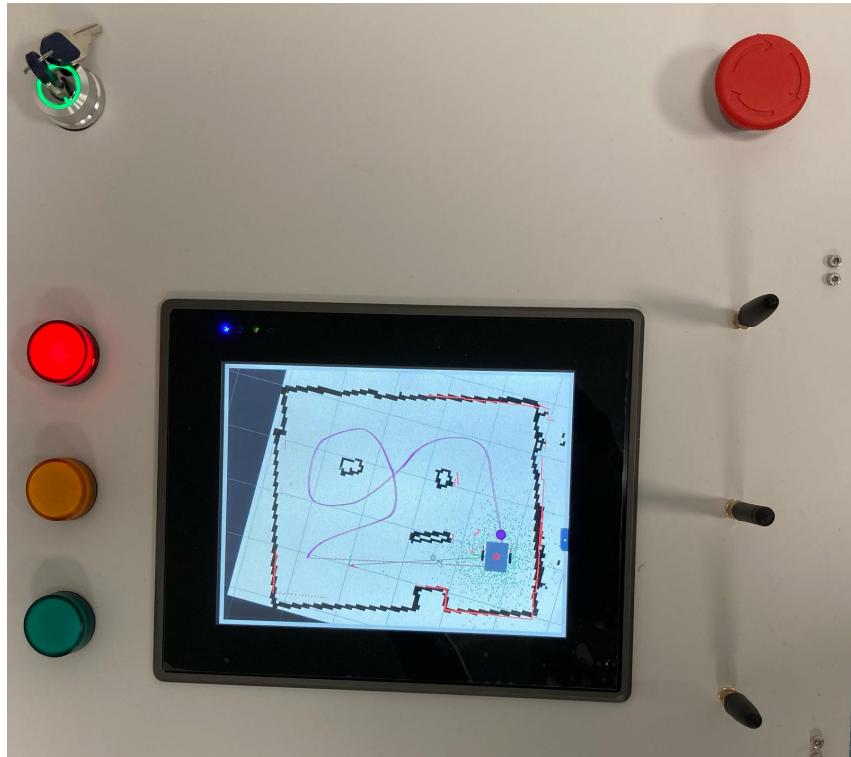
Následování virtuálních drah bylo realizováno algoritmem Pure Pursuit (viz sekce [4.4](#)), který využívá předem nakreslenou trasu v mapě prostředí. Proces zahrnuje mapování, lokalizaci, zadání trasy a její cyklické sledování (viz obrázek [5.27](#)). Tento přístup byl původně navržen pro demonstrační jízdu na veletrzích, avšak ukázal se jako efektivní nástroj pro dlouhodobé testování a kalibraci senzorů a lokalizačních algoritmů. Data z LIDARů jsou kontinuálně zpracovávána nejen pro navigaci, ale i pro detekci narušení varovných a bezpečnostních zón, což je vizualizováno prostřednictvím LED indikátorů. Potenciál této funkcionality přesahuje demonstrační účely a mohl by být rozšířen pro průmyslové aplikace, například přidáním definovaných akcí na specifických pozicích (zastavení na stanovišti, čekání na potvrzení, provedení úkonu), což by zvýšilo její využitelnost v automatizovaných provozech. Současná implemen-

[13] ↑ <https://youtu.be/gN5MGQDTLaU>

[14] ↑ <https://youtu.be/RJeJiCzLGpE>

[15] ↑ <https://youtu.be/hkSZ4L7MABU>

tace zahrnuje kompletní cyklus od mapování po regulaci, jak je demonstrováno na přiložených videoukázkách. [Video 19](#) [16] zachycuje zadání trasy přes aplikaci, druhé [Video 20](#) [17] ukazuje počáteční estimaci polohy a korekci odometrie lokalizací, třetí [Video 21](#) [18] ilustruje vizualizaci pohybu s plovoucím bodem pro Pure Pursuit, čtvrté [Video 22](#) [19] zachycuje prvotní testy, vývoj a ladění regulátoru na simulaci a páté video [Video 23](#) [20] představuje reálnou jízdu robota při testech v kanceláři i na veletrhu, čímž je finálně potvrzena funkčnost systému v kontrolovaných podmínkách.



Obrázek 5.27: Vizualizace mapy prostředí, lokalizace a následování trajektorie na displeji robota.

Po delším testování došlo ještě k úpravám regulátoru. Jedná se o vybírání lookahead bodu pro následování. Standardní Pure Pursuti (tak, jak je popsáný v sekci 4.4) vybírá bod na kružnici se středem v poloze robota. Pokud jdou ale dvě trajektorie velmi blízko u sebe, nebo jsou příliš složité, může dojít k nechtěnému přeskakování bodu na jiný úsek trajektorie. Z toho důvodu bylo implementováno ošetření v podobě vybírání bodu i podle vzdálenosti počítané po indexech bodů v trajektorii, takže robot skutečně sleduje bezchybně libovolnou trajektorii, jak ukazuje [Video 24](#) [21].

V předchozích videích je zřejmé, že největší odchylka v lokalizaci nastává při otáčení robota. Z toho důvodu bylo implementováno rozlišení rotačního a translačního pohybu robota a podle toho jsou přepínány parametry AMCL tak, aby generovaly částice ve vhodném rozptylu. Výsledek ukazuje [Video 25](#) [22].

[16] ↑ <https://youtu.be/A5xVp2POyXs>

[17] ↑ <https://youtu.be/aiPO5do6CC8>

[18] ↑ <https://youtu.be/HEZgW6SYUNU>

[19] ↑ <https://youtu.be/SkuWGRKmKqk>

[20] ↑ <https://youtu.be/Tj8zgDCbYXM>

[21] ↑ <https://youtu.be/Ek70PZHnybw>

[22] ↑ <https://youtu.be/Ek70PZHnybw>

## 5.5 Automatická detekce palety

Během vývoje navaigacního stacku pro všeobecný podvozek vybavený mecanum koly vznikl požadavek na automatizaci převážení palet položených na nožičkách, pod které robot vjíždí a následně je zdvihá. Tento standardní postup byl rozšířen o umožnění autonomní detekce palet v prostoru bez předem známé pozice. Inspirací byl systém SIMOVE od společnosti Siemens [9], který předpokládá přesnou polohu palet v mapě a nepodporuje jejich autonomní detekci. Někteří výrobci paletových systémů implementují detekci pomocí kamer a metod rozpoznávání obrazu, které jsou však citlivé na podmínky prostředí, zejména na osvětlení. V této práci byl navržen inovativní přístup založený výhradně na datech z 2D LIDARových senzorů, který nabízí ekonomickou výhodu (LIDARy jsou již integrovány), nižší výpočetní náročnost, robustnost vůči změnám prostředí a vysokou rychlosť zpracování. Kapitola popisuje postup detekce palet pomocí shlukování dat LIDARu, konkrétně algoritmu DBSCAN, implementovaného na reálném robotovi, s odkazem na potenciální rozšíření o odhad pohybu objektů pomocí Kalmanova filtru jako neimplementovaného, ale perspektivního řešení.

### 5.5.1 Shlukování pro detekci objektů z dat LIDARu

Pro identifikaci malých objektů, jako jsou nohy palet, byla zvolena metoda shlukování dat z LIDARu pomocí algoritmu DBSCAN, jehož teoretické základy byly podrobněji popsány v kapitole 4.5.2.2. Tento přístup byl preferován díky své schopnosti třídit data do shluků bez předem určeného počtu tříd, efektivně eliminovat šum a odlehlé body, a identifikovat oblasti s vysokou hustotou bodů, což je ideální pro zpracování LIDARových dat obsahujících chyby měření. V kontextu reálného robota s všeobecným podvozkem jsou data z LIDARů slučována do virtuálního skenu v počátku souřadné soustavy vozíku (viz sekce 4.2) s periodou 25 ms, což vyžaduje rychlé zpracování pro zajištění reálného času.

Data z LIDARů, která přicházejí každých 25 ms, jsou po sloučení zpracovávána DBSCAN algoritmem s parametry  $\epsilon$  (poloměr sousedství) a  $MinPts$  (minimální počet bodů pro definici shluku), které jsou nastaveny tak, aby oddělily shluky (oblasti vysoké hustoty) od šumu. Na platformě, jako je Raspberry Pi 4, může celkový čas zpracování (včetně asynchronních operací ROS2, např. *publish a subscribe*) dosáhnout až 50 ms. S ohledem na časovou složitost DBSCAN  $O(n \log n)$ , kde  $n$  je počet bodů (přibližně 1356 - 2700 pro použité LIDARy), je zpracování v reálném čase dosažitelné, protože velikost dat umožňuje rychlé procházení.

Pro každý detekovaný shluk je odhadnuta jeho poloha středu a poloměr. Střed shluku  $c$  je vypočten jako průměr souřadnic bodů:

$$\mathbf{c} = \left( \frac{1}{n_s} \sum_{i=1}^{n_s} x_i, \frac{1}{n_s} \sum_{i=1}^{n_s} y_i \right),$$

kde  $n_s$  je počet bodů v shluku a  $(x_i, y_i)$  jsou souřadnice  $i$ -tého bodu. Poloměr  $r$  je odhadnut jako maximální vzdálenost od středu k jakémukoli bodu:

$$r = \max_{i=1}^{n_s} \sqrt{(x_i - c_x)^2 + (y_i - c_y)^2}.$$

Shluky s poloměrem  $r \leq 20$  cm jsou považovány za malé objekty, jako jsou nohy palet, zatímco větší shluky (např. stěny) jsou ignorovány, aby se zabránilo nesprávné klasifikaci.

## 5.5.2 Využití pro detekci palety

Pro detekci palet jsou shluhy z DBSCAN analyzovány s cílem identifikovat čtveřici, která odpovídá nohám palety, typicky obdélníkové konstrukce s rozměry  $A \times B$  (např. standardní EURO paleta s  $A \approx 1200$  mm a  $B \approx 800$  mm). Algoritmus prochází všechny kombinace čtyř shlužek a ověřuje jejich geometrickou konzistentnost pomocí následujících podmínek, které vyplývají z obdélníkového tvaru palety:

1. Výpočet všech vzdáleností mezi čtyřmi body a jejich seřazení podle velikosti (dvě nejkratší, dvě střední, dvě nejdelší hrany).
2. Ověření, zda dvě nejkratší hrany odpovídají délce  $A \pm \delta$  (kde  $\delta$  je toleranční odchylka, např. 50 mm), dvě střední hrany odpovídají délce  $B \pm \delta$ , a dvě nejdelší hrany (úhlopříčky) odpovídají přibližně:

$$\sqrt{A^2 + B^2} \pm \delta.$$

3. Pokud jsou tyto podmínky splněny, čtveřice bodů je klasifikována jako potenciální paleta.

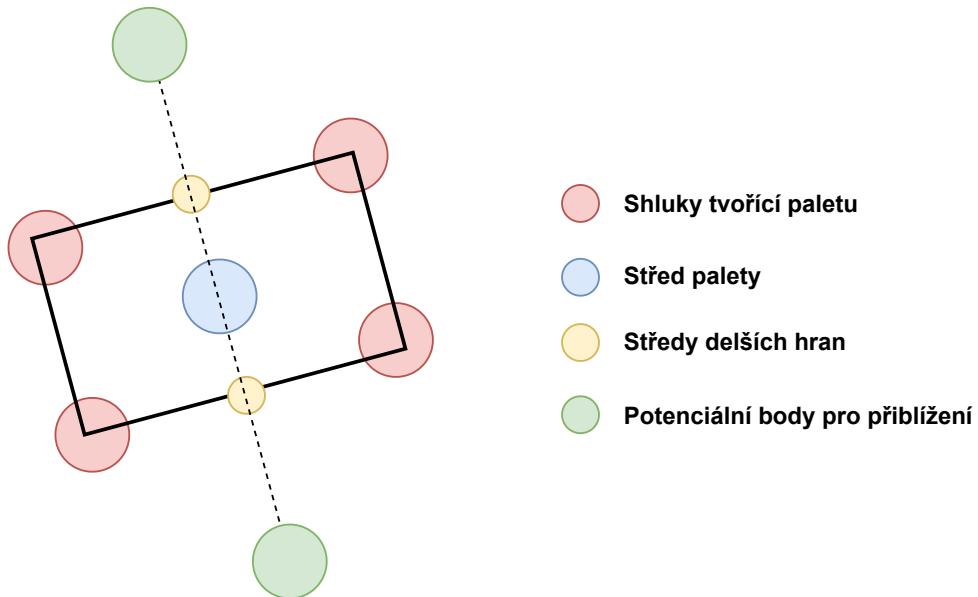
Po identifikaci palety jsou provedeny následující kroky:

- Nalezení středu palety jako průměru souřadnic čtyř nohou:

