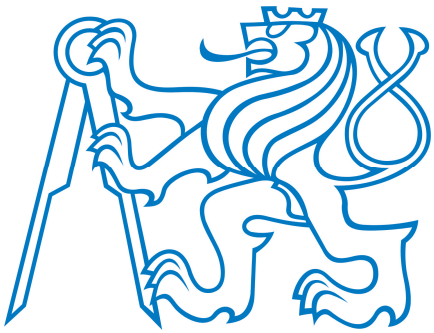
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta strojní – Ústav přístrojové a řídicí techniky



DIPLOMOVÁ PRÁCE

MODEL MALÉHO AUTONOMNÍHO ROBOTICKÉHO VOZIDLA

MODEL OF A SMALL AUTONOMOUS ROBOT CAR

Petr Herkommer **2024**

Prohlašuji, že jsem tuto práci vypracoval samostatně s použitím literárních zdrojů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů.

Datum: . . . .

Podpis

Poděkování

Tímto bych chtěl poděkovat především vedoucímu této práce, panu doc. Ing. Martinu Novákovi, Ph.D. za odbornou pomoc, nápady a vstřícnost. Poděkování patří také celé mé rodině, která mi poskytovala po celou dobu zázemí, podporu a pomoc a bez nichž by tato práce nemohla vzniknout.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Herkommer** Jméno: **Petr** Osobní číslo: **492473**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídící techniky**

Studijní program: **Automatizační a přístrojová technika**

Specializace: **Automatizace a průmyslová informatika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Model malého autonomního robotického vozidla

Název diplomové práce anglicky:

Model of a small autonomous robot car

Pokyny pro vypracování:

Navrhněte, sestavte a naprogramujte model malého autonomního robotického vozidla
1) Vytvořte univerzální platformu vozidla tak, aby bylo možné na ní použít různé senzory, např. LIDAR, kamery, radar atd.
2) Vytvořte řídicí program pro palubní počítač vozidla, který dovolí mapování okolí, jízdu do zadaného bodu a vyhnutí se překážce
3) Experimentálně otestujte a vyhodnoťte výsledky

Seznam doporučené literatury:

(1) Liu S.: Engineering Autonomous Vehicles and Robots: The DragonFly Modular-based Approach, IEEE Press, May 2020, ISBN
978-1119570561
(2) Correll, N., Hayes, B., Heckmann Ch., Roncone, A., Introduction to Autonomous Robots - Mechanisms, Sensors, Actuators, and Algorithms, The MIT Press, 2022, ISBN 9780262047555

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Martin Novák, Ph.D. odbor elektrotechniky FS

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **26.04.2024** Termín odevzdání diplomové práce: **31.05.2024**

Platnost zadání diplomové práce: _____

doc. Ing. Martin Novák, Ph.D. podpis vedoucí(ho) práce	prof. Ing. Tomáš Vyhlídal, Ph.D. podpis vedoucí(ho) ústavu/katedry	doc. Ing. Miroslav Španiel, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

Anotace

Tato práce se zabývá návrhem a realizací modelu autonomního vozidla, který je schopen vyhnout se překážkám. Vozidlo je elektricky napájeno pomocí akumulátoru, který umožňuje provoz po dobu cca 30 minut. Použitá všesměrová kola a motory umožňují vozidlu dosáhnout maximální rychlosti 18 cm/s. Hmotnost tohoto modelu je 4,5 kg a rozměry jsou 250 x 350 x 200 mm.

Model disponuje snímači používanými v reálných autonomních vozidlech, tedy radarem, 2D a 3D lidarem, hloubkovými kamerami a USB webkamerami. O řízení a vyhodnocování dat ze snímačů se stará počítač Nvidia Jetson Orin Nano. Všechny komponenty jsou umístěny na transparentní modulární platformě vlastní konstrukce. Vozidlo je schopno autonomní jízdy v předem zmapovaném prostoru. Toto řízení obstarává software ROS.

Abstract

This thesis deals with the design and implementation of an autonomous vehicle model with obstacle avoidance ability. The vehicle is electrically powered by a battery, allowing for approximately 30 minutes of operation. The use of omnidirectional wheels and motors enables the vehicle to reach a maximum speed of 18 cm/s. The weight of this model is 4.5 kg, and its dimensions are 250 x 350 x 200 mm.

The model is equipped with sensors used in real autonomous vehicles, namely radar, 2D and 3D lidar, depth cameras, and USB webcams. The control and data evaluation from the sensors are managed by a Nvidia Jetson Orin Nano computer. All components are mounted on a transparent modular platform of custom design. Vehicle is able to drive autonomously in previously mapped space. ROS software is used for this purpose.

Obsah

Seznam použitých zkratek

AI	Artificial Intelligency
AMCL	Adaptive Monte Carlo Localisation
API	Application Interface Programming
BCM	Broadcom
CAD	Computer Aided Design
dBm	deciBellmeter
DC	Direct Current
DDR	Double Data Rate
DSP	Digital Signal Processing
DWA	Dynamic Window Approach
FDM	Fused Deposition Modeling
FMCW	Frequency-Modulated Continuous Radar
FOV	Field of View
FPS	Frames per Second
GB	Gigabyte
Gbps	Gigabit per Second
GHz	GigaHertz
GNSS	Global Navigation Satellite Systém
GNU	GNU's not Unix
GPIO	General Purpose Input Output
GPS	Global Positioning System
GPU	Graphics Processing Unit
HDMI	High-Definition Media Interface
HW	Hardware
I2C	Inter-Integrated Circuit
IMU	Inertial Measurement Unit
IP	Internet Protocol

IR	Infrared
kHz	KiloHertz
LED	Light-Emmiting Diode
mAh	MiliAmpere-hour
Mbps	Megabit per Second
MEMS	Micro-Electromechanical Systems
MHz	Megahertz
OLED	Organic Light-Emmiting Diode
OS	Operating System
PC	Personal Computer
PLA	Polyactic Acid
PMMA	Poly Methyl Meta Crylate
PNG	Portable Network Graphics
PWM	Pulse-Width Modulation
px	Pixel
RAM	Random Acces Memory
RC	Radio Control
RGB	Red Green Blue
ROS	Robot Operating System
RS232	Recommended Standard 232
RViz	Robot Visualization
SBC	Single Board Computer
SDK	Software Developement Kit
SLAM	Simultaneous Localization and Mapping
SPI	Serial Peripheral Interface
SSD	Solid-State Drive
SSH	Secure Shell
SW	Software
TFT	Thin Film Transistor

TTL Transistor-Transistor Logic

UART Universal Asynchronous Receiver Transmitter
URDF Unified Robot Description File
USB Universal Serial Bus

VNC Virtual Network Computing

Wh Watt-hour
WiFi Wireless Fidelity

XML Extensible Markup Language
XOR Exclusive OR

1 Úvod

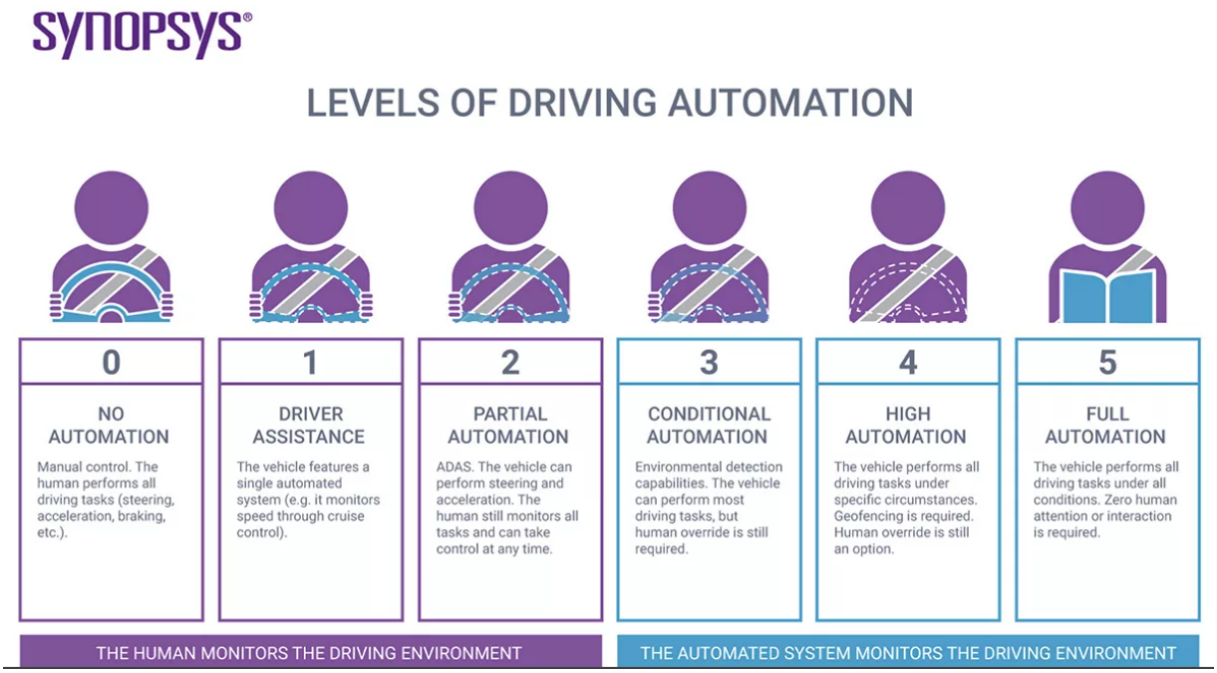
S rozvojem technologií a ve snaze ulehčit lidem práci, či ušetřit náklady na provoz, je dnes velkým trendem automatizace většiny procesů. Výjimkou není ani automobilový průmysl, kde se velmi dynamicky rozvíjí obor autonomních vozidel, tedy vozidel, která jsou dle úrovně automatizace schopna jízdy bez nutnosti zásahu řidiče.

Základem této práce byl projekt na téma senzorika autonomního vozidla. V tomto projektu byly nejprve zvoleny vhodné snímače používané i u reálných autonomních vozidel, vytipován dostatečně výkonný počítač pro řízení a poté byla navržena modulární platforma představující vozidlo, na níž spočívají všechny senzory. Tento projekt byl zpracováván v týmu dvou studentů. Druhý student vytvořil prostředí pro zobrazení výstupů jednotlivých snímačů v jazyce Python, neboť model tohoto vozidla je také využíván ve výuce předmětu technická měření. Následujícím požadavkem dle zadání je vytvoření řídicího programu pro autonomní vozidlo tak, aby bylo schopné jízdy do zadaného bodu a vyhnutí se překážce.

2 Teoretická část

2.1 Popis problematiky

Před vlastním návrhem a samotnou realizací byla provedena rešerše na téma snímače autonomních vozidel.



Obr. 1: Úrovně automatizace [1]

Dle Obr. 1 lze automatizaci v oblasti řízení vozidel rozdělit do 6 kategorií.

Kategorie 0

První kategorie představuje vozidla zcela bez automatizace - řidič obsluhuje vše.