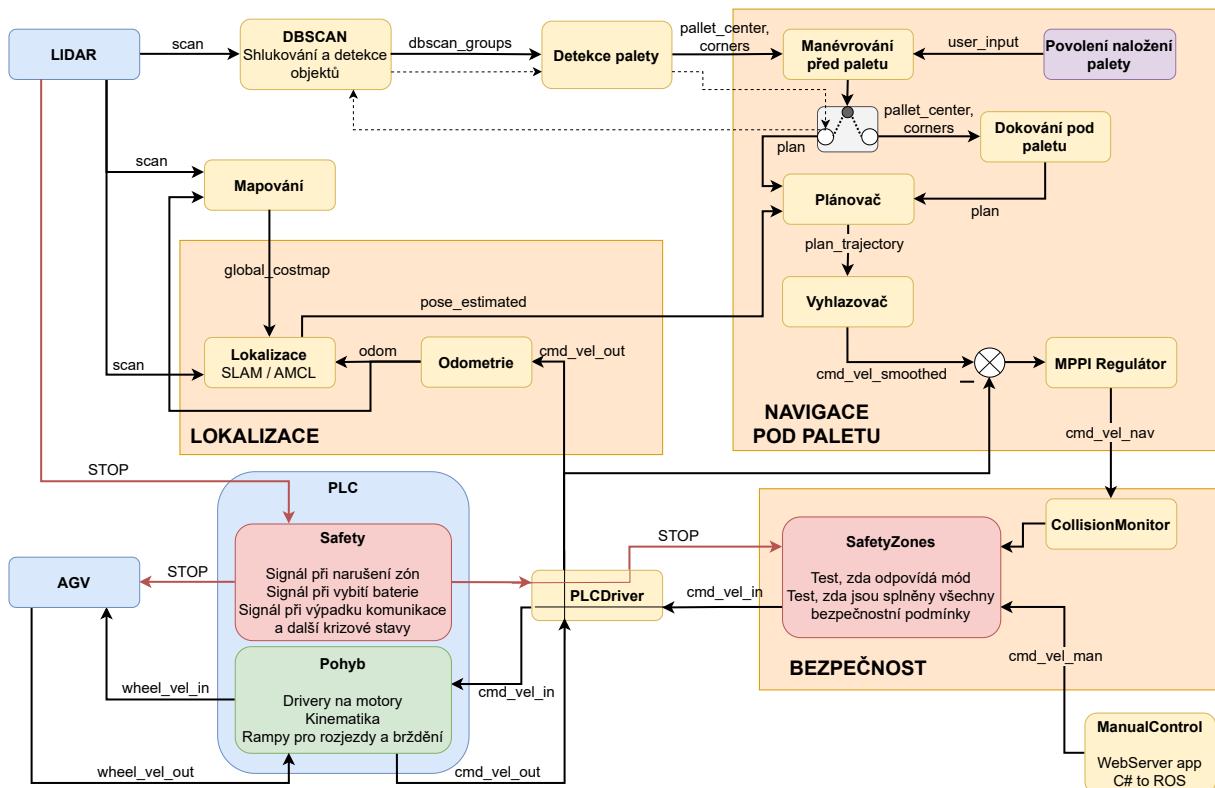
$$\mathbf{c}_{paleta} = \left( \frac{1}{4} \sum_{i=1}^4 x_i, \frac{1}{4} \sum_{i=1}^4 y_i \right).$$

- Nalezení středů delších hran obdélníku (spojujících nohy podél osy  $B$ ) a bodů vzdálených o vzdálenost  $d$  (např. polovina šířky vozíku) od těchto středů podél spojnice se středem palety. Tyto body označují potenciální vstupní pozice pro robota.
- Ověření dostupnosti těchto bodů pro robota s ohledem na překážky v mapě (viz sekce 4.5) a navigaci k blížšímu bodu pomocí navigačního stacku (viz sekce 5.2).

Obrázek 5.28 slouží jako nástroj pro vizuální interpretaci postupu automatické detekce palet, jak byl popsán v předchozích podkapitolách. Tento nákres znázorňuje hlavní prvky detekce – nohy palety, jejich geometrickou konfiguraci, vypočtený střed palety a body pro přiblížení vozíku, čímž ilustruje principy geometrické analýzy (viz sekce 4.5.2.2). Po úspěšné detekci palety a určení jejího středového bodu je tato informace předána uživateli nebo nadřazenému systému k potvrzení naložení. V případě schválení operátorem nebo automatizovaným systémem je zahájen proces výpočtu optimálních bodů pro přiblížení a navigaci k paletě včetně stanovení požadované orientace vozíku. Tento proces zahrnuje využití navigačního stacku ROS2 (viz sekce 5.2) pro plánování trajektorie s ohledem na bezpečnostní zóny (viz sekce 5.1) a překážky v mapě. Po dosažení cílové pozice je provedena finální verifikace správnosti detekce porovnáním aktuálních LiDARových dat s očekávanou geometrií palety, jak je znázorněno na schematickém obrázku 5.29. Následně, ještě po několikanásobné verifikaci detekce palety, se provede manévr zajetí pod paletu za účelem naložení. Pro zvýšení spolehlivosti systému by mělo být navrženo doplnění ověření skutečné přítomnosti palety, například pomocí detekce čárového kódu, protože geometrická analýza by mohla být zmýlena vhodným rozestavěním libovolných objektů. Testy však ukázaly, že k takovému zmatení dochází jen zřídka a vyžaduje úmyslnou snahu o nařušení systému. Ještě před samotným zajetím pod paletu, tedy v době kdy se robot snaží správně napozicovat, je potřeba zmenšit bezpečnostní zóny na straně ROS2, aby bylo možné do takto úzkých prostor vjet.



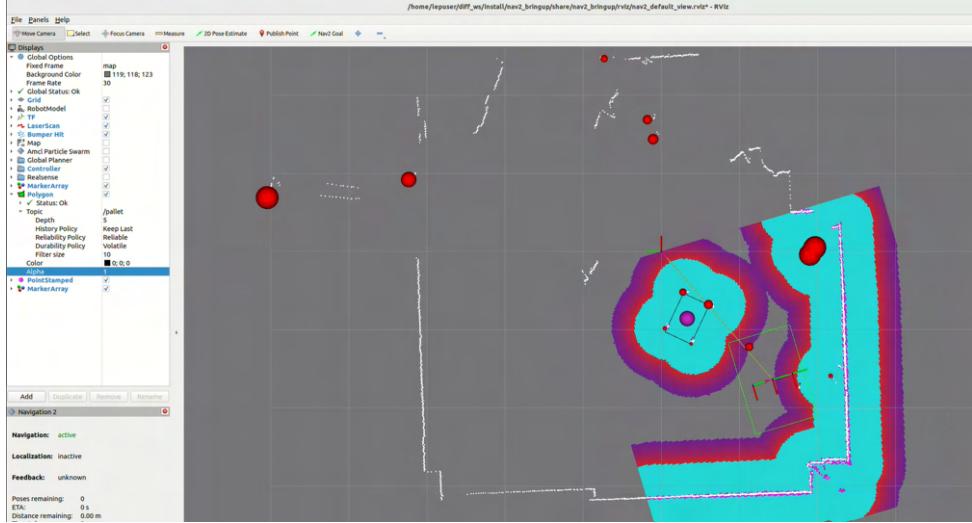
Obrázek 5.28: Schematické znázornění klíčových bodů palety, jejich geometrické konfigurace a vypočteného středu pro detekci pomocí DBSCAN a geometrické analýzy.



Obrázek 5.29: Schéma propojení komponent detekce palety se zbylým systémem.

Obrázek 5.30 poskytuje vizuální reprezentaci výsledků detekce malých objektů a palet ve vizualizačním nástroji RViz (viz sekce 3.9). Červené body na obrázku označují odhadované polohy a rozměry detekovaných objektů, což odpovídá shlukům identifikovaným DBSCAN algoritmem z dat LIDARu. Tato vizualizace umožňuje ověřit přesnost detekce a její soulad

s reálným prostředím. Doprovodná [Video 26](#) [23] demonstruje zpracování pohybu objektů v reálném čase včetně okamžité detekce palety při jejím přemisťování. V rámci videa je paleta znázorněna černým obdélníkem, který spojuje detekované shluky odpovídající nohám palety, přičemž geometrická konfigurace shluků je ověřena podle podmínek popsaných v sekci 5.5.2.



Obrázek 5.30: Vizualizace detekce malých objektů a palet v RViz, s červenými body označujícími odhadované polohy a rozměry objektů.

### 5.5.3 Potenciální rozšíření: Odhad pohybu objektů pomocí Kalmanova filtru

I když aktuální implementace detekce palet zahrnuje pouze statickou analýzu shluků z LIDARových dat, existuje potenciál využití detekce objektů v reálném čase pro rozšíření o odhad pohybu detekovaných objektů pomocí Kalmanova filtru, například pohybujících se lidí. [Video 27](#) [24] ukazuje, že chůze člověka může být velmi dobře zachycena pomocí trasování jeho nohou. Tento přístup by mohl výrazně zvýšit bezpečnost a efektivitu robota tím, že by predikoval trajektorie objektů v reálném čase, což může být velmi užitečné pro dynamické prostředí.

Kalmanův filtr je optimální lineární odhadovací metoda pro systémy s šumem, která kombinuje predikci stavu na základě modelu pohybu s aktualizací pomocí nových měření. Stav objektu by mohl být reprezentován vektorem obsahujícím pozici ( $X, Y$ ), rychlosť ( $\dot{X}, \dot{Y}$ ) a případně zrychlení:

$$\mathbf{x}_k = [X_{k,k}, \dot{X}_{k,k}]^T.$$

Model pohybu by byl definován stavovou rovnicí:

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{w}_k,$$

[23] [↑ https://youtu.be/UupdoPTlnb8](https://youtu.be/UupdoPTlnb8)

[24] [↑ https://youtu.be/95Joo2YPU7k](https://youtu.be/95Joo2YPU7k)

kde  $\mathbf{F}$  je matici přechodu stavu (např. pro model pohybu s konstantní rychlostí):

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$\Delta t$  je časový krok (např. 50 ms), a  $\mathbf{w}_k$  je šum procesu s kovarianční maticí  $\mathbf{Q}$ . Měření  $\mathbf{z}_k$  (pozice  $(x, y)$  z LIDARu) by bylo spojeno se stavem rovnicí:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k,$$

kde  $\mathbf{H}$  je matici pozorování (pro pozici):

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

a  $\mathbf{v}_k$  je šum měření s kovarianční maticí  $\mathbf{R}$ .

Tento filtr by iterativně prováděl predikci a aktualizaci stavu, což by umožnilo predikovat pohyb objektů a reagovat na ně před kolizí, například pomocí Collision Monitoru (viz sekce [5.1.2](#)). Přestože tento přístup nebyl v rámci této práce implementován, nabízí několik výhod:

- Zvýšení bezpečnosti tím, že predikuje pohyb dynamických překážek, jako jsou lidé nebo přesouvající se palety.
- Podpora reálného času díky nízké výpočetní náročnosti na platformách, jako je Raspberry Pi 4.
- Možnost integrace s navigačním stackem pro dynamické plánování trajektorií pro přizpůsobení dráhy pohybu objektu.

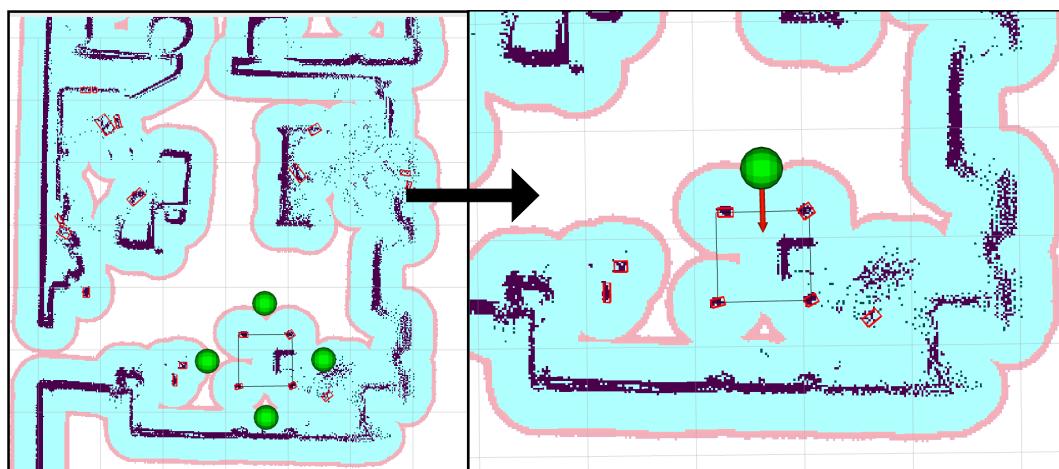
Omezení zahrnují potřebu pečlivého nastavení parametrů ( $\mathbf{Q}$ ,  $\mathbf{R}$ ) a citlivost na nelineární pohyby nebo rychlé změny prostředí, což by vyžadovalo rozšíření o Extended Kalman Filter (EKF) nebo další techniky. Tento přístup zůstává jako perspektivní směr pro budoucí vývoj, připravený pro implementaci na základě aktuálního zpracování LIDARových dat.

## 5.5.4 Aplikace na různých platformách

Tato kapitola se zaměřuje na praktickou implementaci a testování algoritmů detekce palety a manévrování na dvou odlišných platformách – podvozku DJI a AGV s velkým všeobecným podvozkem. Na podvozku DJI byla provedena komplexní řada testů zahrnujících detekci palety pomocí nákladové mapy a LiDARových dat, optimalizace transformace polohy a manévr zajetí pod paletu, čímž byl ověřen základní princip navrženého řešení. Na AGV s velkým všeobecným podvozkem byla stejná funkcionality implementována s využitím jeho pokročilejších technických vlastností, jako jsou kvalitní senzory a zdvihačí plošina, avšak testování zde bylo omezeno časovými a logistickými faktory. Obě platformy demonstrují potenciál navrženého přístupu pro reálné průmyslové aplikace, přičemž jejich výsledky naznačují směry pro další rozvoj řešení a optimalizaci.

### 5.5.4.1 DJI podvozek

Testování detekce palety probíhalo především na podvozku DJI. První experimenty byly zaměřeny na ověření funkčnosti detekce objektů za využití uzavřených namapovaných celků přímo z nákladové mapy prostřednictvím knihovny OpenCV (viz sekce 4.5.1). Tento přístup se ukázal jako funkční a umožnil vývoj algoritmu pro detekci palety v prostoru (viz obrázek 5.31). Na základě tohoto algoritmu byly provedeny první pokusy se zajetím pod paletu, což zachycuje Video 28<sup>[25]</sup>.



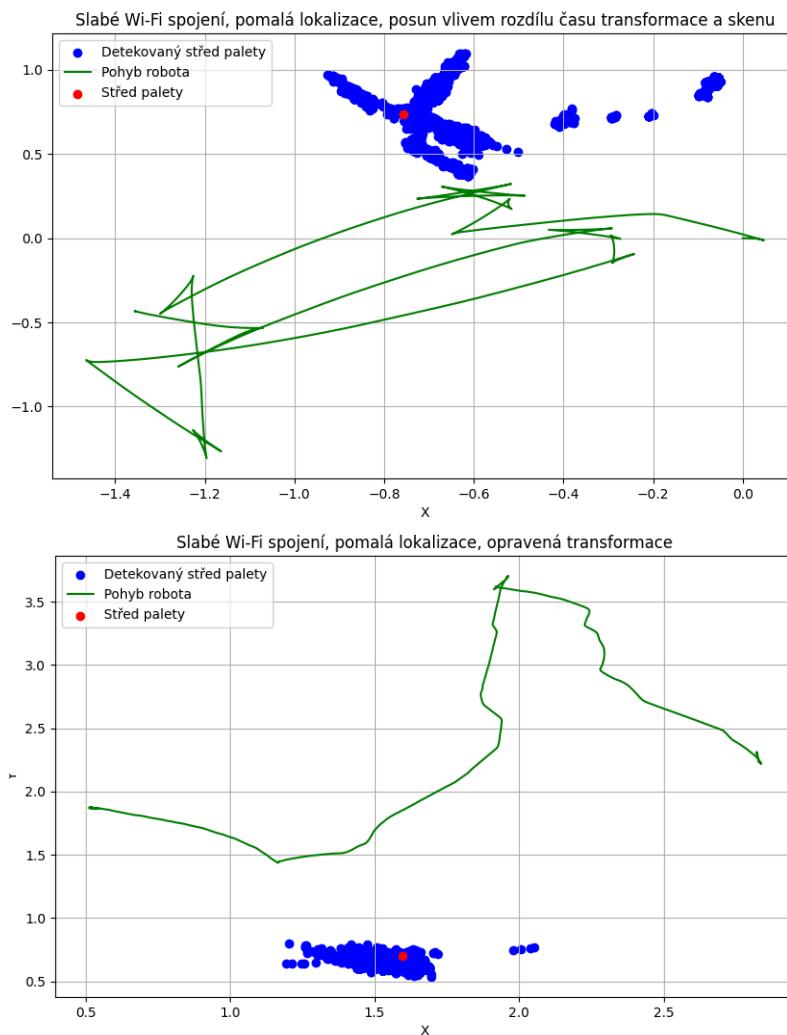
Obrázek 5.31: Detekce palety z nákladové mapy.

Prestože přístup založený na nákladové mapě potvrdil možnost detekce palety a manévrování k ní, ukázal se jako neefektivní. Zpracování velkého pole nákladové mapy bylo výpočetně náročné a vyžadovalo, aby byl daný objekt předem zmapován. Z tohoto důvodu byl zkoumán alternativní přístup využívající přímo data z LiDARů. Po analýze možností byla zvolena metoda shlukování s algoritmem DBSCAN (viz sekce 4.5.2.3). Tento algoritmus umožnil identifikaci potenciálních objektů z kontury LiDARového skenu a odhad jejich velikosti. Výhodou DBSCAN je jeho schopnost správně odhadnout tvar objektů, například sloupku, který LiDAR zachytí pouze částečně jako půlkruhový profil. Další předností je možnost provádět shlukování téměř v reálném čase, i když rychlosť zpracování závisí na výpočetní platformě. Výsledky této detekce byly již dříve znázorněny na obrázku 5.30. Testování ukázalo, že tento přístup umožňuje sledování i pohybujících se objektů, například nohou člověka, což otevřelo potenciál pro

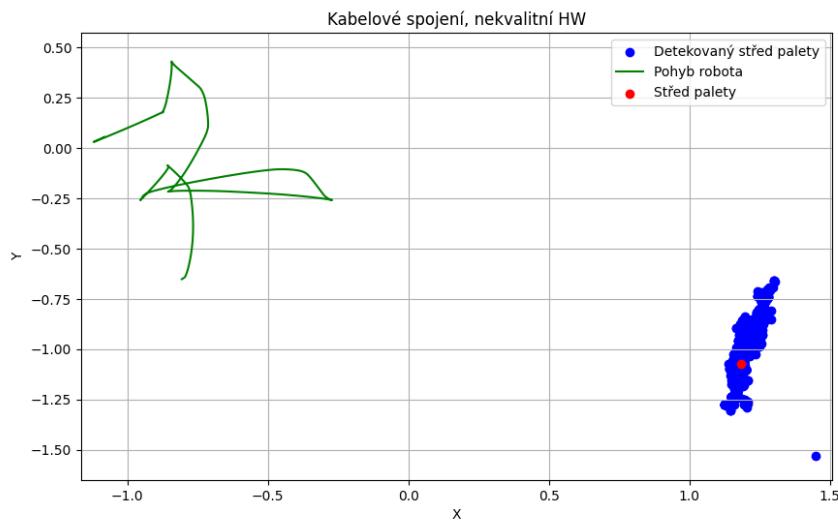
<sup>[25]</sup> ↑ <https://youtu.be/TdZPEyCgbRs>

budoucí využití při odhadu pohybu objektů a výpočtu jejich vektoru rychlosti pro predikci kolizí. Tato funkcionality však nebyla v rámci této práce implementována.

Rychlá detekce objektů pomocí DBSCAN umožnila efektivní odhad polohy palety. Po detekci palety bylo však nutné provést transformaci její polohy do souřadného systému mapy. Při pomalejší komunikaci nebo vyšší rychlosti pohybu robota docházelo k nepřesnostem v důsledku časového posunu mezi okamžikem pořízení LiDARových dat a odepslechem transformace mezi robotem (respektive LiDARem) a mapou. Čím delší byla prodleva při prohledávání detekovaných objektů, tím méně přesná byla výsledná poloha palety v mapě. Tento problém byl částečně vyřešen zlepšením kvality přenosu dat a optimalizací způsobu odepslechu transformace. Původní přístup, kdy byl odepslech transformace vynucen až po detekci palety, vedl k posunu, protože skeny byly vůči aktuální transformaci zastaralé. Výrazné zlepšení přinesla změna, při níž je transformace odepslouchávána pokaždé, když je k dispozici, takže časové razítko transformace poměrně přesně odpovídá okamžiku pořízení LiDARových dat. Porovnání přesnosti detekce palety před a po této optimalizaci je znázorněno na grafech (viz obrázky 5.32 a 5.33).

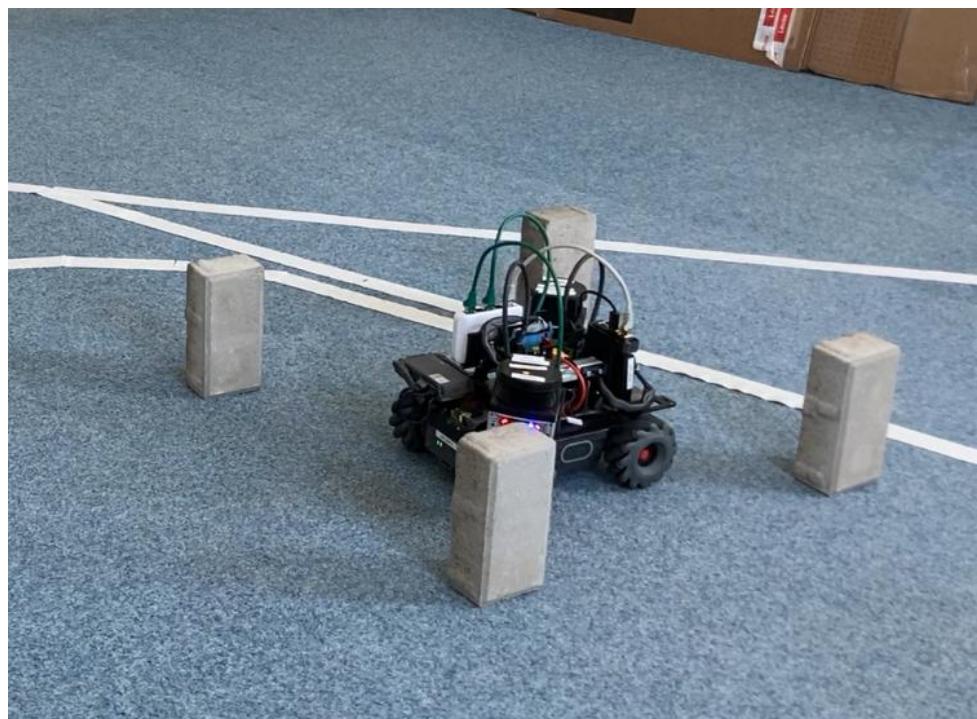


Obrázek 5.32: Grafy detekce palety při průběžném pohybu robota.



Obrázek 5.33: Graf detekce palety při průběžném pohybu robota.

I přes občasné nepřesnosti v detekci palety je možné úspěšně provést manévr zajetí pod paletu, která byla často simulována židlí nebo čtyřmi sloupek představujícími nohy palety (viz obrázek 5.34). Toho bylo dosaženo díky algoritmu, který nejprve navedl robota před paletu a následně čekal na opětovnou několikanásobnou detekci. Z průměru těchto detekcí byl vypočten střed palety, na základě čehož byl proveden samotný manévr zajetí. Celý tento proces je zřetelně ukazuje [Video 29](#) [26], na kterém jde vidět porovnání vizualizace v RViz, detekce palety a reality. Výsledky dalších z mnoha experimentů zachycuje [Video 30](#) [27].



Obrázek 5.34: Fotografie podvozku DJI při testování zajetí pod paletu.

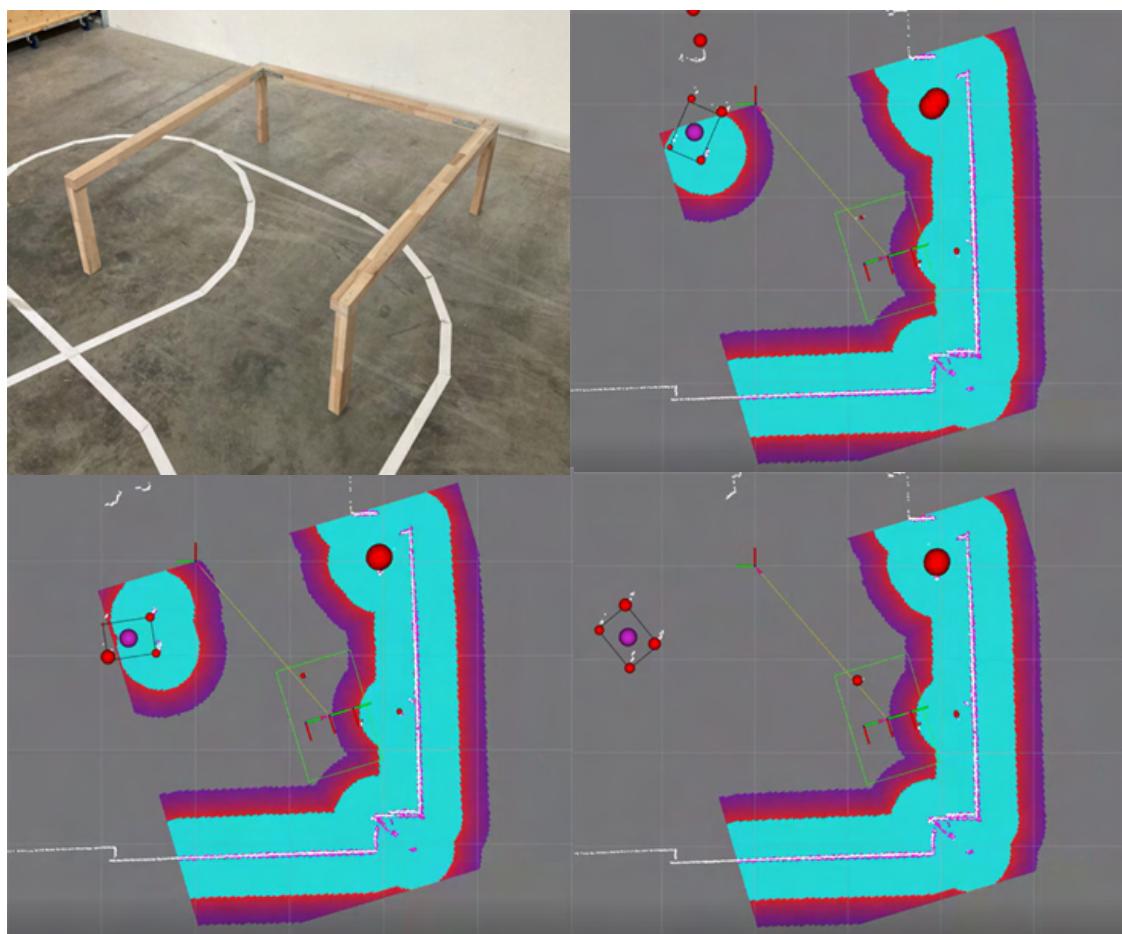
[26] ↑ <https://youtu.be/XpXaTvXdGPs>

[27] ↑ [https://youtu.be/diw-rZ\\_1rrY](https://youtu.be/diw-rZ_1rrY)

#### 5.5.4.2 AGV s velkým všesměrovým podvozkem

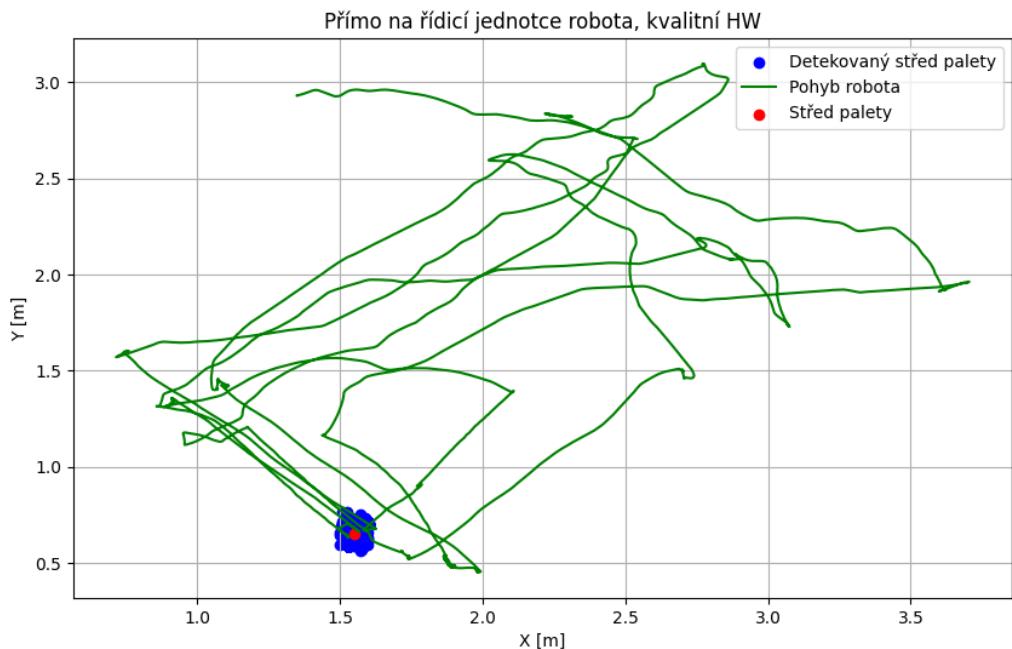
Funkcionalita detekce a manipulace s paletou byla na tomto podvozku implementována stejným způsobem jako ve finální verzi na podvozku DJI. AGV s velkým všesměrovým podvozkem však přináší několik výhod díky vyšší kvalitě jeho součástek, jako jsou motory a enkodéry. Ty zajišťují přesnější odometrii, což vede k výrazně nižším odchylkám při lokalizaci v mapě. Přesná lokalizace se logicky promítá do vyšší přesnosti detekce palety a následného manévrování. Kromě toho je AGV s velkým všesměrovým podvozkem vybaveno zdvihací plošinou, která umožňuje skutečné zdvihnutí a přepravu palety.. Před dokončením této práce však byla dostupnost tohoto podvozku omezená kvůli jeho využití při přípravách na veletrhy. AGV s velkým všesměrovým podvozkem představuje jednu z hlavních prezentačních platform firm, a proto bylo prioritně alokováno pro demonstrační účely.