Kategorie 1

V této kategorii jsou ve vozidle přítomny jízdní asistenty, např. adaptivní tempomat umožňující udržovat stálý odstup za jedoucím vozidlem.

Kategorie 2

Tato úroveň automatizace se nazývá částečná, přičemž vozidlo je schopno jízdy bez zásahu řidiče, tedy samo akceleruje, brzdí, mění směr jízdy. Člověk ale může v jakoukoliv chvíli převzít nad vozidlem kontrolu.

Kategorie 3

Třetí kategorie nese název podmíněná automatizace. Tato úroveň se vyznačuje schopností vozidla monitorovat okolí a autonomního řízení, ale stále je vyžadována přítomnost a pozornost řidiče.

Kategorie 4

Předposlední kategorií je stupeň vysoké automatizace, kdy vozidlo obsluhuje za určitých podmínek všechny úkony při jízdě. Vyžadováno je sledování polohy pomocí systému GPS. Stále zde ale přetrvává možnost zásahu řidiče.

Kategorie 5

Poslední kategorie představuje plnou automatizaci. Při této úrovni již není vyžadována pozornost řidiče, neboť vozidlo je schopno autonomní jízdy za všech podmínek, není tedy třeba žádných zásahů řidiče [1].

Jedním z nejdůležitějších prvků pro autonomizaci jsou bezesporu snímače monitorující okolí vozidla. Dá se říci, že konkrétní snímače mají za úkol nahradit smysly řidiče. Při jejich výběru je tedy nezbytně nutné klást velký důraz na jejich parametry. Další, neméně důležitou součástí jsou akční členy obstarávající zatáčení vozidla, jeho akceleraci a deceleraci, atd.

2.2 Existující řešení

S nástupem technologií umožňujících autonomní ovládání vozidel přicházejí na trh výukové sady v podobě robotických platforem. V této části byl proveden průzkum trhu v oblasti již existujících modelů autonomních vozidel a robotických sad blízkých dané tématice včetně jejich popisu.

2.2.1 UP Xtreme i11 & UP Squared 6000 Robotic Development Kit

Tento produkt firmy UP vznikl ve spolupráci s technologickým gigantem Intel a slouží primárně pro vzdělávací účely. Jedná se o poměrně jednoduchou mechanickou konstrukci (platformu) sestávající z podvozku tvořeného koly, ke kterému jsou následně připevněna dvě patra. Ve spodním patře je umístěn modul pro ovládání motorů, o patro výše se poté nalézá hloubková kamera společnosti Intel. Ve střední části je dále umístěna vyvýšená sekce, na níž je připevněn řídicí *PC Intel Atom x6425RE*. Napájení je realizováno dvojicí síťových adaptérů, jeden s napětím 12 V a proudem 6 A pro řídicí PC, druhý potom s parametry 12 V/5 A, který je využíván pro napájení motorů [2]. Výrobce se rozhodl pro použití tzv. mecanum všesměrových kol, která umožňují pohyb do všech směrů, tedy např. i otáčení na místě. Směr jízdy je dán poměrem rychlostí a směrem otáčení jednotlivých kol. Jediným senzorem použitým v této sadě je hloubková stereo kamera *Intel RealSense DepthCamera D435i*. Ta má zorné pole označované jako FOV 87° v horizontálním a 58° ve vertikálním směru [3]. Pro řízení motoru slouží zakázkově navržený plošný spoj s procesorem STM32f103rct6, který ovládá otevření čtyř čtveřic tranzistorů v zapojení tzv. H-můstku. Řídicí jednotkou této sady je PC UC Squared 6000 s procesorem Intel Atom x6000E, disponující 8 GB RAM, integrovanou grafickou kartu, několika USB porty, HDMI a DisplayPort konektory pro připojení zobrazovačů, nabízí také sběrnice UART, I2C, RS232. Stejně jako většina SBC má také 40pinový GPIO header umožňující připojit další příslušenství od tlačítek, přes LED diody, po již hotové moduly se sofistikovanějším komunikačním protokolem, jako je např. I2C, UART, atd. Často u SBC bývá také operační systém Ubuntu, případně alternativní distribuce GNU/Linux. Počítač navíc také podporuje systém ROS 2. Jedná se o operační systém přímo určený k řízení robotů, neboť obsahuje sadu nástrojů, ovladačů a algoritmů přímo určených pro účely robotiky [4].

Obr. 2: UP Squared 6000 Robotic Development Kit [2]

2.2.2 iRobot Create 3

Tato výuková sada vychází z robotického vysavače iRobot Roomba. Oproti originálnímu produktu však tato sada nedisponuje vysávací jednotkou, ale jsou zde zachovány původní snímače. O napájení této platformy se stará lithium-iontový akumulátor o kapacitě 26 Wh. Nabíjení probíhá v dokovací stanici. O chod se stará, stejně jako v případě předchozí zmíněné sady ROS 2, případně byl vyvinut i SDK pro jazyk Python. Další možností je potom využití aplikace výrobce s názvem *iRobot Coding App* . Komunikace má poměrně široké možnosti. První z nich je drátová komunikace přes USB kabel, neboť platforma disponuje USB typ C konektorem. Dále je možno využít technologii bluetooth, či komunikovat pomocí WiFi.

Sada obsahuje poměrně velké množství vstupních prvků, konkrétně dvě uživatelsky programovatelná tlačítka, dva nárazníky, dva kolové enkodéry pro odometrii, tedy snímání rychlosti a ujeté dráhy jednotlivých kol. Dále sada obsahuje čtyři infračervené senzory pádu, šest infračervených snímačů překážek, optický snímač ujeté vzdálenosti, jeden 3D gyroskop a akcelerometr a v poslední řadě indikátor stavu akumulátoru. Vzhledem k tomu, že se jedná primárně o sadu mobilního robota, jehož hlavním úkolem je jízda, je i výčet výstupních prvků výrazně menší než těch vstupních. Konkrétně jde o dva stejnosměrné motory obstarávající pohyb robota, dále LED pásek obsahující šest RGB diod, a jeden reproduktor [5].

Obr. 3: iRobot Create 3 [6]

2.2.3 Yahboom ROSMaster X3

Dalším zástupcem z kategorie výukových sad je produkt firmy Yahboom zabývající se výhradně robotikou. Mechanicky je tato konstrukce bližší první jmenované. Pro pohyb jsou i zde použita čtyři všesměrová mecanum kola. Pohon kol zajišťují motory vlastní výroby osazeny magnetickými enkodéry s rozlišením až 1320 pulzů na otáčku.

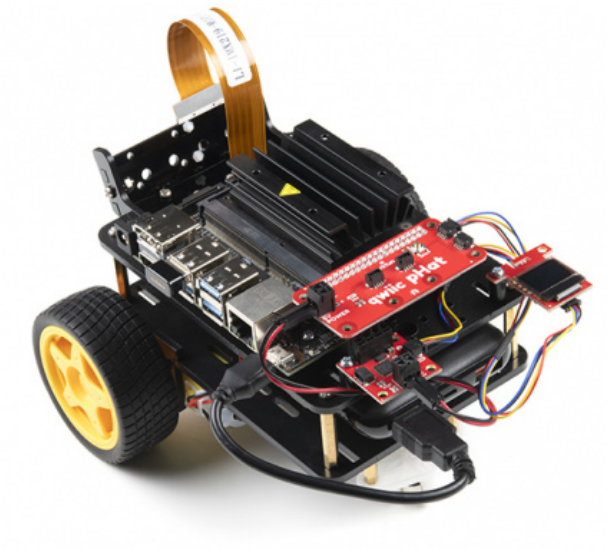
Tato sada je oproti UP Xtreme i11 již vybavenější. Nabízí hloubkovou stereo kameru ORBBEC Astra Pro Plus, navíc se zde nachází i 2D lidar RPLidar A1 firmy Slamtec. Sada obsahuje navíc i hlasový modul, který lze následně využít k hlasovému ovládání robota. Pro pohodlnější a přehlednější programování lze robota doplnit ještě dotykovým TFT displejem. Stejně jako sada od výrobce iRobot je i tento výrobek napájen z akumulátoru stejné technologie, tentokrát s kapacitou 6000 mAh, která umožňuje až hodinový provoz na jedno nabití [7]. Z hlediska řídicího PC je možnost vybrat z více možností. Základní verze nabízí Raspberry Pi 4B s paměťovým médiem v podobě USB disku. Vylepšená verze pak používá microSD kartu. Alternativou může být počítač značky Nvidia, konkrétně typy Jetson Nano a Jetson Orin Nano, opět s volitelnými konfiguracemi paměťového média včetně možnosti použití SSD disku namísto microSD karty [7]. Operačním systémem je zde opět Ubuntu Linux s možností instalace ROS 2.

Obr. 4: Yahboom ROSMaster X3 [7]

2.2.4 Nvidia JetBot AI

V tomto případně se nejedná o konkrétní produkt, ale open-source projekt vyvinutý společností Nvidia. Základ tvoří zpravidla řídicí PC Nvidia Jetson buď ve verzi Nano či Orin Nano, které se od sebe liší především výpočetním výkonem. Jak již v názvu poslední dvě písmenka napovídají, tyto počítače podporují umělou inteligenci, která je v tomto případě využívána pro rozpoznávání objektů, vyhýbání se překážkám, plánování trasy, atd. [8] Uživatel má k dispozici dokumentaci k tomuto projektu obsahující informace o obsluze jednotlivých vstupních a akčních členů, veškeré 3D modely, které si následně s pomocí 3D tiskárny může sám zhotovit.