Přes tato omezení byly provedeny přípravné kroky pro testování. Konkrétně byl navržen a vyroben stojan na paletu, pod který může AGV zajet a paletu zdvihnout (viz obrázek 5.35). Rovněž byla otestována detekce palety, a to stejným způsobem, jak tomu bylo u předchozích platform. Výsledky detekce byly velmi uspokojivé, což potvrzuje graf na obrázku 5.36. Tyto testy ukazují, že algoritmus detekce palety, optimalizovaný na základě zkušeností z podvozku DJI, je schopen spolehlivě fungovat i na této platformě.



Obrázek 5.35: Ukázka držáku na paletu a jeho detekce.

AGV s velkým všesměrovým podvozkem je vybaveno širokými bezpečnostními zónami, které jsou definovány podle průmyslových norem a automaticky upravovány prostřednictvím PLC. Pro úspěšné zajetí je nutné, aby PLC na základě povelu z ROS dočasně změnilo bezpečnostní zóny, což umožní robotovi přiblížit se k paletě, a současně nastavilo příslušná rychlostní omezení pro bezpečné provedení manévrů. V době psaní této práce však nebyl čas na implementaci a otestování této komunikace mezi ROS a PLC. Tato funkcionality tak zůstává nedokončena, ačkoliv její realizace je technicky přímočará a vyžaduje pouze doplnění odpovídajících povelů a podprogramů. Na straně PLC je pohyb v úzký prostorách již vyřešen pro SIMOVE, jak naznačuje obrázek 5.37.



Obrázek 5.36: Graf detekce palety při průběžném pohybu robota.

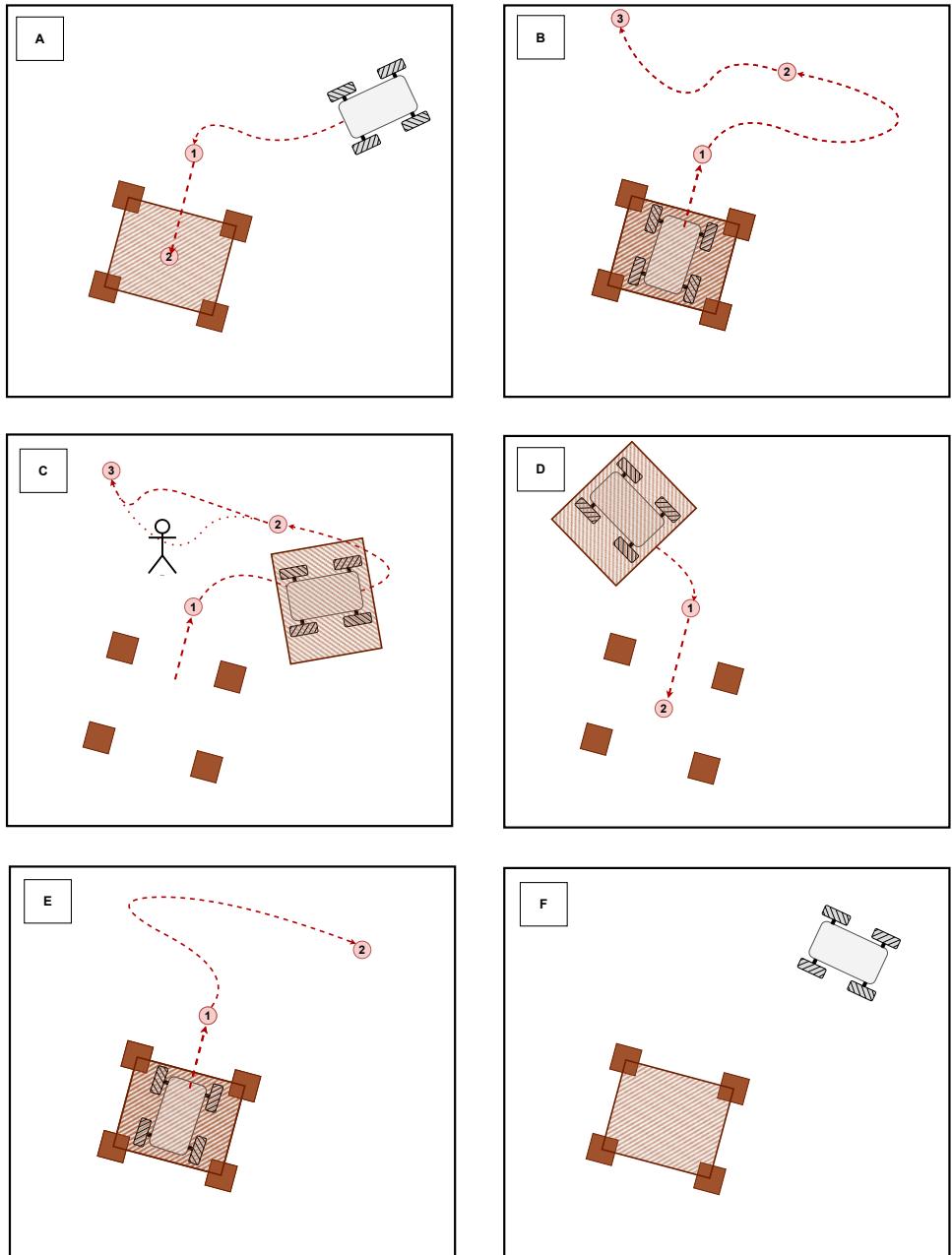
Navzdory tomu lze s vysokou jistotou předpokládat, že manévr zajetí pod paletu by na AGV s velkým všesměrovým podvozkem fungoval úspěšně. Tento předpoklad je založen na několika faktorech. Manévr byl úspěšně otestován na podvozku DJI, který má nižší kvalitu senzorů a odometrie. AGV s velkým všesměrovým podvozkem disponuje výrazně lepšími podmínkami pro přesné manévrování – díky kvalitním enkodérům a oficiálním verzím senzorů RSL200, které eliminují problémy s nepřesným spojováním skenů. Na rozdíl od podvozku DJI, kde vývojové verze senzorů RSL200 občas vedly k chybám v měření a tím i k nepřesnosti v lokalizaci a detekci palety, AGV s velkým všesměrovým podvozkem využívá stabilní a kalibrované senzory, což zajišťuje konzistentní a spolehlivá data. Mechanická konstrukce velkého AGV, včetně zdvižací plošiny, je navržena přímo pro manipulaci s paletami, což dále zvyšuje pravděpodobnost úspěšného provedení manévrů. Tyto faktory společně naznačují, že AGV s velkým všesměrovým podvozkem má potenciál provádět velmi přesné manévrů a dosahovat požadované pozice a orientace s vysokou spolehlivostí.



Obrázek 5.37: Ilustrační fotografie velkého AGV podjíždějícího paletu

Pro otestování byl naplánován následující testovací scénář:

- A Robot stojí někde v prostoru a paleta je také umístěna na určitém místě. Detekce palety je zapnuta. Robot přijede před paletu, zorientuje se a poté pod ni vjede.
- B Robot je pod paletou, ověří si správnou pozici, zvedne paletu a naplánuje trasu, kterou s paletou pojede.
- C Robot opatrně vyjede zpod palety, vypne detekci palety a jede po připravené trase. Pokud někdo vstoupí do trasy nebo se objeví překážka, robot trasu přeplánuje.
- D Robot dorazí tam, kam měl. Zastaví se a znova zapne detekci palety. Když detekuje paletový regál, přijede k němu a zajede pod něj.
- E Robot zkontroluje pozici a položí paletu zpět na regál. Poté se opět připraví na manévr po trase.
- F Robot je znova někde v prostoru. Scénář lze opakovat. Pokud je to žádoucí, lze paletu vzít a přemístit ji jinam, aby se prokázala automatická detekce. Zde bychom chtěli vyřešit, aby byl vždy volný prostor pro průchod jednotlivými body trasy (kroužky s čísly). Body trasy lze zadat ručně, ale nevím, zda je to ideální pro expo – byl by potřeba operátor. Pokud by body byly pevně definované, paleta by se musela přemisťovat jen na volná místa.



Obrázek 5.38: Testovací scénář

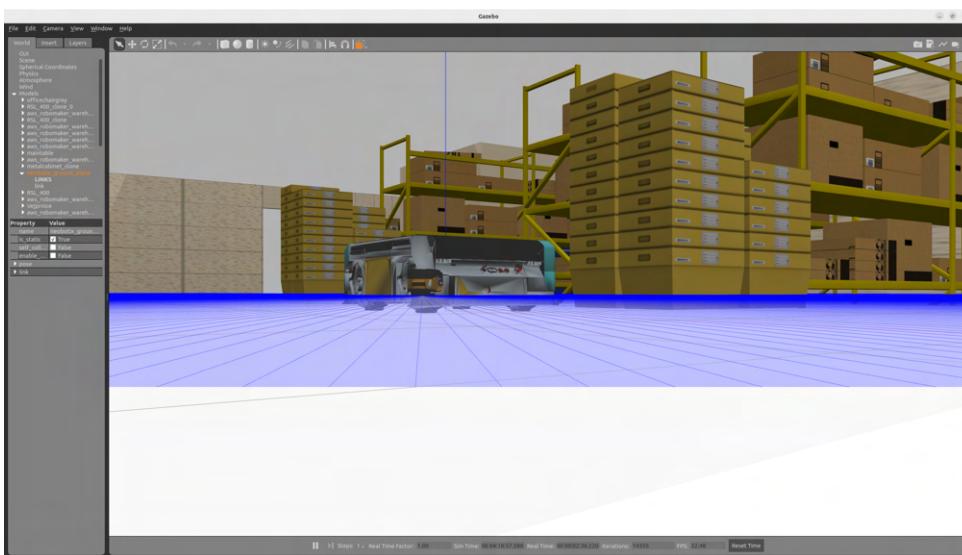
## 5.6 Simulace

Před nasazením navržených algoritmů a navaigacích postupů na reálnou robotickou platformu je nezbytné ověřit jejich funkčnost v kontrolovaném prostředí simulace. Tento přístup umožňuje identifikovat potenciální problémy a vyloučit chyby v softwarových komponentech dříve, než by mohly způsobit komplikace v praxi. Simulace navíc usnadňuje ladění výpočtů, regulaci a plánování trajektorií bez rizika poškození hardwaru nebo ohrožení bezpečnosti. Vzhledem k tomu, že ROS2 poskytuje robustní nástroje pro tvorbu simulačních prostředí, byla v rámci této práce využita platforma Gazebo (viz sekce 3.8) k testování různých scénářů autonomní navigace, včetně virtuálních drah, regulace pohybu a plánování trajektorií. Tato kapitola popisuje vytvoření simulačního modelu robota, konfiguraci senzorů a prostředí a analyzuje specifika a limity tohoto přístupu.

### 5.6.1 Model robota a simulace senzorů

Pro potřeby simulace byl v Gazebo vytvořen model robota, který zahrnuje dva LIDARy s parametry odpovídajícími reálným senzorům RSL200 a RSL400 (podrobnosti viz kapitoly 2.1.2 a 2.1.1). Tyto parametry zahrnují frekvenci skenování, maximální dosah měření a úhlové rozlišení. Simulace LIDARu v Gazebo je založena na principu vysílání paprsků z definované pozice a zachycení jejich odrazu od povrchu kolidujících objektů v prostředí. Výsledná data jsou převedena do formátu *PointCloud* a publikována v ROS2 jako zprávy, což zajišťuje kompatibilitu s reálnými daty získanými z fyzických senzorů.