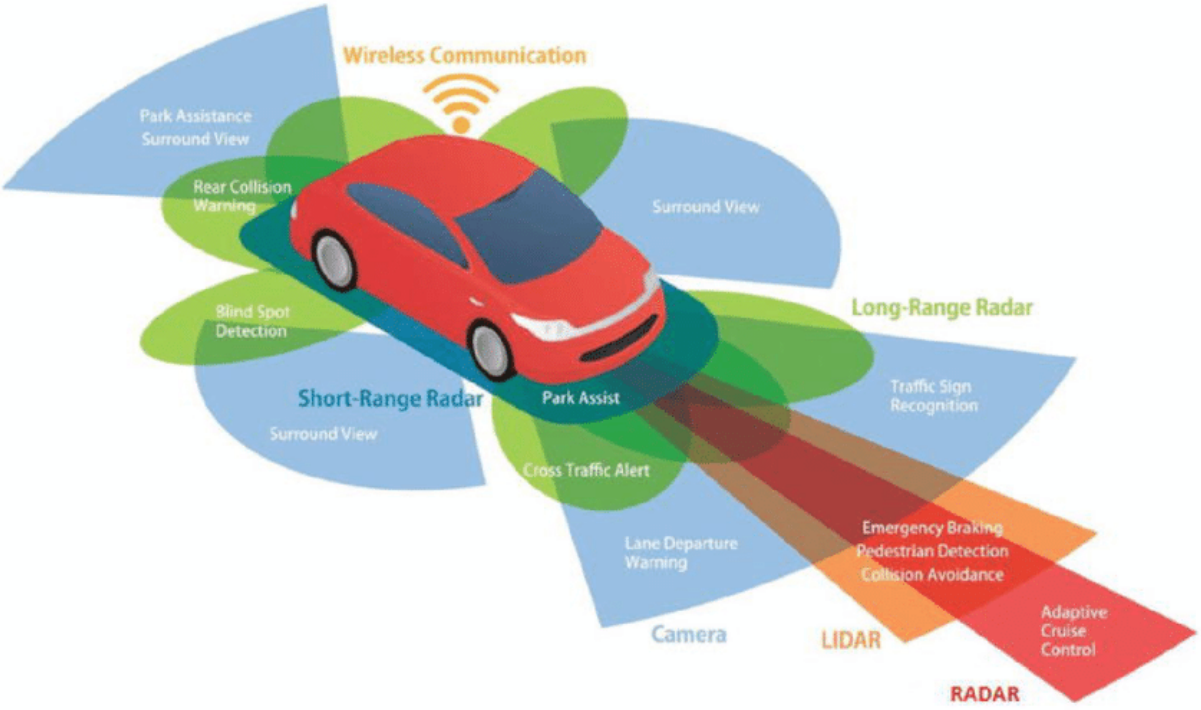
Společnost SparkFun nabízí základní sadu obsahující počítač Nvidia Jetson Nano, širokoúhlou kameru, OLED displej, řídicí desku pro motory, dvojici stejnosměrných motorů a mechanické díly pro stavbu robota. Uživatelé na platformě GitHub následně mohou sdílet své nápady, doplňovat roboty o snímače, akční členy, atd.



Obr. 5: SparkFun JetBot AI Kit [9]

2.3 Používané snímače

Na základě provedené rešerše a také ze znalosti snímačů využívaných u reálných autonomních vozidel bylo vytipováno několik snímačů, které byly použity pro tento model. Jak je patrné z Obr. 6, vozidlo využívá pro monitorování svého okolí několik kamer, dále je vybaveno radary, které využívá např. adaptivní tempomat. Pro detekci chodců a nouzové brzdění se používá lidar. Funkce, principy a využití jednotlivých snímačů jsou popsány v následujícím textu.



Obr. 6: Přehled snímačů autonomního vozidla [10]

2.3.1 Kamera

Nejjednodušším snímačem, který nahrazuje řidičův zrak je kamera. Běžně bývají autonomní vozidla osazována více než pouze jednou kamerou, aby bylo pokryto co nejširší zorné pole. Zpracováním obrazu kamer pak lze získávat cenné informace. Kamery jsou používány např. pro parkování a adaptivní tempomat. Umístěním dvou kamer vedle sebe, případně použitím stereo kamery lze zároveň získat třetí rozměr obrazu, tedy hloubku, resp. vzdálenost objektů. Výhodou kamery může být s rozvojem umělé inteligence a neuronových sítí poměrně jednoduchá klasifikace a detekce objektů. Dalším faktorem napomáhajícím poměrně masivnímu využití těchto snímačů je také jejich příznivá cena.

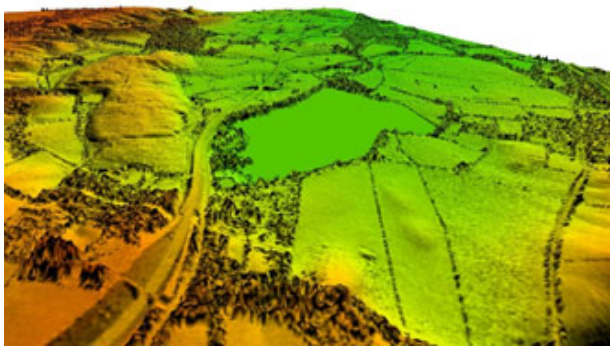
Naopak mezi nevýhody kamery patří bezesporu její citlivost na světelné a povětrnostní podmínky, neboť pro provoz v noci musí být použita buď IR kamera, případně přísvit. Problematický je také hustý déšť, či mlha.

2.3.2 Lidar

Zkratka pochází z anglického Light Detection and Ranging. Tento snímač pracuje nejčastěji na principu doby letu laserového paprsku, případně triangulace, tedy měřením úhlu mezi vyslaným a odraženým paprskem.

Jednodušší variantou lidaru je 2D čárový lidar, který umožňuje měření vzdáleností od objektů v jedné rovině. Na rotační hlavě lidaru je umístěna laserová dioda a přijímač, a ze znalosti času mezi vysláním a návratem tohoto paprsku po odrazu od překážky, a rychlosti světla dané vlnové délky lze určit vzdálenost [11]. V případě využití principu triangulace je měřen úhel mezi vyslaným a odraženým paprskem. Tento úhel je tím větší, čím větší je vzdálenost objektu od snímače. Nejčastějším výstupem z tohoto snímače je pole polárních souřadnic obsahující vždy dvojici představující úhel v rozsahu $\langle 0, 360 \rangle^o$ a vzdálenost. Z důvodu přítomnosti pohybujících se částí u 2D lidaru, tedy náchylnosti k mechanickým poruchám se používají i 3D lidary, tzv. solid-state lidars, které již nemají žádné rotující části.

Princip doby letu paprsku ovšem zůstává zachován, pouze jeden paprsek nahrazuje celý svazek paprsků, čímž je získáván 3D obraz scény. Stejně jako v případě kamer zde ale nastává problém se světelnými podmínkami, které mohou výrazně ovlivnit kvalitu a přesnost měření. Obtíže mohou způsobit např. sluneční paprsky vzhledem k podobným vlnovým délkám s laserem použitým ve snímači. 3D lidary jsou v současné době poměrně hojně používány k tvorbě 3D modelů krajin, budov, atd.



Obr. 7: Obraz z 3D lidaru [11]

2.3.3 Radar

Zkratka pochází z anglického Radio Detection and Ranging. V tomto případě je výhodou nezávislost na povětrnostních či světelných podmínkách, neboť radary využívají princip radiových vln. V autonomních vozidlech jsou používány radary pro menší vzdálenosti, operující na frekvenci 24 GHz s dosahem až 30 m, pro delší vzdálenosti až do 250 m je potom frekvence 77 GHz. Ve vozidlech jsou používány tzv. FMCW radary, tedy radary s frekvenční modulací spojitých vln. Vysílaná frekvence radiovln se v čase mění v rozsahu 77 GHz až 81 GHz [12]. Postupná změna frekvence zjednodušuje detekci odrazů od překážek.

Navíc přináší další výhody, kterými jsou vyšší přesnost, velmi nízká minimální detekovatelná vzdálenost, která je na úrovni vlnové délky radiovlny, současné měření vzdálenosti a relativní rychlosti objektu, atd. Radary mají obecně dobrou schopnost detekovat kovové předměty. Jejich nevýhodou je však obtížná, téměř až nemožná klasifikace objektů a proto se snímače zpravidla používají v kombinaci s jinými.

2.3.4 Sonar

Toto slovo je zkratkou Sound Navigation and Ranging. Jedná se o snímač spíše s nižším uplatněním. Princip je podobný radaru, ale místo radiovln se používají vlny zvukové. Nespornou výhodou tohoto snímače je, že povětrnostní či světelné podmínky nemají na měření vliv. Naopak nevýhodou je limit v podobě rychlosti zvuku, jednoduchá zarušitelnost tohoto snímače a neschopnost klasifikace objektů. Nicméně i tímto snímačem mohou být autonomní vozidla vybavena pro případ reakce na méně obvyklé situace [13].

2.3.5 Senzory zrychlení

Často označované zkratkou IMU s významem Inertial Measurement Unit jsou snímače používané k určení zrychlení - akcelerometry a natočení - gyroskopy. Tyto veličiny měří ve třech osách. Nejčastější provedení je ve společném pouzdře, kdy je technologií MEMS (Micro-Electromechanical Systems) vyrobena mechanická struktura akcelerometru společně s gyroskopem. Tyto snímače mohou pomoci s určením směru pohybu vozidla, a tyto informace následně použít ke korekci svých vstupů.

2.3.6 GNSS

Tato zkratka má význam Global Navigation Satellite Systém. Tento typ snímače sloužící k určování polohy je znám spíše pod zkratkou GPS, neboli Global Positioning System. Nicméně GPS je pouze konkrétní implementace systému GNSS. Tento druh snímače slouží k určení polohy vzhledem k Zemi. Díky přítomnosti množství satelitních družic na oběžné dráze Země a trilateraci (určení polohy bodu na základě délky tří křivek) je tento snímač schopen určit polohu vozidla ve třech souřadnicích, tedy zeměpisnou šířku, délku a nadmořskou výšku. Z důvodu možného výpadku signálu či nepřesnosti je ale tento snímač vždy redundantní.

2.4 Výběr komponent

Snahou této práce bylo vytvořit co nejvěrnější model vozidla, a to zejména z pohledu jeho senzoriky. Následující část se věnuje výběru konkrétních snímačů a řídicího PC na základě poznatků z předchozího textu. Inspirací při výběru konkrétních modelů senzorů byla také provedená rešerše, neboť lze předpokládat, že tyto snímače se již pro dané použití osvědčily.

2.4.1 Kamera

Bylo rozhodnuto, že toto vozidlo bude obsahovat několik typů kamer vzhledem k využití tohoto modelu v podobě výukové pomůcky. Konkrétní vybrané typy jsou popsány v následující části textu.

Arducam B0200

Prvním použitým typem kamery je USB webkamera **Arducam B0200**. Tato kamera byla vybrána především z důvodu, že je tzv. plug & play, tedy k její funkci není třeba žádných ovladačů v nadřazeném systému (počítači). Dalším důvodem výběru bylo široké zorné pole (100 °) , které při použití čtyř těchto kamer otočených vzájemně o 90 ° pokrývá celé okolí vozidla (360 °).

Obr. 8: USB webkamera Arducam B0200 [14]

Senzor:	Sony IMX291
Rozlišení:	2 Mpx 1945x1109 px
Snímkovací frekvence:	30 FPS
Zorné pole:	100 °

Intel RealSense D435i

Jedná o stereo hloubkovou kameru. Tato kamera má celkem tři snímače, přičemž dva z nich jsou určeny pro hloubkový obraz, třetí je RGB snímač pro vykreslení barevného obrazu. Navíc je zařízení vybaveno IMU jednotkou umožňující měřit orientaci v prostoru a zrychlení [3]. Na modelu vozidla byly použity dvě tyto hloubkové kamery.