Pro zvýšení autentičnosti byl do simulovaných měření přidán Gausovský šum, jehož parametry (střední hodnota 0 a rozptyl odvozený z charakteristik reálných LIDARů) napodobují nejistotu měření v praxi. Vzhledem k přítomnosti dvou simulovaných LIDARů bylo nutné implementovat jejich sloučení do jednoho virtuálního senzoru (viz sekce 4.2). Tento proces zahrnuje synchronizaci dat a jejich spojování v reálném čase, což zavádí mírnou latenci odpovídající skutečným podmínkám. Latence, typicky v řádu milisekund, je dána výpočetní náročností algoritmu sloučení a ovlivňuje dynamiku navaigacích reakcí robota, čímž simulace lépe odpovídá realitě. Model robota i se simulovanými LIDARy je ukázán na obrázku 5.39

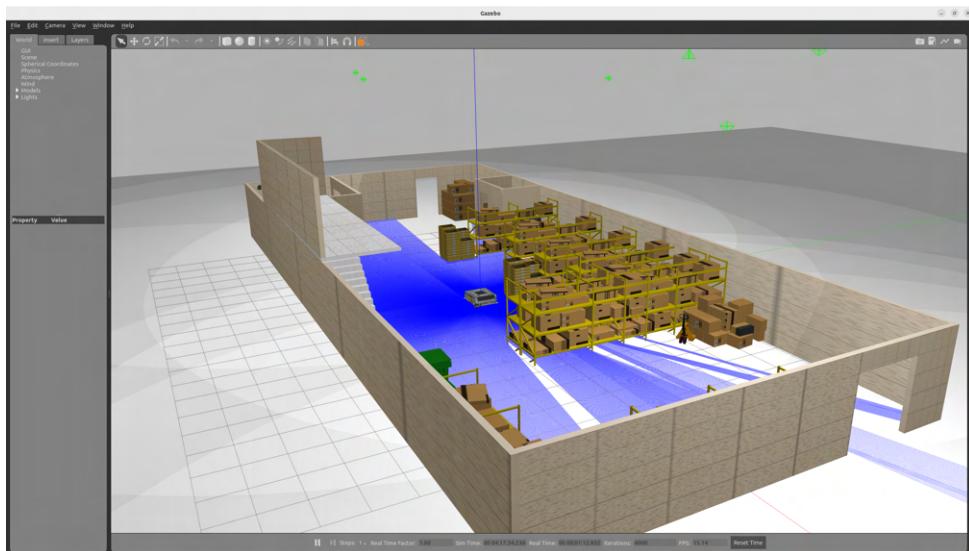


Obrázek 5.39: Model robota

Robot v simulaci je řízen zadáváním rychlostí jednotlivých kol, které jsou určeny na základě algoritmů regulace a plánování trajektorie. Zpětná vazba o poloze a rychlosti je odvozována z kinematického modelu robota, avšak absence simulace nepřesnosti enkodérů vede k příliš ideálnímu odhadu odometrie. V reálném prostředí jsou data z enkodérů ovlivněna faktory, jako je skluz kol nebo mechanické tolerance, což zvyšuje nejistotu lokalizace. V Gazebo není tento efekt modelován, což omezuje věrnost simulace procesů lokalizace, například algoritmu AMCL, který v reálných podmírkách provádí korekci odometrických chyb pomocí senzorových měření. Zde jej není možné dobře odladit, protože odometrie je příliš dokonalá.

## 5.6.2 Simulační prostředí

Simulační prostředí bylo navrženo tak, aby odpovídalo reálné skladové hale (viz obrázek 5.40), v níž jsou testovány navigační scénáře. Model zahrnuje základní prvky, jako jsou stěny, sloupy, regály apod., a umožňuje simulovat dynamické překážky, například pohybující se osoby (ručním vložením objektu během simulace). Prostor byl vytvořen s důrazem na realistické rozměry a odpovídající fyzikální vlastnosti povrchů (koeficient tření, odrazivost). To umožnilo testovat manévry, jako je objízdění překážek, následování virtuálních drah nebo přesné zastavení na určených bodech, v podmírkách blízkých těm, které robot potká v praxi.



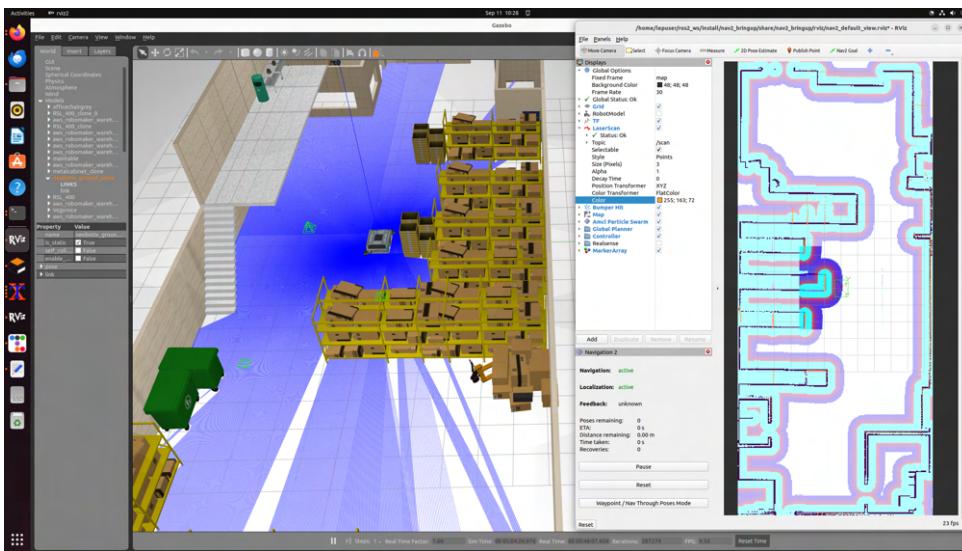
Obrázek 5.40: Model simulačního prostředí

## 5.6.3 Testované scénáře

Simulace v prostředí Gazebo byla využita k ověření několika klíčových scénářů autonomní navigace a jejich dílčích komponent. Níže jsou podrobně popsány jednotlivé testované scénáře, které zahrnují autonomní navigaci, plánování trajektorie, jízdu po virtuálních čarách, kombinaci autonomní navigace s virtuálními čarámi, regulaci pohybu a jízdu po virtuálních drahách. Každý scénář byl navržen tak, aby ověřil specifické aspekty navigačního systému a jeho robustnost v simulovaném prostředí, přičemž výsledky poskytují cenné poznatky pro další vývoj a nasazení na reálných platformách.

### 5.6.3.1 Autonomní navigace

Tento scénář se zaměřuje na testování schopnosti robota autonomně navigovat v otevřeném prostoru směrem k zadaným cílovým bodům. Simulace využívá data z virtuálního LiDARu k detekci překážek a plánování trajektorií. Pro výpočet optimální cesty byl použit algoritmus NavFnPlanner (viz sekce 3.6), který zohledňuje dynamická omezení robota, jako jsou maximální rychlosť a zrychlení, a zároveň zajišťuje bezpečné vyhýbání se překážkám. Robot úspěšně navigoval k zadaným cílům, přičemž simulace umožnila ověřit robustnost algoritmu v různých prostředích, včetně prostorů s dynamickými překážkami. Vizualizace tohoto scénáře v prostředí Gazebo a RViz je znázorněna na obrázku 5.41, případně podrobněji vše ukazuje Video 31 [28].



Obrázek 5.41: Zachycení simulace v Gazebo a ovládání a vizualizace v RViz při aplikaci autonomní navigace.

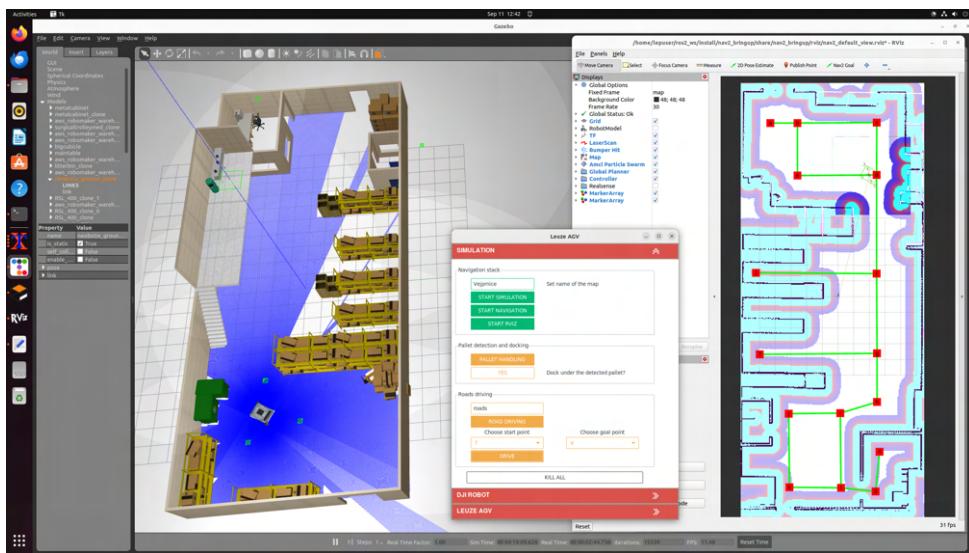
Scénář plánování trajektorie byl navržen k ověření algoritmů pro výpočet optimální cesty v simulovaném prostředí. Testování zahrnovalo zohlednění různých omezení, jako jsou maximální rychlosť robota, bezpečnostní zóny detekované virtuálním LiDARem a přítomnost překážek v mapě. Simulace umožnila analyzovat, jak algoritmy plánování reagují na změny v prostředí, například při objevení nových překážek, a jak efektivně optimalizují trasu s ohledem na bezpečnost a časovou efektivitu. Výsledky ukázaly, že navigační stack je schopen generovat plynulé a bezpečné trajektorie.

Tento scénář se zaměřuje na testování přesnosti regulace pohybu robota, konkrétně zadávání rychlostí na kola a jejich dopadu na stabilitu a přesnost trajektorie. Simulace v Gazebo umožnila analyzovat, jak robot reaguje na různé rychlostní profily a jak se tyto změny promítají do jeho pohybu. Během testování bylo vyzkoušeno několik typů regulátorů, přičemž algoritmus MPPI se ukázal jako nejlepší varianta pro tuto aplikaci díky své schopnosti zajistit plynulé a robustní řízení. Díky absenci nepřesnosti enkodérů, které jsou typické pro reálné platformy, byla regulační v simulaci příliš ideální, což vedlo k nerealistickému hladkému pohybu. Tento poznatek naznačuje potřebu budoucí úpravy simulačního modelu, například přidáním simulovaného šumu do odometrických dat, aby lépe odpovídaly reálným podmínkám. Scénář přesto poskytl cenné informace o chování regulačních algoritmů a jejich nastavení pro dosažení optimální stability trajektorie.

[28] ↑ <https://youtu.be/Kb0JmeryOV4>

### 5.6.3.2 Jízda po virtuálních čarách

Tento scénář ověřuje schopnost robota pohybovat se po předem definovaných virtuálních čarách, které představují pevně stanovené dráhy nebo silnice v simulovaném prostředí. Na rozdíl od standardní autonomní navigace, kde je trajektorie plánována dynamicky, je v tomto případě trajektorie předem určena a robot ji sleduje za využití algoritmu MPPI a existujícího systému autonomní navigace. Plánování trajektorie je tedy vynecháno, protože robot pouze následuje zadanou cestu, přičemž MPPI zajistí plynulé a robustní řízení s ohledem na dynamická omezení robota. Kromě toho byl testován mechanismus reakce na překážky: pokud virtuální LiDAR detekoval překážku na trase, robot zastavil a čekal na uvolnění cesty. Tento scénář potvrdil, že systém autonomní navigace v kombinaci s MPPI je schopen spolehlivě následovat předem definované trasy a efektivně reagovat na neočekávané překážky, a hlavně umožnil odladit GUI, hledání v síti drah a vlastní povelování. Vizualizace tohoto scénáře v prostředí Gazebo a RViz je znázorněna na obrázku 5.42 a dále vše ukazuje Video 32 [29].



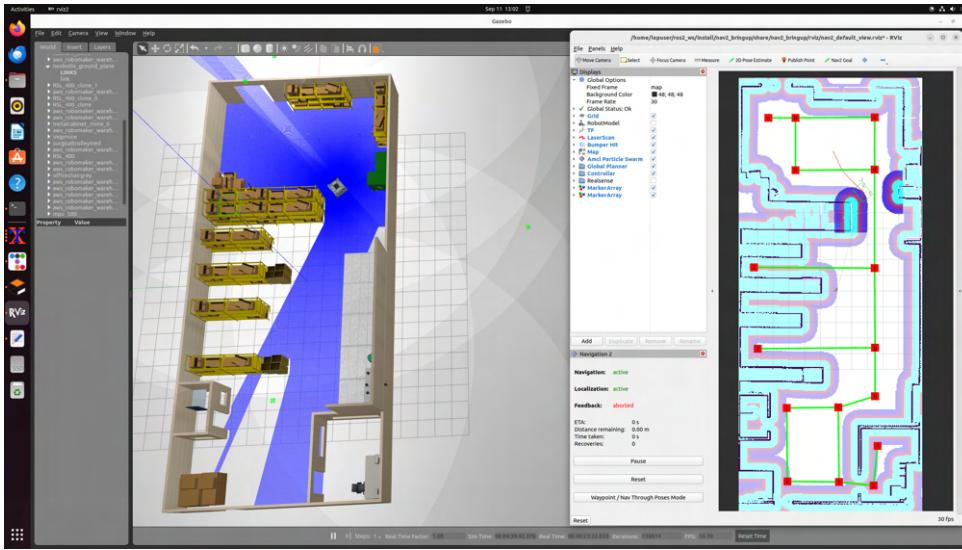
Obrázek 5.42: Zachycení simulace v Gazebo a ovládání a vizualizace v RViz při aplikaci jízdy po virtuálních čarách.

### 5.6.3.3 Kombinace autonomní jízdy a virtuálních čar

Scénář kombinace autonomní jízdy a virtuálních čar zkoumá možnost přepínání mezi těmito dvěma režimy navigace v rámci jednoho prostředí. Robot nejprve využívá autonomní navigaci k dosažení nejbližšího bodu na předem definované virtuální čáre, přičemž se vyhýbá překážkám a optimalizuje trasu. Po dosažení virtuální čáry přechází do režimu následování pevné trasy, kde dynamicky reguluje rychlosť a reaguje na překážky podle pravidel definovaných pro virtuální čáry. Tento scénář simuluje reálné průmyslové aplikace, kde je potřeba kombinovat flexibilitu autonomní navigace s přesností pohybu po pevných trasách, například při dopravě materiálu ve skladech. Simulace potvrdila funkčnost přepínání mezi režimy a schopnost robota plynule přecházet mezi různými navigačními strategiemi tak, aby mohly být spouštěné zároveň a odstranily se nepříjemné prodlevy během inicializace. Vizualizace tohoto scénáře v prostředí Gazebo a RViz je znázorněna na obrázku 5.43 a kompletní proces opět ukáže lépe Video 33 [30].

[29] ↑ [https://youtu.be/u1\\_5eXqmrBQ](https://youtu.be/u1_5eXqmrBQ)

[30] ↑ <https://youtu.be/gN5MGQDTLaU>



Obrázek 5.43: Zachycení simulace v Gazebo a ovládání a vizualizace v RViz při kombinování autonomní navigace a jízdy po virtuálních čarách.