Obr. 9: Hloubková kamera Intel RealSense D435i [3]

Rozsah:	0,3 až 3 m
Přesnost hloubkové kamery:	$\leq 2\%$ při 2 m
Rozlišení hloubkové kamery:	1280x720 px
Snímkovací frekvence hloubkové kamery:	až 90 FPS
Rozlišení RGB kamery:	1920x1080 px
Snímkovací frekvence RGB kamery:	30 FPS
Zorné pole hloubkové kamery:	87 x 58 °
Zorné pole RGB kamery:	69 x 42 °

2.4.2 Lidar

RPLidar A2

Prvním zástupcem z kategorie lidarů je 2D lidar společnosti Slamtec. Zařízení sestává z pevné a otočné části. Pevnou částí je lidar přichycen k vozidlu, rotační část obsahuje měřicí prvky, infračervenou laserovou diodu a fotodetektor. Současnou rotací, vysláním a příjmem paprsku po odrazu od objektu vzniká půdorys místnosti, ve které se lidar nachází. Pomocí sériového rozhraní lze nastavit otáčky a počet měřených bodů na otáčku. Tento lidar je určený k interiérovému použití, kdy je navíc doporučeno zařízení provozovat v prostorech bez přítomnosti slunečních paprsků, neboť může docházek k interferenci vlnových délek laseru (785 nm) a slunečního záření (400 nm až 800 nm) [15].

Obr. 10: 2D lidar RPLidar A2 [16]

Rozsah:	0,2 až 12 m
Jmenovité otáčky:	600 $ot.min^{-1}$
Jmenovitá vzorkovací frekvence:	16 kHz
Úhlové rozlišení:	0,225 ^{o}
Vlnová délka laseru:	785 nm
Přesnost:	$\leq 1\%$ z rozsahu

Intel RealSense L515

Druhým zástupcem z kategorie lidarů je 3D lidar Intel RealSense L515. Oproti předchozímu lidaru je tento robustnější, neboť nemá žádné pohyblivé části. O pohyb laserového paprsku se zde stará MEMS systém. Tento lidar je určen pro průmyslové použití např. pro roboty operujících ve skladech. Stejně jako hloubková kamera tohoto výrobce je i zde přítomen RGB snímač, takže lze barevný obraz prokládat hloubkovým dle potřeby. Princip měření opět využívá čas mezi vysláním a příjmem laserového paprsku mezi snímačem a objekty. I zde se nachází IMU jednotka pro detekci natočení a zrychlení tohoto snímače ve třech osách [17].

Obr. 11: 3D lidar IntelRealSense L515 [17]

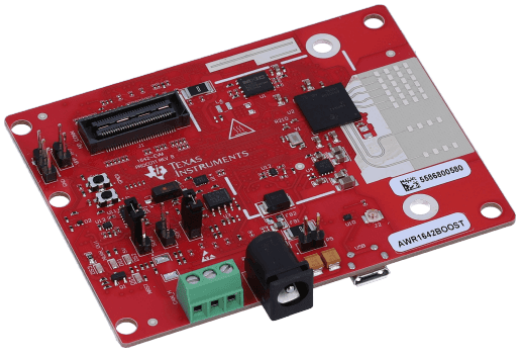
Rozsah:	0,25 až 9 m
Rozlišení hloubkového obrazu:	1024 x 768 px
Snímkovací frekvence hloubkového obrazu:	30 FPS
Přesnost hloubkového obrazu:	až 5 mm
Rozlišení RGB kamery:	1920x1080 px
Snímkovací frekvence RGB kamery:	30 FPS
Zorné pole hloubkového obrazu:	70 x 55 °
Zorné pole RGB kamery:	70 x 43 °

2.4.3 Radar

TI Radar AWR1642BOOST-ODS

Vzhledem k omezenému rozpočtu a skladové dostupnosti byl vybrán radar výrobce Texas Instruments. Jedná se o radar typu FMCW, který vysílá spojité frekvenčně modulované radiové vlny a následně vyhodnocuje jejich odrazy od objektů. Tato vývojová deska disponuje jednotkou DSP, neboli Digital Signal Processing, což výrazně usnadňuje práci, neboť o vyhodnocení změřených dat se stará přímo radar. Tato data jsou následně posílána po sériové lince do nadřazeného řídicího systému, který zároveň před začátkem měření radaru

posílá konfigurační soubor, ve kterém jsou uvedeny všechny potřebné parametry pro měření. Označení *ODS* znamená, že radar dokáže měřit ve dvou na sebe kolmých směrech [18]. Měří tedy azimut (úhel od severního pólu) a elevaci (výšku nad povrchem). Vzhledem k legislativnímu omezení maximálního vysílaného výkonu na hodnotu 12 dBm má tento radar dosah pouze 15 m.



Obr. 12: FMCW radar TI 1642BOOST-ODS [18]

Rozsah:	0,05 až 15 m
Přesnost:	40 mm
Zorný úhel azimutu:	$\pm 70°$
Zorný úhel elevace:	$\pm 40°$

2.4.4 Sonar

Vzhledem k již popsaným nevýhodám tohoto snímače, kterými jsou jeho snadná zarušitelnost, limit v podobě rychlosti zvuku, nebyl snímač tohoto typu v modelu použit.

2.4.5 Senzory zrychlení

Díky použití hloubkové kamery a 3D lidaru od společnosti Intel řady RealSense jsou akcelerometry a gyroskopy osazeny přímo v tělech těchto snímačů a tudíž nebylo potřeba používat dodatečné moduly pro měření zrychlení.

2.4.6 GNSS

Vzhledem k tomu, že se jedná o model vozidla, které je provozováno výhradně v interiéru, nebyl využit žádný GPS (GNSS) modul. Důvody tohoto rozhodnutí jsou možné problémy se stabilitou signálu v interiéru a malou relativní přesností vzhledem k velmi malým vzdálenostem, které vozidlo bude překonávat. Pro představu, přesnost evropského satelitního systému Galileo je 20 cm horizontálně a 40 cm ve vertikálním směru [19], zatímco požadovaná přesnost určení polohy modelu vozidla se pohybuje v řádech maximálně desítek mm.

2.4.7 Řídicí PC

Neméně důležitou částí byl také výběr vhodného PC pro řízení vozidla a také vyhodnocování dat z jednotlivých senzorů. Hlavní požadavky jsou následující:

- Dostatečný výpočetní výkon
- Možnost obsluhy motorů přímo z PC
- HW kompatibilita s ostatními komponenty
- SW podpora, dokumentace

Raspberry Pi 4B

Průkopníkem na poli jednodeskových počítačů je Raspberry Pi. Tento počítač byl zvažován primárně z důvodu velké podpory komunity, množství návodů a dostupnosti. Tato deska obsahuje USB porty, kterými lze připojit PC k myši a klávesnici, dále microHDMI porty pro připojení monitoru, ethernetovou zásuvku pro pevný internet. V této konfiguraci lze Raspberry Pi používat jako klasický stolní PC. Výhodu přináší přítomnost 40pinového „hřebínku“, jehož piny slouží jako GPIO, neboli vstupně-výstupní piny. Jejich pomocí lze komunikovat s vnějším světem, připojovat další moduly, na některé piny jsou vyvedeny sběrnice jako I2C, SPI, UART.

Vzhledem k obavám o nedostatečný výkon grafického procesoru tohoto počítače, byly zváženy i jiné možnosti. První alternativou bylo připojit externí grafickou kartu, nicméně vhodnější bylo vybrat jiný, výkonnější počítač. Požadavkem bylo, aby nově zvolený počítač měl podobné rozhraní jako Raspberry Pi.

Obr. 13: Raspberry Pi 4B [20]

Procesor:	4jádrový ARM v8 1.5 GHz 64 bit
Operační paměť:	8 GB DDR
Takt GPU:	500 MHz
Konektivita:	2x USB 2.0, 2x USB 3.0, Bluetooth, WiFi, Ethernet
Podporované protokoly:	I2C, UART, SPI

Nvidia Jetson Orin Nano

Trh s jednodeskovými počítači je poměrně široký, proto bylo náročné vybrat nejvhodnější PC. Z důvodu použití tohoto PC u výukové sady Yahboom X3 a vyššího výkonu oproti Raspberry Pi byl vybrán počítač od firmy Nvidia. Velká část výrobců SBC se drží standardu zavedeného Raspberry Pi v podobě 40pinového GPIO „hřebínku“, což přináší vzájemnou kompatibilitu mezi deskami různých výrobců. Tato deska navíc byla navržena s důrazem na podporu AI, má tedy výkonný procesor a GPU. Rozdíl oproti Raspberry Pi je v připojení příslušenství, neboť místo licencovaného HDMI je zde použit DisplayPort konektor.

Obr. 14: Nvidia Jetson Orin Nano [21]

Procesor:	6jádrový ARM v8.2 1.5 GHz 64 bit
Operační paměť:	8 GB DDR5
Takt GPU:	625 MHz
Konektivita:	4x USB 3.1, Bluetooth, WiFi, Ethernet
Podporované protokoly:	I2C, UART, SPI

2.4.8 Ostatní HW

Poté, co byly vybrány snímače a řídicí PC, bylo třeba zvolit zbývající komponenty, které obstarávají pohyb vozidla, apod. Inspirací pro jejich výběr se staly některé produkty z rešerše.

USB hub

Vzhledem k velkému počtu snímačů a naopak nedostatečnému počtu USB portů na řídicím PC bylo potřeba zvýšit jejich počet tak, aby mohly být připojeny všechny snímače. Celkem tento model vozidla disponuje osmi snímači. Jednodeskový počítač nabízí čtyři USB 3.1 porty, přičemž dva z toho jsou navíc použity pro myš a klávesnici. Z tohoto důvodu byl zvolen USB hub s celkem sedmi USB 2.0 porty pro připojení potřebného příslušenství

a snímačů. Výhodou tohoto rozšiřujícího zařízení je možnost jeho externího napájení pro případ většího proudového odběru, neboť vlastní USB port je možno zatížit proudem max. 1 A [22].

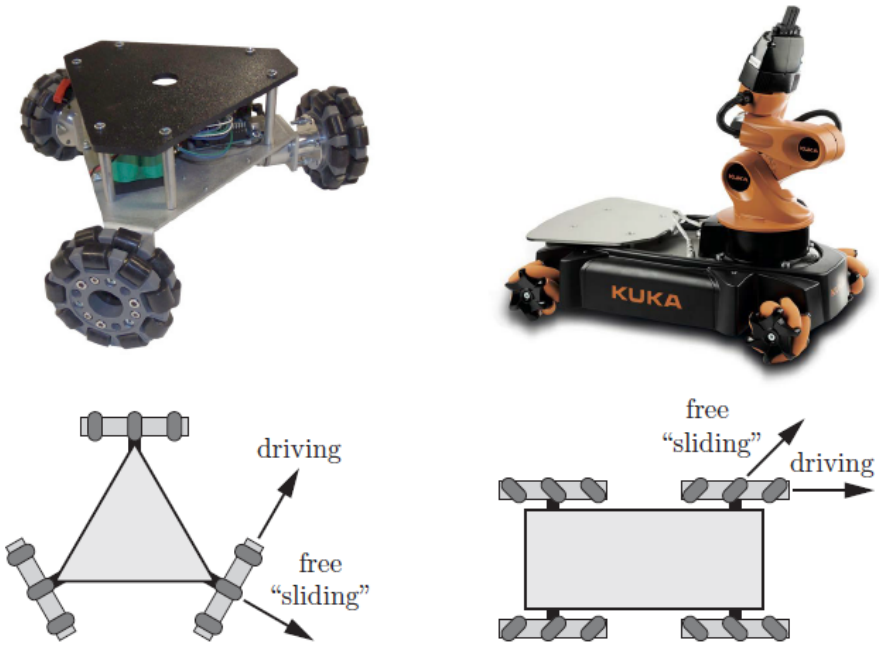
Obr. 15: USB hub se 7 porty [23]

Pohon

Dalším krokem byla volba typu pohonu. Podvozky používané v robotice se dají rozdělit na všesměrové a neholonomní [24]. Rozdílem mezi těmito dvěma typy je, že neholonomní podvozky neumožňují jízdu do boku. Důvodem je existence geometrické rychlostní podmínky. Neholonomní podvozky sestávají z kol, která jsou známa například z osobních automobilů, mají tedy jednu osu rotace kolmou na vektor rychlosti, případně je možné tato kola natáčet v ose kolmé k místě dotyku kola s podložkou. Kola se odvalují bez smýkání. Tento fakt je příčinou již zmíněné neholonomní vazby.

Naopak všesměrové podvozky lze rozdělit na ty s všesměrovými koly (omniwheels) a mecanum koly (mecanum wheels). Tyto dva typy kol mají společnou tu vlastnost, že kromě jedné osy rotace, jako tomu bylo u běžných kol, se zde nachází valivé elementy po obvodu těchto kol, které přidávají další osu rotace. Rozdíl mezi všesměrovým a mecanum kolem je v orientaci elementů na obvodu. Zatímco všesměrová kola mají tyto elementy umístěny kolmo k hlavní ose rotace, u mecanum kol je osa elementů natočena o 45 °. Přítomnost těchto elementů umožňuje pohyb podvozku i do boku, a tím je odstraněna geometrická rychlostní vazba. Odvalování těchto kol probíhá nejlépe na tvrdých hladkých površích [24].

V případě použití „klasického“ automobilového podvozku by bylo nutné řešit ještě jeho řízení (natáčení), což by nutně zvýšilo komplexnost celé konstrukce. Proto bylo rozhodnuto o použití všesměrového podvozku s mecanum koly průměru 80 mm.



Obr. 16: Robot s všesměrovými koly (vlevo) a s mecanum koly (vpravo) [24]

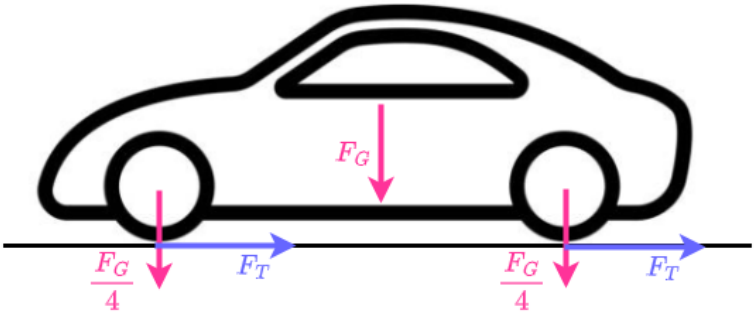
Motory

Aby bylo vozidlo schopné jízdy, bylo třeba ke kolům ještě zvolit vhodné motory, které mají za úkol obstarávat pohon modelu. Základním parametrem pro jejich výběr je jejich krouticí moment, k němuž je třeba dospět návrhovým výpočtem. Byl proto vytvořen seznam komponent s jejich hmotnostmi - viz Tab. 1, aby mohla být zjištěna přibližná celková hmotnost vozidla, a z ní navržen potřebný krouticí moment.

Tab. 1: Hmotnosti jednotlivých komponentů

Název	Hmotnost 1 ks [g]	Počet ks [-]	Celková hmotnost [g]
Arducam B0200	70	4	280
Intel DepthCamera D435i	200	2	400
RPLidar A2	200	1	200
Intel Lidar L515	100	1	100
TI AWR1642BOOST Radar	240	1	240
Nvidia Jetson Orin Nano	180	1	180
Celkem			1400

Ze znalosti celkové hmotnosti a silového rozkladu - viz Obr. 17 lze spočítat potřebný krouticí moment motorů pro jejich následný výběr.



Obr. 17: Silové působení na vozidlo

Nejprve je vhodné spočítat velikost síly $F_G = m \cdot g$, kde m je celková hmotnost vozidla a g je tíhové zrychlení.

$$F_G = m \cdot g = 1,4 \cdot 9,81 = 13,734\ N. \tag{1}$$

Ze znalosti tíhové síly a předpokladu, že tlak je rozložen mezi všechna čtyři kola rovnoměrně lze určit, že

$$F_N = \frac{F_G}{4} = \frac{13,734}{4} = 3,433\ N. \tag{2}$$

Vztah mezi normálovou silou F_N a tečnou silou F_T je dán vztahem

$$F_T = F_N \cdot f, \tag{3}$$

kde f je součinitel smykového tření. Jeho hodnota byla odhadnuta pro třecí dvojici guma - dřevo na 0,8. Po dosazení rovnice (2) do rovnice (3) dostaneme

$$F_T = \frac{F_G}{4} \cdot f = 3,433 \cdot 0,8 = 2,747\ N. \tag{4}$$

Krouticí moment pro danou sílu a poloměr kola, který je $r = 40\ mm$.

$$M_T = F_T \cdot r = 2,747 \cdot 0,04 = 0,109\ Nm. \tag{5}$$

V oblasti modelářské techniky se využívají jednotky momentu $kgcm$. Pro snazší výběr motoru byl výsledek na tyto jednotky převeden. Po převodu dostaneme

$$M_{T_{kgcm}} = \frac{M_T}{g} \cdot 100 = \frac{0,109}{9,81} \cdot 100 = 1,12\ kgcm. \tag{6}$$

S ohledem na výsledný moment, požadovanou rychlost cca 20 cm/s a výkonovou rezervu byly vybrány čtyři stejnosměrné kartáčové motory JGA25-370 s ocelovou převodovkou s otáčkami 60 min^{-1} a krouticím momentem $1,8$ $kgcm$. Ze znalosti použitých kol a motorů bylo možné určit maximální rychlost, kterou vozidlo vyvine při jízdě. Je třeba zmínit, že skutečná rychlost bude vůči vypočtené nižší, neboť motory budou zatíženy krouticím momentem, což způsobí pokles otáček. Rychlost vozidla lze spočítat jako

$$v = \pi \cdot d \cdot n = \pi \cdot 80 \cdot 60 = 15079\ mm/min. = 0,251\ m/s. \tag{7}$$

Redukční poměr	4. 4	9. 6	21. 3	35	45	78	103	171	226	377
Proud bez zatížení (mA)	60	60	60	60	60	60	60	60	60	60
Otačky bez zatížení (RPM)	1360	620	280	170	130	77	60	35	26	16
Jmenovitý točivý moment kg*cm	0. 1	0. 22	0. 5	0. 8	1	1. 8	2. 4	4	5. 2	8. 4
Jmenovité otáčky (RPM)	1000	450	220	130	100	60	46	27	20	12
Jmenovitý proud (A)	0. 45	0. 45	0. 45	0. 45	0. 45	0. 45	0. 45	0. 45	0. 45	0. 45
Maximální točivý moment (kg*cm)	0. 35	0. 75	1. 7	2. 8	3. 36	6. 2	8. 2	9	Maximální zatížení	
Stop proud (A)	1. 3	1. 3	1. 3	1. 3	1. 3	1. 3	1. 3	1. 3	Nepodporuje st	
Délka	17		19	21		23		25		27

Obr. 18: Přehled stejnosměrných motorů [25]

Obr. 19: Stejnosměrný motor JGA25 [25]

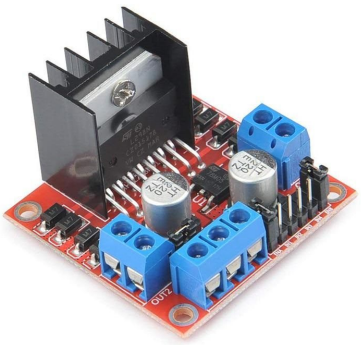
Enkodéry

Jsou zařízení pro snímání otáček, případně úhlového natočení hřídele motoru. Aby mohla být vytvořena zpětná vazba o natočení všech kol, resp. ujeté dráze, bylo třeba motory osadit těmito snímači. Způsobů snímání je celá řada, od mechanických, přes magnetické až po infračervené. Nejjednodušší variantou je měření přímo na hřídeli motoru, nicméně to by bylo vzhledem ke konstrukci motoru komplikované řešení. V této aplikaci není nezbytně nutné rozlišit směr otáčení kol, tedy odpadá nutnost mít dva vzájemně fázově posunuté pulzy, které jsou známy ze standardních rotačních enkodérů. Proto bylo rozhodnuto o použití infračervených snímačů přiblížení, které budou snímat jednotlivé pulzy vytvořené rozdílnou reflexivitou snímaného povrchu.

Řízení motorů

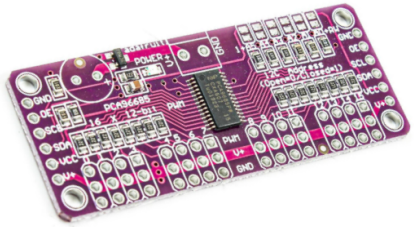
Vzhledem k použití mecanum kol bylo nezbytně nutné mít možnost řídit otáčky každého kola zvlášť, neboť výsledný vektor rychlosti vozidla je dán otáčkami jednotlivých kol. Změna směru otáčení u stejnosměrných motorů není složitá, neboť stačí změnit polaritu napájecího napětí [26]. Aby tento úkon nevyžadoval neustálé přepojování vodičů, bylo použito zapojení v tzv. H-můstku. Jedná se o zapojení se čtyřmi spínači (nejčastěji tranzistory) umožňující změnu polarity. Další částí regulace stejnosměrných motorů je i řízení velikosti jejich otáček. I tohoto lze docílit poměrně snadno změnou velikosti napájecího napětí. Tato změna

je realizována pomocí rychlého spínání a vypínání tranzistorů, které se projeví snížením efektivní hodnoty napětí. Tomuto řízení se říká PWM (Pulse-Width Modulation). Pro tento účel byl vybrán modul s integrovaným obvodem L298, který umožňuje řízení dvou stejnosměrných motorů včetně regulace směru a velikosti otáček.



Obr. 20: H-můstek L298 [27]

Řídicí PC Nvidia má čtyři piny, na kterých je umožněno generovat PWM signál, nicméně tyto piny jsou tzv. multiplexované, takže ve skutečnosti jsou k dispozici pouze dva PWM kanály. Proto muselo být přistoupeno k použití speciálního modulu. Pro tyto účely byl zvolen PWM expander s integrovaným obvodem PCA9685, který má 16 PWM kanálů s 12bitovým rozlišením. Komunikace s nadřazeným PC probíhá pomocí protokolu I2C [28].