#### 5.6.3.4 Virtuální dráhy

Scénář virtuálních drah ověřuje schopnost robota pohybovat se po předem nakreslených trasách za využití algoritmu Pure Pursuit, jak bylo popsáno v předchozích kapitolách. Simulace zahrnovala cyklické následování pevně definované trajektorie, přičemž robot dynamicky upravoval rychlosť a orientaci podle geometrie trasy. Tento scénář byl navržen s ohledem na demonstrační a testovací účely, například pro ověření dlouhodobé stability navigace a přesnosti lokalizace v simulovaném prostředí. Dále byla testována reakce na překážky, kdy robot zastavil a čekal na uvolnění trasy, což simuluje reálné scénáře v průmyslových provozech. Simulace potvrdila robustnost algoritmu Pure Pursuit a jeho vhodnost pro aplikace, kde je vyžadován přesný pohyb po pevných trasách. Zároveň pomohla s počátečním odladěním regulátoru. Vizualizace tohoto scénáře v prostředí Gazebo je znázorněna na obrázku 5.44. Tento obrázek pochází přímo z Video 34<sup>[31]</sup>, které kromě simulace jízdy po virtuální čáře ukazuje i odhad počáteční polohy a následnou kompenzaci algoritmem AMCL.



Obrázek 5.44: Zachycení simulace při aplikaci jízdy po virtuální dráze.

<sup>[31]</sup>↑ <https://youtu.be/hkSZ4L7MABU>

## 5.6.4 Shrnutí

Simulace v prostředí Gazebo prokázala funkčnost navržených algoritmů pro autonomní navigaci, jízdu po virtuálních drahách a čarách, regulaci pohybu a plánování trajektorií. Robot v simulovaném prostředí skladové haly spolehlivě plnil zadané úkoly, jako je navigace k cílovým bodům, vyhýbání se překážkám nebo následování předem definovaných tras. Data z virtuálního LiDARu umožňují přesnou detekci překážek a efektivní plánování tras, zatímco integrace s ROS2 zajišťuje plynulou komunikaci mezi simulačními komponentami a navigačním softwarem. Tato integrace usnadňuje přenos výsledků do reálné implementace a potvrzuje kompatibilitu navrženého systému s ROS2 frameworkm, což je klíčové pro jeho budoucí nasazení na reálných platformách.

Navzdory těmto úspěchům simulace vykazuje několik omezení, která je třeba zohlednit při interpretaci výsledků. Absence modelování nepřesnosti enkodérů v simulačním prostředí vede k nerealisticky přesné odometrii, což ovlivňuje věrnost lokalizačních algoritmů, jako je AMCL. Tento nedostatek může vést k odchylkám při přenosu algoritmů na reálné platformy, kde jsou nepřesnosti enkodérů běžné. Simulace sice zohledňuje latenci způsobenou slučováním dat z LiDARů, čímž částečně napodobuje reálné podmínky, avšak fyzikální interakce, jako je skluz kol, deformace povrchu nebo vliv nerovností terénu, nejsou plně zahrnuty. Tyto faktory mohou v praxi významně ovlivnit chování robota a způsobit odchylky, které simulace nezachytí. Kromě toho je výpočetní náročnost simulace, zejména při zpracování dat ze dvou virtuálních LiDARů a složitého prostředí skladové haly, značná. Tato náročnost klade vysoké požadavky na hardware a může omezit použitelnost simulace na méně výkonných systémech, což je třeba zohlednit při plánování dalšího vývoje.

Simulace v Gazebo představuje klíčový krok k ověření navigačních algoritmů a postupů před jejich nasazením na reálné platformy. Vytvořený model robota vybavený dvěma virtuálními LiDARy a realistické prostředí skladové haly umožnily testování široké škály scénářů, od autonomní navigace přes regulovalý pohyb po virtuálních čarách až po cyklické následování virtuálních drah. I přes zmíněná omezení simulace poskytla cenné poznatky o chování navigačního systému v kontrolovaných podmírkách a potvrdila jeho připravenost pro další vývoj. Výsledky simulace tvoří pevný základ pro optimalizaci algoritmů, zejména v oblastech lokalizace a regulace pohybu, a jejich přizpůsobení reálným podmírkám, kde budou muset čelit fyzikálním a senzorickým omezením. Tím simulace významně přispívá k celkovému cíli této práce, kterým je vývoj robustního a spolehlivého systému autonomní navigace pro průmyslové aplikace.

# Kapitola 6

## Závěr

---

Tato práce představuje komplexní příspěvek k vývoji autonomních navaigacích systémů pro AGV (Automated Guided Vehicles), případně AMR (Autonomous Mobile Robots) s důrazem na jejich praktickou implementaci a testování v reálných podmínkách. Výsledkem je podrobný návod, jak navrhnout a postavit AGV s různými typy navigace, kde základem je ROS2 a přidružené knihovny, jako je Nav2 pro autonomní navigaci. Práce pokrývá široké spektrum navaigacích přístupů – od tradiční jízdy po čáře přes virtuální dráhy až po plně autonomní navigaci – a poskytuje ucelený pohled na jejich implementaci, testování a optimalizaci. Součástí práce je také analýza senzorů, návrh regulátorů, vývoj softwaru a zhodnocení bezpečnostních aspektů, což činí tuto práci cenným zdrojem informací pro vývojáře i průmyslové inženýry.

Během vývoje byly vytvořeny čtyři prototypy AGV, každý zaměřený na ověření specifických aspektů navaigacního systému. První prototyp, podvozek DJI, sloužil jako testovací platforma pro funkčnost senzorů, jejich integraci do ROS2 a optimalizaci veškerých algoritmů. Druhý prototyp, diferenciální podvozek, byl využit k testování bezpečnostních funkcí zajištěných LiDARy a PLC a k ověření a demonstraci jízdy po čáre ve skladových prostorách. Třetí prototyp, AGV s velkým všeobecným podvozkem s mecanum koly, představuje hlavní výsledek práce. Umožnil ověření algoritmů z DJI na kvalitnějším hardwaru a integraci kompletního autonomního řídicího systému včetně bezpečnostních prvků, díky čemuž je použitelný v průmyslových aplikacích. Čtvrtý prototyp, diferenciální podvozek TinyAGV, byl vybaven systémem sledování cyklických virtuálních drah, primárně pro prezentační účely, avšak s potenciálem pro testování senzorů, lokalizace a odometrie. Doplňkově byl vytvořen simulační model v Gazebo, který poskytl kontrolované prostředí pro analýzu algoritmů autonomní navigace a virtuálních drah před jejich nasazením na reálné platformy. Tyto prototypy společně demonstруjí široké možnosti navrženého systému a jeho adaptabilitu různým požadavkům.

Významným výsledkem práce je podrobný popis práce se senzory a vývoj driverů pro ROS2, doplněný ukázkovými zdrojovými kódy dostupnými na GitHubu [35], které slouží jako inspirace pro tvorbu driverů pro různé senzory. Drivery byly navrženy pro senzory firmy Leuze, která tuto práci zaštiťuje, a zahrnují například zpracování dat ze senzorů OGS a DCR přes sériovou linku. Senzory byly důkladně testovány z hlediska jejich použitelnosti pro různé aplikace od jízdy po čáre přes detekci palet až po autonomní navigaci, přičemž prokázaly spolehlivost i v náročných podmínkách, například při proměnlivém osvětlení nebo opotřebovaných čarach. Testování přineslo cenné poznatky pro další vývoj senzorů v kontextu AGV, včetně možnosti integrace některých pokročilých funkcí přímo do senzorů, což by mohlo zjednodušit budoucí implementace.

Práce poskytuje stručný úvod do ROS2, který usnadňuje začátečníkům pochopení základních konceptů a nástrojů, jako jsou topicy, nody či konfigurace navaigacního stacku Nav2. Nabízí praktické tipy a postupy založené na vlastních zkušenostech, jež zkracují čas potřebný k osvojení systému. Dále řeší základní problémy při návrhu navaigacních systémů včetně konfigurace senzorů, ladění regulátorů, optimalizace algoritmů a zajištění bezpečnosti. Tím slouží jako užitečný průvodce pro začínající vývojáře i zkušené inženýry, kteří hledají efektivní a srozumitelné řešení.

Výsledkem práce je autonomní řídicí systém, nasaditelný na libovolný podvozek disponující LiDARy pro detekci překážek a lokalizaci, enkodéry pro odometrii, SIEMENS PLC zajišťujícím kinematický model a bezpečnostní funkce, a dostatečným výpočetním výkonem pro zpracování dat v reálném čase. Systém umožňuje plně autonomní navigaci – po zadání cíle robot naplánuje trasu, bezpečně ji projede a optimalizuje pohyb podle dynamických omezení. Součástí systému je i sledování virtuálních drah. Po doplnění senzorů systém podporuje jízdu po čáre s navrženým PD regulátorem pro stabilní sledování trasy. Testování na senzorech Leuze přineslo zkušenosti s jejich aplikací na AGV, což bylo jedním z cílů práce. Dále je kladen důraz na bezpečnost zajištěnou integrací LiDARů a PLC od SIEMENS splňujících příslušné bezpečnostní normy. Tento systém umožňuje okamžité zastavení AGV při narušení bezpečnostních zón a bezpečný provoz v prostředích s lidmi.

Mezi inovativní přístupy práce patří detekce palet z LiDARových skenů, implementovaná a otestovaná na podvozku DJI a AGV s velkým všeobecným podvozkem. Tento přístup využívá LiDARová data a algoritmy shukování k určení polohy palet, což robotovi umožňuje zajet pod paletu a manipulovat s ní. Výhodou je robustnost v prostředích s proměnlivým osvětlením, kde kamerové systémy selhávají, bez potřeby dodatečných senzorů. Dalším zajímavým přístupem je generování virtuálních drah, které kombinuje předvídatelnost pevných tras s flexi-

bilitou autonomní navigace. Tento přístup byl otestován v simulaci i na reálných platformách a ukázal svůj potenciál pro aplikace, kde je potřeba rychle měnit trasy bez fyzických zásahů do prostředí. Práce poskytuje podrobný popis a matematické odvození klíčových algoritmů – plánování trajektorií, regulace pohybu, detekce palet nebo lokalizace, což usnadňuje pochopení principů a tím umožňuje přesnou parametrizaci pro specifické aplikace. Tím je zajištěna centralizace všech nezbytných informací v jednom dokumentu. Popisy zároveň slouží jako základ pro implementaci v ROS2 nebo jiných prostředích, čímž zvyšují praktickou využitelnost práce.

Tato práce přispívá k rozvoji autonomních navaigačních systémů pro AGV prostřednictvím praktických návodů, otestovaných prototypů, optimalizovaných algoritmů a poznatků z reálného nasazení. Její výsledky mohou zlepšit průmyslové aplikace a inspirovat další výzkum v autonomní robotice, zejména v propojení tradičních a moderních navaigačních metod. Práce tak splnila svůj hlavní cíl – získání zkušeností s AGV, testování senzorů firmy Leuze a vyvinutí kompletního řídicího systému v ROS2 s různými typy navigace včetně inovativního přístupu pro detekci palet.

# **Appendices**

## Příloha A

### Porovnání s komerčním řešením SIMOVE

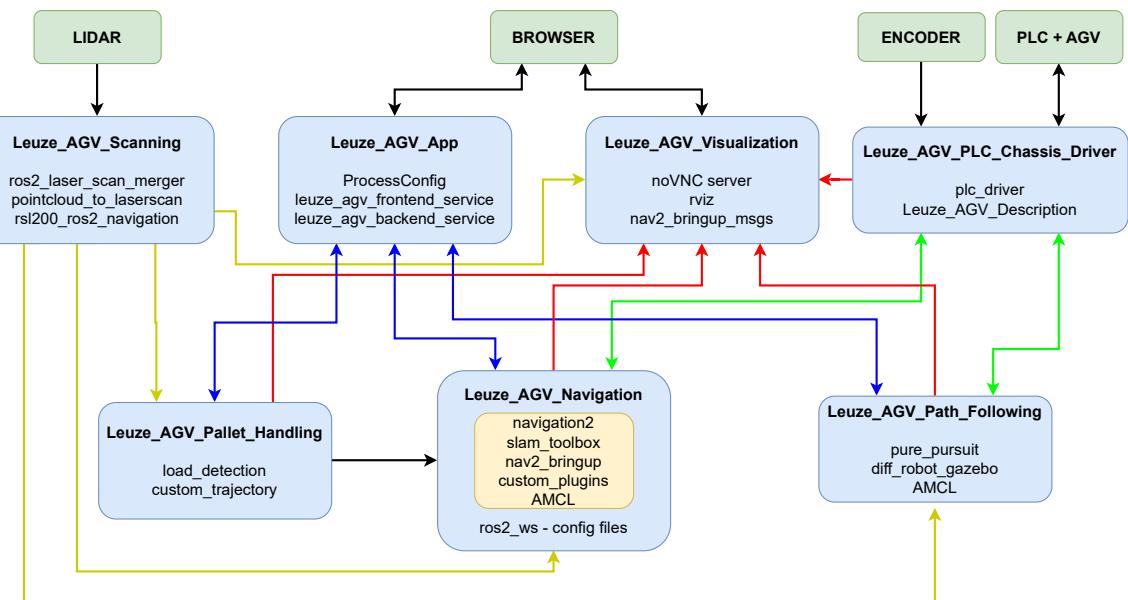
---

Na začátku této práce byl zmíněn SIMOVE jako existující komerční varianta řídicího systému pro AGV a AMR, která je navržena pro nasazení prostřednictvím PLC a specializovaného hardwaru od společnosti SIEMENS. SIMOVE představuje uzavřené řešení, které je optimalizováno pro specifické funkce, jako je jízda po předem definovaných trasách, základní detekce překážek a jednoduchá správa flotily. Na jednu stranu tato uzavřenosť nevadí, protože SIMOVE pokrývá logicky nejběžnější požadavky zákazníků, jako je přesné následování pevných tras v průmyslových provozech nebo automatizované zastavování na stanovištích. Na druhou stranu však uzavřenosť systému představuje významnou nevýhodu – není možné nad SIMOVE provádět jakýkoliv vývoj, přidávat nové funkcionality ani přizpůsobovat systém specifickým potřebám uživatele. Tato omezení vedla k vývoji vlastního řídicího systému, který rovněž umožňuje zahrnout například LiDARy libovolné značky, avšak je zcela otevřený a poskytuje flexibilitu pro přidávání dalších funkcionalit a jejich vzájemné propojení. Tato otevřenosť přináší obrovskou výhodu, protože umožňuje implementovat nové a inovativní přístupy. Přesně tak byly k navigačnímu systému přidávány další nadstavby a funkční balíčky, jako například detekce palet z LiDARových skenů, jízda po virtuálních čarách nebo generování virtuálních drah, což SIMOVE ve své standardní podobě nenabízí.