Obr. 21: PWM expander PCA9685 [28]

2.5 Konstrukce

V době, kdy byly známy všechny komponenty, bylo možno přistoupit ke konstrukci vozidla. Vzhledem k tomu, že model má sloužit i pro edukativní účely, bylo rozhodnuto, že materiál, z něhož bude vozidlo postaveno, bude transparentní PMMA (plexisklo) tak, aby byly patrné všechny části konstrukce. Dále bylo požadavkem, aby byla konstrukce co nejvíce modulární v případě, že by byl model do budoucna rozšiřován. Je tedy žádoucí používat časté opakování stejných prvků. Podpůrné části konstrukce byly vyrobeny na 3D FDM tiskárně z materiálu PLA, neboť je tato technologie v daných podmínkách nejdostupnější. Platforma bude rozdělena do několika celků dle účelu a prostoru, tedy model bude víceúrovňový.

Nepsaným pravidlem je vést vodiče silové a měřicí části co nejdále od sebe, čímž lze předejít problémům s rušením signálů ze senzorů. Proto byla i platforma takto koncipována.

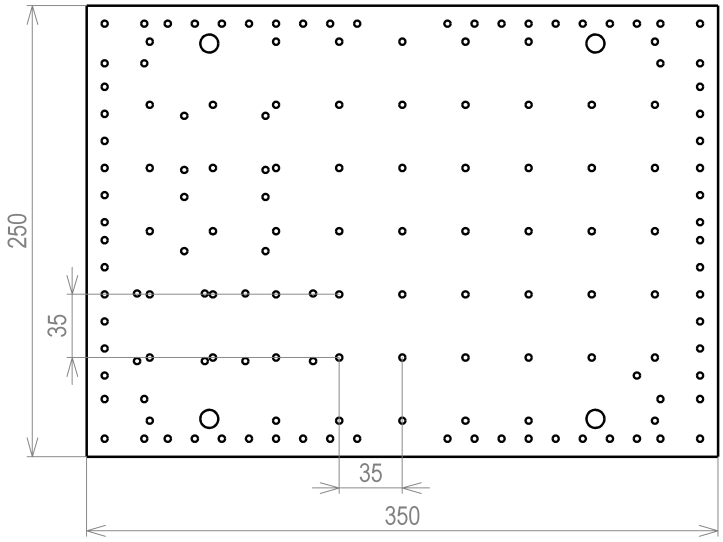
Ve spodní úrovni jsou umístěny komponenty obstarávající napájení PC, motorů, a zbytku snímačů. Ve druhé úrovni jsou poté umístěny všechny snímače a řídicí PC. Třetí úroveň slouží jako kryt snímačů, a zároveň je na ní upevněn lidar. V úrovni lidaru nejsou žádné další prvky, které by mohly blokovat či ovlivňovat jeho zorné pole.

2.5.1 Základní modul

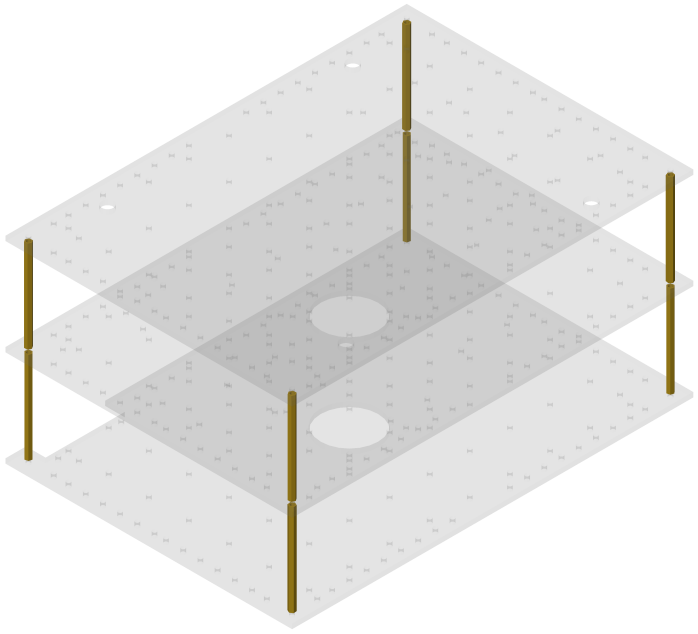
Vzhledem k požadavku na maximální modulárnost konstrukce byl v několika iteracích navržen základní modul (deska), jehož konečné rozměry mají hodnotu 250 x 350 mm. Dále musela být určena vhodná tloušťka materiálu, z něhož budou jednotlivé desky vozidla vyrobeny. Protichůdnými požadavky byly na jedné straně snaha o co nejnižší hmotnost, na straně druhé co nejvyšší tuhost. Nakonec byla zvolena tloušťka materiálu 4 mm. Pro případ jednoduché rozšířitelnosti je každá deska vyplněna čtvercovým rastrem děr s roztečí 35 mm pro spojovací materiál velikosti M3. Pro optimální využití místa jsou v deskách umístěny otvory i mimo 35mm rastr, připravené pro montáž snímačů, apod. Středem vrchních dvou desek vedou dva otvory pro jednoduché vedení kabeláže.

2.5.2 Modulárnost

Z důvodu snadné rozšířitelnosti byl navržen jednoduchý systém modulárnosti, kdy je na sebe možno umístit teoreticky „nekonečný“ počet desek. Pro spojování jednotlivých desek byly zvoleny distanční sloupky M3 s délkou 80 mm. Tyto sloupky jsou na jedné straně opatřeny vnějším, na druhé vnitřním závitem, aby je bylo možné řadit sériově. Výhodou je dostupnost tohoto spojovacího materiálu, a volba v podstatě libovolné výšky každé úrovně. Zde byla volena nejvyšší dostupná délka z důvodu prostornosti konstrukce.



Obr. 22: Základní modul

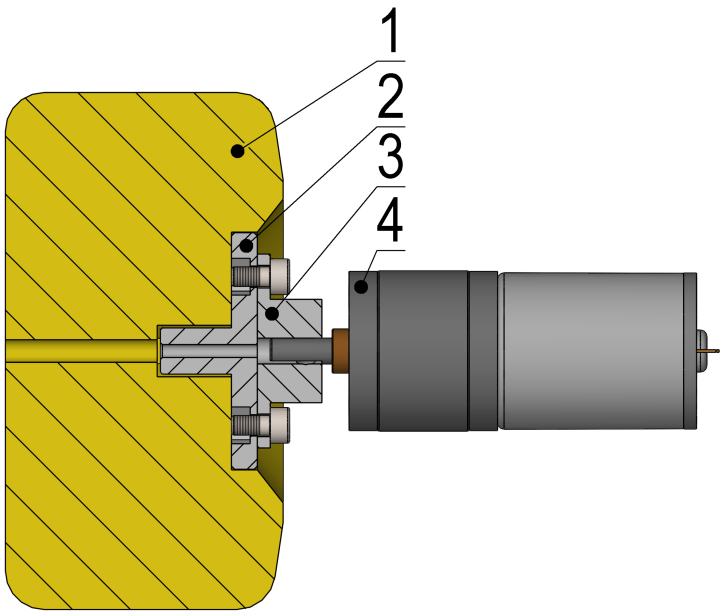


Obr. 23: Princip modulárnosti konstrukce

2.5.3 Podvozek

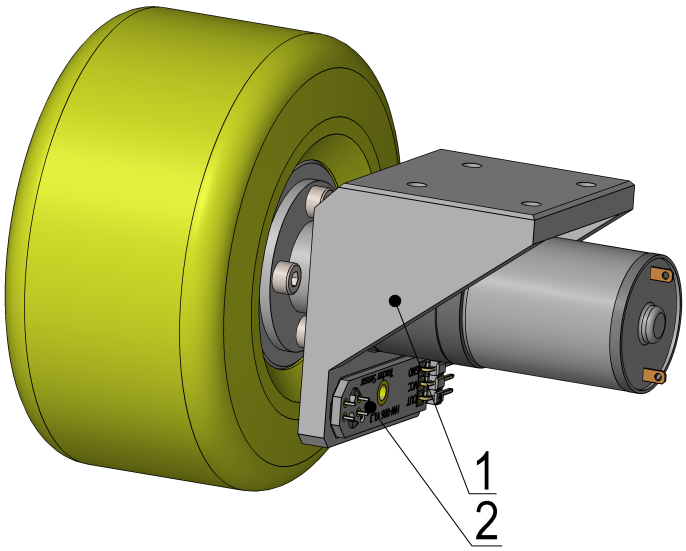
Dalším úkolem bylo vyřešit spojení kol s motory a následně uchycení této sestavy k rámu vozidla. Jelikož ke kolům byly dodány pouze adaptéry na jiný než v této konstrukci použitý motor, musela být navržena vlastní konstrukce uchycení kol k hřídelům. Pozice 2 na Obr. 24 představuje 3D tištěný adaptér, který je proti vypadnutí z kola zajištěný šroubem do plastu z vnější strany kola. Na pozici 3 se poté nachází vyrobený duralový adaptér spojující 3D tištěný adaptér s hřídelí motoru. Tyto dva adaptéry jsou vzájemně spojeny čtveřicí šroubů M3. V plastovém adaptéru jsou zhotovena tvarová vybrání pro matice, neboť závity v

tištěných dílech nevykazují dostatečnou pevnost a spolehlivost spoje. Pro zajištění axiální i radiální polohy tohoto adaptéru vůči hřídeli motoru slouží stavěcí šroub M3.



Obr. 24: Spojení kola a motoru

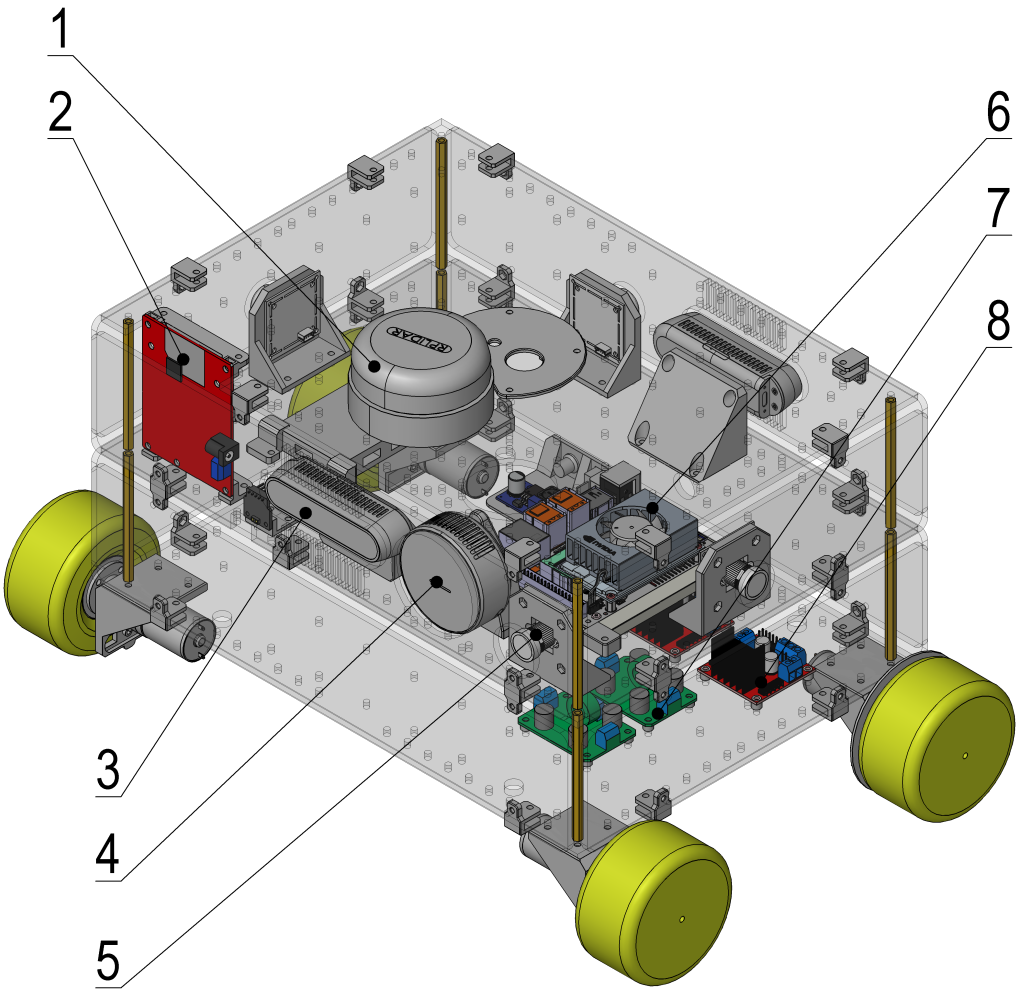
Na Obr. 25 je patrná celá sestava kola i s držákem motoru (1) a infračerveného snímače (2), který je umístěn proti čelní ploše kola a bude sloužit pro snímání otáček jednotlivých kol. Na zmíněné čelní ploše kol budou muset být vyhotoveny opticky dostatečně kontrastní plošky tak, aby je snímač mohl zaznamenávat.



Obr. 25: Kompletní kolo s držákem a snímačem

2.5.4 Celková sestava modelu

Na Obr. 26 je patrná celková sestava vozidla. Již dříve bylo uvedeno, že na spodní desce jsou umístěny výkonové součástky, konkrétně na pozici 7 dva měniče napětí XL6009 pro provoz na akumulátory. Dále se zde nachází modul s rozšířením PWM kanálů PCA9685 (v zákrytu), své místo zde mají i dva H-můstky pro řízení motorů (8). Ve spodní desce jsou připraveny otvory pro pohodlné protažení kabeláže od motorů. Čelní stranu vyššího patra potom zabírají zprava radar (2), hloubková kamera (3), 3D lidar (4) a webkamera (5). Po obou stranách vozidla jsou umístěny webkamery. Na zadní stěně se poté nalézají vedle sebe hloubková kamera a webkamera. Na třetí úrovni modelu vozidla se nalézá lidar. Všechny snímače jsou pak upevněny speciálně zkonstruovanými držáky k deskám. Pro návrhy uchycení jednotlivých snímačů byly s výhodou využity 3D modely poskytované jejich výrobci.



Obr. 26: Sestava vozidla

3 Praktická část

Ve chvíli, kdy byly vybrány a nakoupeny všechny potřebné díly a komponenty, mohlo být přistoupeno k samotné výrobě modelu vozidla. Transparentní desky byly zhotoveny technologií laserového pálení, což velmi usnadnilo a zrychlilo celý proces. Ostatní díly byly vytištěny na 3D tiskárně z materiálu PLA a spojovací materiál nakoupen. Použity jsou výhradně šrouby, matice a podložky velikosti M3.

Obr. 27: Praktická realizace modelu vozidla

3.1 Řídicí PC

Prvním krokem v praktické části bylo zprovoznění výpočetní jednotky celého stroje. Tím byl počítač Nvidia Jetson Orin Nano. Nejprve bylo nutné nainstalovat do slotu zespodu počítače SSD disk, který slouží jako úložiště pro všechny soubory včetně operačního systému. K vytvoření instalačního média je zapotřebí speciálního nástroje Nvidia SDK Manager, který je kompatibilní pouze s OS Ubuntu Linux [29]. Z tohoto důvodu musel být na osobní PC nainstalován jako druhý systém Ubuntu Linux ve verzi 18.04, na nějž mohl být následně nahrán instalační nástroj. Deska následně byla zkratováním dvou pinů uvedena do tzv. recovery módu a systémové soubory se nahrály na SSD disk. Po několika minutách se systém spustil a byl připraven k použití.

3.2 Snímače

Dalším úkolem bylo zprovoznit, otestovat a ověřit funkčnost a kompatibilitu jednotlivých senzorů s řídicím PC. Výhodou je, že všechny senzory se připojují USB konektorem. Jejich instalace je proto velmi snadná. Je důležité zmínit, že všechny snímače nebyly využity pro mapování okolního prostředí a následnou autonomní jízdu. Jsou ale využívány v případě využití tohoto modelu vozidla jako výukové pomůcky.

3.2.1 Intel RealSense

Snímače řady RealSense výrobce Intel byly zapojeny do USB hubu. Jejich funkčnost byla následně testována na jednoduchých ukázkových příkladech napsaných v jazyce Python, které jsou k dispozici v rámci SDK výrobce. Se spuštěním příkladů ovšem nastaly problémy. Jedním z problémů bylo nedostatečné rozlišení hloubkové stero kamery DepthCamera D435i, které bylo pouze 320x240 px, zatímco dle výrobce mělo být dosažitelné rozlišení až dvojnásobné, tedy 640x480 px. Nicméně u 3D lidaru L515 nebyl problém dosáhnout vyššího rozlišení. Následně bylo zjištěno, že obě kamery D435i mají zastaralý firmware, který nepodporoval toto vyšší rozlišení. Proto byl firmware aktualizován za použití programu Intel RealSense Viewer, který pro změnu podporuje pouze OS Windows.

Náhled v programu RealSense Viewer již vykazoval požadované rozlišení hloubkového obrazu, proto mohly být snímače nainstalovány zpět na počítač Nvidia. Po opětovné instalaci se ale ukázalo, že jednoduché programy stále nejsou funkční a vyšší rozlišení není dostupné. Následně bylo pokusem zjištěno, že pokud se kamera zapojí přímo do USB portu na řídicím PC, vyšší rozlišení funguje bez problémů.

Vzhledem k tomu, že datový kabel dodaný ke kameře byl specifikace USB 3.0, USB porty na desce jsou ve verzi 3.1, bylo zřejmé, že nejslabším článkem v této kaskádě je USB hub, jež má 7x USB 2.0. O tom, že rozdíl mezi těmito dvěma generacemi je více než značný, svědčí maximální přenosové rychlosti u USB 2.0 480 Mbps, zatímco USB 3.0 nabízí až 5 Gbps. Jde tedy o více než desetinásobný nárůst rychlosti [30]. Z tohoto důvodu byl zakoupen nový hub, tentokrát se specifikací USB 3.0. Po jeho instalaci již bylo dosaženo rozlišení 640x480 px.

3.2.2 Radar

Zapojení radaru bylo velmi jednoduché. Stačilo propojit USB kabelem vývojovou desku TI AWR1642BOOST-ODS s USB hubem a dále připojit radaru externí napájení pomocí síťového adaptéru s parametry 5 V/3 A.

Jak bylo již uvedeno v části 2.4.3, aby mohl radar měřit, je třeba nejprve odeslat po sériové lince správnou konfiguraci, např. kolik deska obsahuje přijímacích a vysílacích antén, jaká jsou jejich zorná pole, požadované měřicí frekvence, atd. K tomuto účelu slouží

webové prostředí TI MMWave Demo Visualizer, které umožňuje vytvořit konfigurační soubor a rovnou otestovat jeho funkčnost na několika jednoduchých grafech. Při testování se ukázalo, že pravidelně po krátkém čase dochází k výpadkům komunikace. Obnovení měření vždy vyžadovalo fyzické odpojení radaru od počítače. Na základě doporučení výrobce byl upgradován firmware z verze SDK 2.0 na SDK 3.0 pomocí nástroje UniFlash. Po této aktualizaci již k výpadkům nedocházelo. Výrobce poskytl jednoduché testovací programy pro jazyk Python, které v tomto případě fungovaly dle očekávání.

3.2.3 Lidar

Posledním snímačem, který bylo třeba zprovoznit, byl RPLidar A2. Tento proces, stejně jako všechny ostatní začal zapojením USB konektoru do USB hubu. Výrobce poskytuje k tomuto snímači jednoduchý demo program, který vizualizuje naměřená data v polárních souřadnicích. Nicméně tento program je určen pouze pro OS Windows, kde byla předtím testována jeho funkčnost.

Na platformě GitHub byly nalezeny ukázkové kódy pro jazyk Python, které měly být schopny obsluhovat lidar a vizualizovat data nezávisle na použité platformě. Python program byl poměrně komplexní, neboť obstarával konkrétní příkazy pro ovládání motoru lidaru, odeslání konfigurace, atd. Po krátkém čase se však ukázalo, že dochází k problémům s rychlostí zpracování příchozích dat. Důvodem byla vysoká rychlost komunikace mezi snímačem a PC, tedy velké množství dat.

Vzhledem k tomu, že Python je jazyk **interpretovaný**, u nějž je obecně běh programu pomalejší, bylo rozhodnuto o použití **kompilovaného** jazyka C++ [31]. Rozdíl mezi těmito dvěma přístupy je, že zatímco u **interpretovaného** jazyka jsou instrukce vykonávány přímo za běhu programu, v případě **kompilovaného** jazyka dojde nejprve k překladu do strojového kódu, který je následně vykonáván.

Následně byla využita knihovna *FastestRplidar*, která představovala vhodné řešení pro daný problém. Jedná se o Python wrapper programu psaného v jazyce C++ poskytnutého výrobcem, tedy využívání funkcí jazyka C++ v prostředí Python. Toto řešení se ukázalo jako vyhovující.

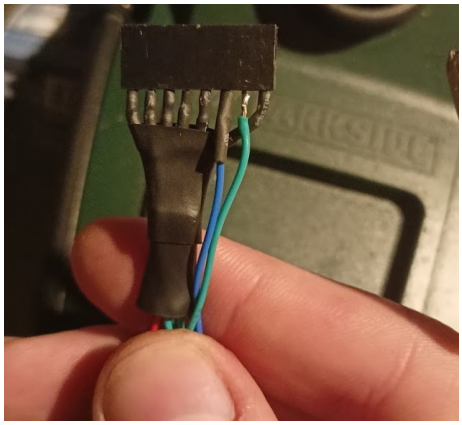
3.3 Vstupně-výstupní prvky

Po zapojení a otestování všech snímačů mohlo být přikročeno k zapojení a zprovoznění částí týkajících se pohybu vozidla. Konkrétně se jedná o zapojení motorů, jejich řízení a vyřešení snímání otáček kol.