Z hlediska připravenosti pro nasazení je SIMOVE v současnosti lépe přizpůsoben pro uživatele bez hlubších technických znalostí. Systém poskytuje automatickou konfiguraci skenerů, tedy vyladění odhadu jejich vzájemné pozice, což zajišťuje přesné měření a minimalizuje potřebu manuálních zásahů. Tato funkce je zvláště užitečná při inicializaci AGV v novém prostředí, protože zjednodušuje proces kalibrace a umožňuje rychlé uvedení robota do provozu. SIMOVE také nabízí předdefinované šablony pro běžné scénáře, jako je jízda po čáre nebo základní vyhýbání se překážkám, což usnadňuje nasazení v prostředích s pevně danými trasami. Na druhé straně však SIMOVE trpí nevýhodou v podobě obrovského množství parametrů, které je potřeba nastavit a které jsou často velmi neintuitivně pojmenovány, což může být pro uživatele matoucí a časově náročné. Například konfigurace bezpečnostních zón nebo nastavení rychlostních profilů vyžaduje detailní pochopení interní logiky systému, což může být překážkou pro méně zkušené uživatele. V případě navrženého řídicího systému je rovněž potřeba provádět jisté konfigurace, ale základní prvky jsou připravené tak, aby fungovaly obecně. To znamená, že je nutné upravit například rozměry podvozku nebo vzájemné polohy senzorů (bohužel zatím na

více místech, protože není dokončen jednotný konfigurační systém). Tento proces je v současnosti manuální a vyžaduje úpravy v konfiguračních souborech, což může být časově náročnější ve srovnání s automatickou kalibrací SIMOVE. Avšak je v plánu připravit jednotnou konfiguraci s co nejmenším počtem parametrů, přičemž v ideálním případě se některé z nich, například vzájemné polohy senzorů, automaticky dopočítají na základě inicializačního skenování prostředí. Tím by se výrazně zjednodušil proces nasazení a přiblížil se uživatelské přívětivosti SIMOVE, aniž by byla obětována flexibilita otevřeného systému.

Dalším klíčovým rozdílem mezi oběma systémy je jejich přístup k modularitě a škálovatelnosti. SIMOVE je navržen jako monolitické řešení, kde jsou všechny funkce pevně integrovány do jednoho systému, což sice zajišťuje stabilitu, ale omezuje možnosti rozšíření. Například SIMOVE nepodporuje pokročilé funkce, jako je detekce palet nebo autonomní navigace bez definovaných trajektorií, a jeho integrace s externími systémy je omezená kvůli proprietární povaze softwaru. Naopak navržený systém klade důraz na modularitu, což umožňuje snadné přidávání nových funkcí a jejich vzájemnou kombinaci. Jelikož se systém poměrně rozrostl, jsou jednotlivé funkční celky zabalené v docker kontejnerech, což přináší několik výhod. Jednak je systém hezky zakonzervovaný a přehlednější, protože každý kontejner má jasně definovanou funkci a závislosti. Dále je mnohem jednodušší na spuštění pro uživatele, protože místo složitého spouštění všech uzlů v příkazové řádce stačí pouze zapnout daný kontejner pro požadovanou funkcionality. Především se však vyhýbáme problému s instalací všech potřebných balíčků, knihoven, se správným nastavením prostředí, sítí a tak dále – vše je předem nastaveno v dockingu a uživatel se o tyto technické detaily nemusí starat. Dále je možné tento kontejner spustit na libovolné platformě, takže uživatel je omezen pouze dostatečným výpočetním výkonem. Každý docker je definován pouze několika požadavky, například že požaduje data z LiDARů a enkodérů. Díky tomu stačí při změně hardwaru upravit pouze některé části, například ovladače pro nové senzory, zatímco samotná navigace a algoritmy zůstávají přenositelné a připravené k použití. Tento přístup umožňuje poskytovat software jako modulární balíček nad již existujícím hardwarem, což je výrazný kontrast k pevně vázanému ekosystému SIMOVE. Rovněž je možné jednotlivé funkcionality libovolně kombinovat.



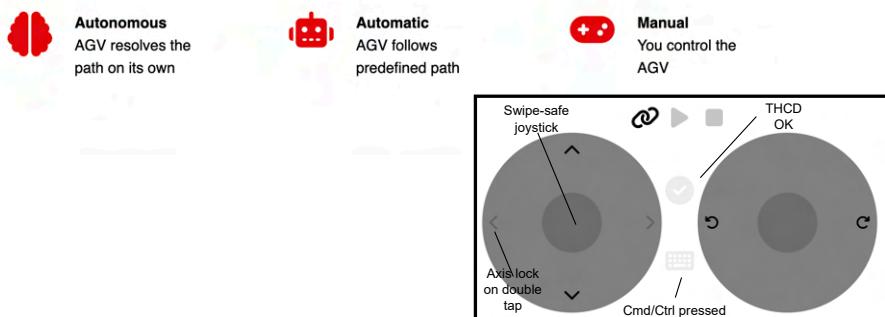
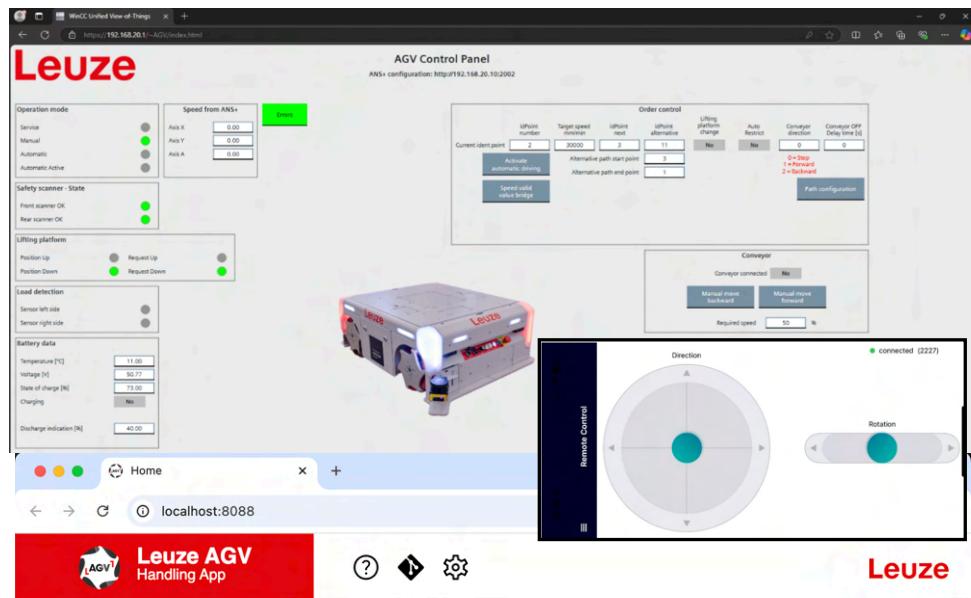
Obrázek A.1: Schéma docker kontejnerů a jejich propojení.

Schéma jednotlivých kontejnerů a jejich vzájemných závislostí je znázorněno na obrázku A.1. Kontejner *Leuze\_AGV\_Scanning* obsahuje zpracování LiDARových dat, včetně jejich sloučení z více senzorů, předání do ROS2 a filtrace šumu, a tvoří základní vrstvu pro všechny další algoritmy. Kontejner *Leuze\_AGV\_Pallet\_Handling* se zaměřuje na detekci palet z těchto dat, využívá algoritmy jako DBSCAN k identifikaci objektů a odhadu jejich polohy, a poveluje následné manévry, například zajetí pod paletu. Kontejner *Leuze\_AGV\_Navigation* zpracovává jak LiDARová, tak odometrická data a zajišťuje komplexní navigační funkce – od mapování a lokalizace přes plánování trajektorie, ohodnocení nákladové mapy až po samotnou regulaci pohybu. Oddělený kontejner *Leuze\_AGV\_Path\_Following* umožňuje sledovat cyklickou virtuální dráhu s využitím LiDARových a odometrických dat, což je užitečné pro scénáře, kde je potřeba kombinovat předvídatelnost pevných tras s flexibilitou autonomní navigace. Komunikaci mezi PLC a ROS2 zajišťuje kontejner *Leuze\_AGV\_PlC\_Chassis\_Driver*, který poskytuje informace o bezpečnostních zónách, hodnotách enkodérů a dalších parametrech, modeluje transformace jednotlivých komponent robota a vypočítává odometrii z enkodérů. Veškeré informace ze všech balíčků jsou vizualizovány pomocí kontejneru *Leuze\_AGV\_Visualization*, který prostřednictvím technologie noVNC hostuje vzdálenou plochu s aplikací RViz. Tuto plochu lze otevřít na dané adresu v prohlížeči, pokud se uživatel připojí na stejnou síť, ve které je AGV, což usnadňuje monitorování a ladění systému. Poslední komponentou je kontejner *Leuze\_AGV\_App*, který hostuje webovou aplikaci umožňující spouštění, vypínání a ovládání všech ostatních dockerů, a zároveň poskytuje možnost manuálního ovládání robota. Ihned po spuštění řídicího počítače jsou nastartovány nezbytné kontejnery, konkrétně *Leuze\_AGV\_Scanning*, *Leuze\_AGV\_PlC\_Chassis\_Driver*, *Leuze\_AGV\_Visualization* a *Leuze\_AGV\_App*. Další dockery nebo jejich části se spouštějí či vypínají výběrem různých módů v aplikaci – například v navigačním kontejneru je možné spustit pouze mapování nebo plánování trajektorie.

Dalším aspektem porovnání je uživatelské rozhraní a správa systému. SIMOVE nabízí základní rozhraní pro konfiguraci a ovládání, které je však často kritizováno za svou nepřehlednost a omezené možnosti přizpůsobení. Naproti tomu navržený systém zahrnuje webovou aplikaci (*Leuze\_AGV\_App*), která poskytuje intuitivní rozhraní pro správu všech funkcí, včetně manuálního ovládání, spouštění jednotlivých módů a monitorování stavu robota. Tato aplikace překonává SIMOVE díky své přehlednosti, rychlé odezvě a podpoře pro pokročilé funkce, jako je dynamické přepínání mezi autonomním, automatickým a manuálním režimem v reálném čase. Kromě toho technologie noVNC v kontejneru *Leuze\_AGV\_Visualization* umožňuje vzdálený přístup k RViz, který umožňuje mnohem vyšší míru interakce se systémem než obdobná vizualizace u SIMOVE.

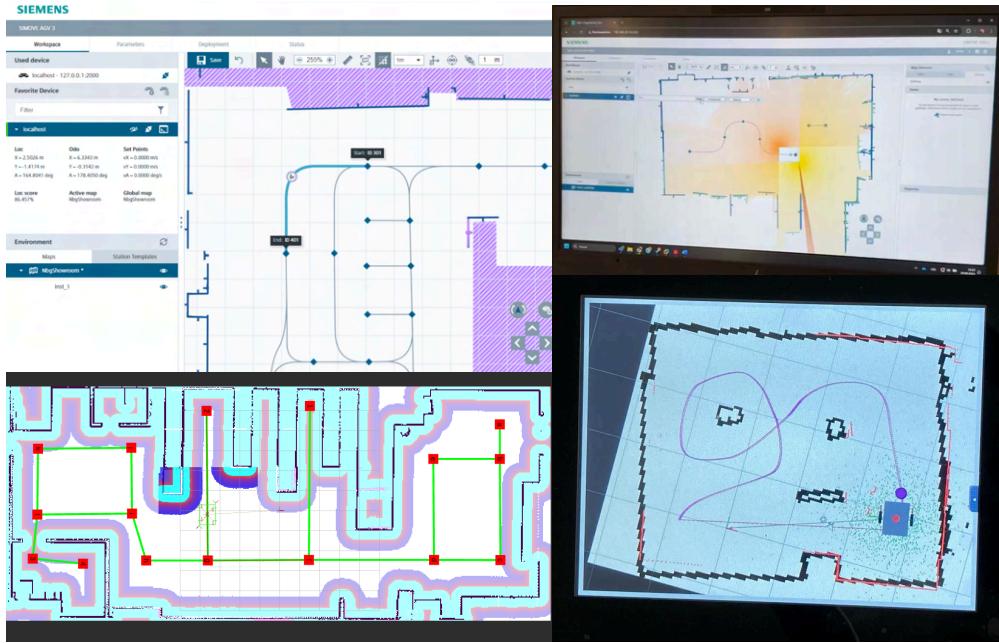
Na obrázku A.2 je znázorněna webová aplikace pro manuální ovládání a přepínání režimů, která umožňuje uživateli interagovat s navrženým řídicím systémem prostřednictvím webového serveru. V případě SIMOVE je třeba poznámenat, že webový server byl dodán integrující firmou, která zároveň vytvářela konstrukci podvozku. Kromě toho SIMOVE disponuje oficiálním webovým serverem, jehož rozhraní však obsahuje pouze rozsáhlou tabulkou parametrů určených ke konfiguraci, což odpovídá jeho zaměření na detailní nastavení systému. SIMOVE dále poskytuje vizuální prostředí, které slouží k interakci se systémem a povelování navigace, jak je patrné na obrázku A.4. Toto prostředí plní obdobnou roli jako RViz v navrženém systému, avšak s odlišným přístupem k navigaci. SIMOVE se nejvíce přibližuje navrženému řešení v případě cyklických virtuálních drah, kde lze prostřednictvím konfiguračního souboru nahrát body křivky nebo je následně upravovat přímo v mapě, což je velmi podobné možnosti kreslení dráhy do formátu SVG, jak ukazuje obrázek A.3. SIMOVE však tuto funkci kombinuje s určitou mírou autonomie tím, že uživatel může definovat pásmo kolem naplánované trasy, ve kterém se robot smí pohybovat. Přesto SIMOVE obecně přistupuje k navigaci striktněji, spíše s důrazem

na přesné dodržení plánované trajektorie a definici chování robota, než na jeho samostatné roz-hodování. Naproti tomu navržený systém je výrazně volnější a poskytuje robotovi vyšší míru autonomie, díky čemuž musí být schopen chytře reagovat na dynamické změny prostředí a přizpůsobovat své chování. Obrázek A.3 obsahuje čistě vizuální porovnání obdobných přístupů na straně navrženého systému s těmi, které poskytuje SIMOVE.

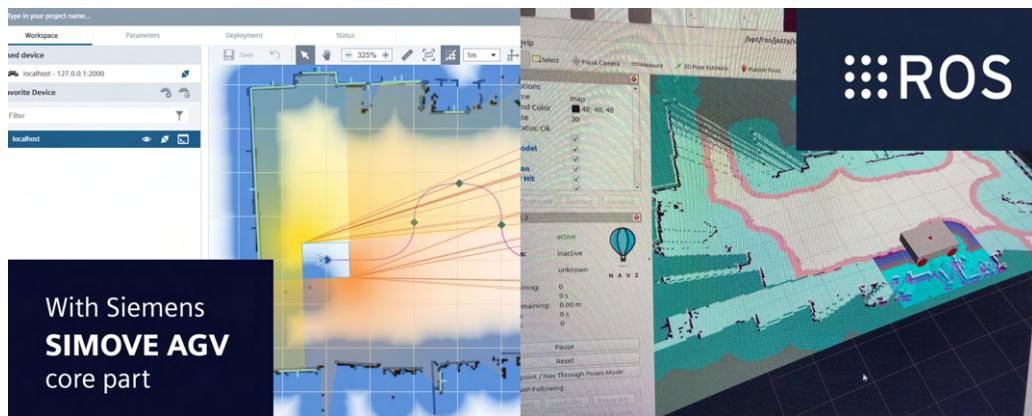


Obrázek A.2: Porovnání aplikací pro ROS2 a k dodanému řešení SIMOVE.

Celkově lze konstatovat, že navržený řídící systém představuje krok trochu jiným směrem oproti SIMOVE, zejména díky své otevřenosti, modularitě a podpoře pro pokročilé funkce. Zatímco SIMOVE je vhodný pro jasné aplikace s pevně danými požadavky, navržený systém je ideální pro dynamická a komplexní prostředí. Velký význam má z hlediska pokroku v testování a implementaci nových či inovativních funkcí, algoritmů a přístupů, což otevírá cestu k dalším vylepšením v oblasti průmyslové robotiky. Dockerizace jednotlivých funkčních celků zajišťuje snadné nasazení a přenositelnost, díky čemuž se systém stává univerzálním řešením pro širokou škálu průmyslových aplikací – například pro repasi starých robotů nebo přidávání specifických funkcí na již existující platformy bez nutnosti zásadních hardwarových úprav. Budoucí vývoj směřující k vytvoření jednotného konfiguračního systému a automatizaci kalibrace senzorů může tento systém ještě více přiblížit uživatelské přívětivosti SIMOVE, aniž by ztratil své výhody.



Obrázek A.3: Porovnání podobných přístupů v ROS2 a SIMOVE.



Obrázek A.4: Porovnání vizualizace v SIMOVE a ROS2.

Tato práce v kombinaci s přiloženou dokumentací poskytuje sice ne úplně kompletní, ale přesto obsáhlý a ucelený vhled do problematiky vývoje řídicích algoritmů pro průmyslové mobilní roboty typu AGV a AMR. Zahrnuje základní matematické podklady – od kinematických modelů přes návrh regulátorů až po plánování trajektorií – a přibližuje potřebnou senzoriku. Práce je zaměřena na framework ROS2, jehož základy jsou zde představeny tak, aby usnadnily orientaci začínajícím vývojářům. Závěrečná část práce pak nabízí náhled na možnou architekturu kompletního řídicího systému, včetně schémat provázání jednotlivých komponent, napojení klíčových částí, jako je autonomní navigace, detekce palet nebo komunikace s PLC, a přehledně vysvětluje roli nejdůležitějších prvků systému. Tím práce nejen shrnuje dosažené výsledky, ale také poskytuje pevný základ pro další výzkum a vývoj v oblasti autonomní robotiky, s potenciálem přispět k efektivnější a bezpečnější automatizaci průmyslových procesů.

# Literatura

- [1] SSI SCHAFER. Weasel Automated Vehicle | Schaefer Shelving — schaefershelving.com. <https://www.schaefershelving.com/t-warehouse-automated-guided-vehicle-weasel.aspx>. [Accessed 26-03-2025].
- [2] Linde Material Handling. C-MATIC — linde-mh.cz. <https://www.linde-mh.cz/cs/Vyrobky/Automatizovane-voziky/C-MATIC/>. [Accessed 26-03-2025].
- [3] KUKA. Mobilní plošina KMP 600-S diffDrive pro intralogistiku | KUKA AG — kuka.com. <https://www.kuka.com/cs-cz/produky,-slu%C5%BEby/amr-autonomni-mobilni-robotika/mobiln%C3%AD-plo%C5%A1iny/kmp-600-s-diffdrive>. [Accessed 26-03-2025].
- [4] Asseco-CEIT. Podbíhací AGV (Automated Guided Vehicles) | Asseco CEIT — asseco-ceit.com. <https://www.asseco-ceit.com/cz/agv-system/podbihaci-mobilni-roboty/>. [Accessed 26-03-2025].
- [5] ServisControl. RoboMec | ServisControl — serviscontrol.cz. <https://www.serviscontrol.cz/agv/robomec>. [Accessed 26-03-2025].
- [6] ABB. PHS-Flexley Tugs | AMR tugs | PHS Innovate — phsinnovate.com. <https://phsinnovate.com/automated-warehouse/autonomous-robots/phs-flexley-tug/>. [Accessed 26-03-2025].
- [7] Toyota. Automatizovaná bezvidlicová přeprava palet — toyota-forklifts.cz. <https://toyota-forklifts.cz/automatizovane-agv-voziky/automatizovana-bezvidlicova-preprava-palet/>. [Accessed 26-03-2025].
- [8] myFABER. myFABER® AMR Q3-600 - myFABER — myfaber.cz. <https://myfaber.cz/produkt/myfaber-q3-600/>. [Accessed 26-03-2025].
- [9] SIEMENS. SIMOVE ANS+ — xcelerator.siemens.com. <https://xcelerator.siemens.com/global/en/all-offerings/products/s/simove-ans.html>. [Accessed 26-03-2025].
- [10] Leuze. Rsl455p-xl-cu400p. <https://www.leuze.com/en-int/rsl455p-xl-cu400p-3m12/53800327>. [Accessed 14-11-2024].
- [11] Leuze. UM\_RSL450P\_en\_50137671. [https://files.leuze.com/Volumes/Volume0/opasdata/d100001/medias/docus/413/\\$v12/UM\\_RSL450P\\_en\\_50137671.pdf](https://files.leuze.com/Volumes/Volume0/opasdata/d100001/medias/docus/413/$v12/UM_RSL450P_en_50137671.pdf). [Accessed 25-03-2025].

- [12] Leuze. Ogs600 datasheet. [https://files.leuze.com/Volumes/Volume0/opasdata/d100001/medias/docus/200/\\$v3/UM\\_OGS600\\_en\\_50137686.pdf](https://files.leuze.com/Volumes/Volume0/opasdata/d100001/medias/docus/200/$v3/UM_OGS600_en_50137686.pdf). [Accessed 08-11-2024].
- [13] Leuze. Dcr 248i fix 11-102-r3-p. <https://www.leuze.com/en-int/dcr-248i-fix-11-102-r3-p/50146853>. [Accessed 08-11-2024].
- [14] DJI Shop oficiální e-shop DJI pro Česko. DJI RoboMaster EP Core | DJI Shop - oficiální e-shop DJI pro Česko — djishop.cz. <https://www.djishop.cz/podle-vyrobce/dji/dji-robomaster-ep-s6753.html>. [Accessed 07-11-2024].
- [15] theconstruct. A list of robots running on ROS2 - The Construct — theconstruct.ai. <https://www.theconstruct.ai/a-list-of-robots-running-on-ros2/>, 2024. [Accessed 21-11-2024].
- [16] automatizace HW. TEST - Zabezpečení strojů laserovým skenerem Leuze RSL 400. <https://automatizace.hw.cz/bezpecnost-stroju-komponenty/test-zabezpeceni-stroju-laserovym-skenerem-leuze-rsl-400.html>. [Accessed 20-03-2025].
- [17] ISO 13849-1: Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design. Technical Report ISO 13849-1, International Organization for Standardization (ISO). Available from ISO.
- [18] IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems. Technical Report IEC 61508, International Electrotechnical Commission (IEC). Available from IEC.
- [19] ISO 3691-4: Industrial trucks – Safety requirements and verification – Part 4: Driverless industrial trucks and their systems. Technical Report ISO 3691-4, International Organization for Standardization (ISO). Available from ISO.
- [20] IEC 62061: Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems. Technical Report IEC 62061, International Electrotechnical Commission (IEC). Available from IEC.
- [21] IEC 61496-3: Safety of machinery – Electro-sensitive protective equipment – Part 3: Particular requirements for Active Opto-electronic Protective Devices responsive to Diffuse Reflection (AOPDDR). Technical Report IEC 61496-3, International Electrotechnical Commission (IEC). Available from IEC.
- [22] IEC 60664-1: Insulation coordination for equipment within low-voltage systems – Part 1: Principles, requirements and tests. Technical Report IEC 60664-1, International Electrotechnical Commission (IEC). Available from IEC.
- [23] IEC 60529: Degrees of protection provided by enclosures (IP Code). Technical Report IEC 60529, International Electrotechnical Commission (IEC). Available from IEC.
- [24] Leuze. UM\_OGS600GUI\_en\_50142232. [https://files.leuze.com/Volumes/Volume0/opasdata/d100001/medias/docus/144/UM\\_OGS600GUI\\_en\\_50142232.pdf](https://files.leuze.com/Volumes/Volume0/opasdata/d100001/medias/docus/144/UM_OGS600GUI_en_50142232.pdf). [Accessed 25-03-2025].

- [25] Leuze. UM\_DCR248i\_en\_50135643. [https://files.leuze.com/Volumes/Volume0/opasdata/d100001/medias/docus/355/\\$v2/UM\\_DCR248i\\_en\\_50135643.pdf](https://files.leuze.com/Volumes/Volume0/opasdata/d100001/medias/docus/355/$v2/UM_DCR248i_en_50135643.pdf). [Accessed 24-03-2025].
- [26] DJI. RoboMaster Developer Guide documentation. <https://robomaster-dev.readthedocs.io/en/latest/>, 2014. [Accessed 07-11-2024].
- [27] Jan Holub. Development of control system and handling interface for automated guided vehicle. FEL ČVUT, 05 2025.
- [28] Sebastian Thrun. Particle Filters in Robotics. <http://robots.stanford.edu/papers/thrun.pf-in-robotics-uai02.pdf>, 2002. [Accessed 19-11-2024].
- [29] Dieter Fox. Kld-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems*, 14, 04 2002.
- [30] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. pages 343–349, 01 1999.
- [31] ROS. amcl - ROS Wiki — wiki.ros.org. <https://wiki.ros.org/amcl>, 27-08-2020. [Accessed 19-11-2024].
- [32] Syed Abdullah Fadzli, Sani Iyal Abdulkadir, Mokhairi Makhtar, and Azrul Amri Jamal. Robotic indoor path planning using dijkstra's algorithm with multi-layer dictionaries. In *2015 2nd International Conference on Information Science and Security (ICISS)*, pages 1–4, 2015.
- [33] Gazebo. physics; Gazebo documentation. <https://gazebosim.org/libs/physics/>, 2023. [Accessed 21-11-2024].
- [34] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. 09 2015.
- [35] GitHub - tvrzjak/DP\_Tvrz\_AGV: Diplomová práce na téma Vývoj autonomní mobilní platformy pro přepravu nákladů — github.com. [https://github.com/tvrzjak/DP\\_Tvrz\\_AGV](https://github.com/tvrzjak/DP_Tvrz_AGV). [Accessed 08-05-2025].
- [36] ROS 2 Documentation 2014; ROS 2 Documentation: Humble documentation — docs.ros.org. <https://docs.ros.org/en/humble/index.html>. [Accessed 05-03-2025].
- [37] Nav2 2014; nav2 1.0.0 documentation — docs.nav2.org. <https://docs.nav2.org/index.html>. [Accessed 05-03-2025].
- [38] BehaviorTree.CPP — behaviortree.dev. <https://www.behaviortree.dev/>. [Accessed 06-05-2025].
- [39] Nav2. Navigation Plugins &x2014; Nav2 1.0.0 documentation — docs.nav2.org. <https://docs.nav2.org/plugins/index.html#smoothers>. [Accessed 28-03-2025].
- [40] Steve Macenski and Ivona Jambrecic. Slam toolbox: Slam for the dynamic world. *Journal of Open Source Software*, 6(61):2783, 2021.

- [41] Jeguzzi. GitHub - jeguzzi/robomaster\_ros: ROS2 for DJI Robomaster EP and S1 — gitHub.com. [https://github.com/jeguzzi/robomaster\\_ros](https://github.com/jeguzzi/robomaster_ros). [Accessed 30-03-2025].
- [42] Snap7. Welcome to python-snap7's documentation! &x2014; python-snap7 0.0rc0 documentation — python-snap7.readthedocs.io. <https://python-snap7.readthedocs.io/en/latest/>. [Accessed 30-03-2025].
- [43] REX controls. Pidlab | První pokročilý nástroj pro analýzu a návrh Hinf regulátorů PID — pidlab.com. <https://www.pidlab.com/cs/>. [Accessed 21-03-2025].