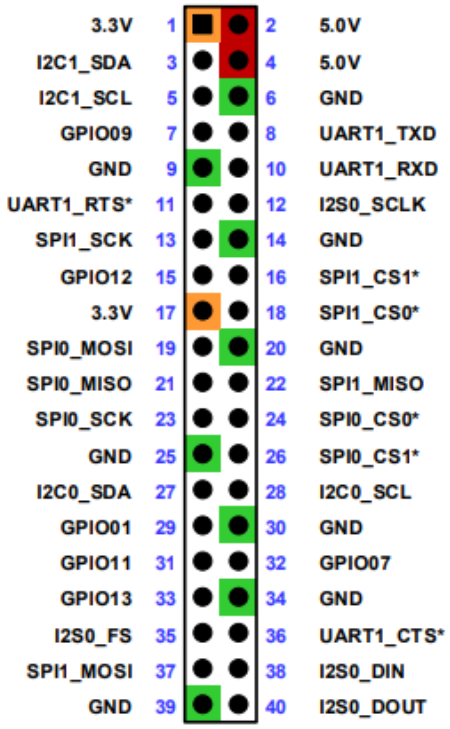
3.3.1 40 pin GPIO header

Všechny zmíněné prvky byly zapojeny do GPIO konektoru, který má stejné rozložení s PC Rasberry Pi 4B. Byl použit 2x20pinový jednořadý konektor, aby byla usnadněna práce při připojování všech komponent. Vzhledem k nespolehlivosti elektrického kontaktu propojovacích vodičů opatřených dupont konektory, které jsou na vodiče nalisovány bylo rozhodnuto o výrobě kabelových svazků na míru - viz Obr. 28, kde je elektrický kontakt vodiče s konektorem zajištěn pájením olovnatým cínem.

Z Obr. 29 je patrné rozložení a použití jednotlivých pinů na GPIO headeru. Za pomocí tohoto obrázku byl vytvořen návrh zapojení jednotlivých komponent, který je v tabulce Tab. 2.



Obr. 28: Výroba konektorů kabelových svazků



Obr. 29: Rozložení pinů na desce Nvidia Jetson Orin Nano [32]

Tab. 2: Schéma zapojení jednotlivých komponentů

Použití	Číslo pinu		Použití
Napájení - 3,3 V	1	2	-
PCA9685 - I2C_SDA	3	4	-
PCA9685 - I2C_SCL	5	6	-
IR senzor 1	7	8	-
Napájení - GND	9	10	-
IR senzor 2	11	12	-
IR senzor 3	13	14	-
IR senzor 4	15	16	-
-	17	18	-
H-můstek 1 - IN1	19	20	-
H-můstek 1 - IN2	21	22	-
H-můstek 1 - IN3	23	24	-
-	25	26	-
-	27	28	-
H-můstek 1 - IN4	29	30	-
H-můstek 2 - IN1	31	32	-
H-můstek 2 - IN2	33	34	-
H-můstek 2 - IN3	35	36	-
H-můstek 2 - IN4	37	38	-
-	39	40	-

3.3.2 H-můstky

Tyto prvky obstarávají ovládání směru a rychlosti otáčení jednotlivých motorů. Modul vyžaduje napájení logické části 5 V. Vzhledem k přítomnosti lineárního stabilizátoru LM7805 na tomto modulu bylo použito společné napájení motorů a logiky. Dále se zde nacházejí dvě svorkovnice pro připojení motorů. Ovládání výstupu je pak realizováno pomocí logické funkce XOR v podobě TTL logiky. Hodnotu logické 1 či 0 reprezentuje napětí 3,3 V, resp. 0 V generované počítačem Nvidia Jetston Orin Nano.

Jsou-li oba vstupy na hodnotě logické 0 nebo 1, motory stojí. V závislosti na kombinaci logické 1 a 0 dochází pak k otáčení motoru požadovaným směrem, toto chování popisuje Tab. 3. Posledním vstupem pro každý kanál je potom svorka označená „ENA“, resp. „ENB“, díky níž lze řídit otáčky motorů. Standardně je tato svorka zkratována propojkou s potenciálem 5 V, takže motory se otáčejí maximálními otáčkami. V tomto případě ale byla propojka vyjmuta, a svorky připojeny k PWM modulu PCA9685.

Tab. 3: Pravdivostní tabulka logické funkce XOR

IN1 (IN3)	IN2 (IN4)	Motor
0	0	Stojí
0	1	Otáčení doprava
1	0	Otáčení doleva
1	1	Stojí

3.3.3 PWM expander PCA9685

V kapitole 3.3.2 již bylo zmíněno připojení svorek „ENA“, resp. „ENB“ k PWM výstupům tohoto modulu. Celkem jsou použity čtyři výstupy, každý pro jeden motor. Zapojení tohoto modulu je realizováno přivedením napájecího napětí z pinů 1 a 9 GPIO (viz Tab. 2) headeru řídicího PC a připojením vodičů pro I2C komunikaci.

3.3.4 Měření otáček a natočení kol

Aby bylo možné vyhodnocovat natočení kol, bylo třeba zhotovit dvě kontrastní pole, která vytvářejí elektrické pulzy odpovídající natočení kol.

Infračervený snímač

Dalším krokem byla realizace zapojení IR snímačů. Jejich zapojení bylo velmi jednoduché, stačilo přivést napájení. To bylo realizováno stejně jako u PWM expanderu.

Dále byly připojeny výstupy snímačů na piny řídicího PC 7, 11, 13 a 15 (viz Tab. 2). IR modul obsahuje integrovaný obvod se Schmittovým klopným obvodem, který obstarává vyhodnocení měření. Zároveň zlepšuje robustnost, neboť vykazuje hysterezi, takže nedochází k velmi rychlému cyklickému spínání a vypínání výstupu.

Snímač má na svém výstupu logické hodnoty v podobě potenciálů napájecího napětí, kdy 0 V odpovídá logické 0, napájecí napětí 3,3 V potom reprezentuje přítomnost objektu, tedy logickou 1.

Opatření kol reflexními prvky

Potřebný počet pulzů na jednu otáčku byl určen z požadavku na minimální lineární vzdálenost mezi dvěma pulzy. Ta byla stanovena na 10 mm. Z tohoto požadavku mohl být poté proveden výpočet potřebného počtu pulzů. Pro rozvinutou délku oblouku platí vztah

$$s = \varphi \cdot r, \tag{8}$$

kde s je délka oblouku, φ reprezentuje středový úhel tohoto oblouku a r je poloměr kola.

$$\varphi = \frac{s}{r} = \frac{10}{40} = 0,25\ rad = 14,32\,^{\circ}. \tag{9}$$

Známe-li tento úhel, není již složité určit potřebný počet pulzů:

$$p = \frac{360}{\varphi} = \frac{360}{14,32} = 25,13. \tag{10}$$

Minimální potřebný počet pulzů je 26 na jednu otáčku.

Prvotní nápad byl po obvodu nalepit reflexní pásku, která by vytvořila kontrast oproti povrchu kola a tím generovala pulzy. Při testování se ale vyskytly dva problémy. Pro senzor bylo velmi obtížné rozlišit od sebe reflexní pásku a povrch kola, které je zhotoveno z lesklého plastu. Lesklé materiály obecně vykazují vyšší odrazivost než povrchy matné. Při zkoušení různých kombinací reflexních prvků a vzdáleností mezi snímačem a těmito prvky, se nakonec rozlišení těchto dvou povrchů podařilo docílit. Nevýhodou však byla příliš velká potřebná vzdálenost mezi snímačem a kolem, přibližně 20 mm.

Vzhledem k tomu, že tento IR modul nedisponuje možností nastavit jeho citlivost, bylo třeba vhodným způsobem provést úpravu reflexivity povrchu kola.

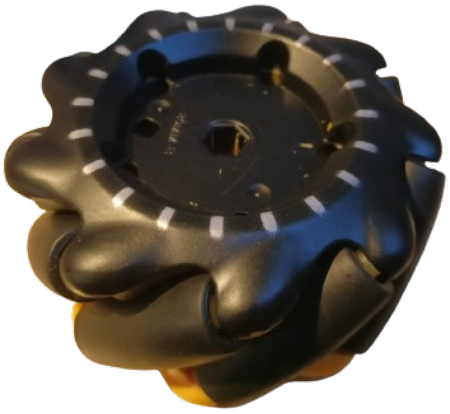
Proto byl povrch kola zdrsněn brusným papírem. Následně byla nanesena základová barva na plasty a na ni přišla vrstva černé matné akrylátové barvy. Po opětovné montáži kola bylo zjištěno, že nyní je odrazivost téměř nulová a snímač nezaregistruje přítomnost černého kola, ani když je s ním téměř v kontaktu. Nyní tak zbývalo vyřešit otázku reflexních prvků. Stále se nabízela možnost využití reflexní pásky, nicméně kvůli pracnosti tohoto postupu od něj bylo upuštěno. Nakonec bylo přistoupeno k výrobě tištěné šablony s „okénky“, do kterých byla následně nanesena stříbrná barva s vyhovující odrazivostí.

Při zkoušení této varianty vyšlo najevo, že snímač má stále sepnutý vstup. Po zkoumání bylo zjištěno, že problém je s geometrií reflexních prvků, konkrétně jejich odstupem a šířkou. Prvky byly příliš blízko sebe, a navíc příliš široké, což v kombinaci s geometrickou konfigurací dvou IR diod na modulu způsobovalo stále sepnutý výstup.

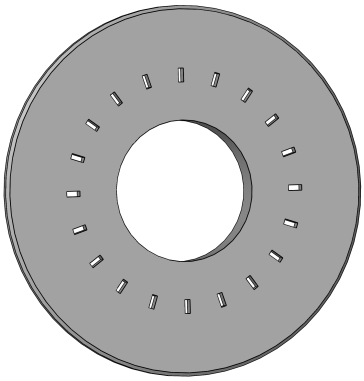
Proto bylo v dalším kroku přistoupeno ke snížení počtu pulzů na 20 a minimalizací jejich šířky. Takto zvolené rozmístění bylo již plně vyhovující. Lineární vzdálenost mezi pulzy se vlivem redukce počtu pulzů změnila na

$$s = \frac{360}{20} \cdot \frac{\pi}{180} \cdot 40 = 12,56\ mm \tag{11}$$

Obr. 30: Kolo před úpravou



Obr. 31: Kolo po úpravě



Obr. 32: Šablona na stříkání reflexních prvků

3.4 Řízení vozidla

Po zapojení a otestování všech HW částí bylo možné zabývat se výběrem software pro řízení autonomního vozidla.

3.4.1 Připojení periferií

Během instalace jednotlivých snímačů na řídicí PC k němu byly připojeny periferie pro ovládání a zobrazení, tedy myš, klávesnice a monitor. Z důvodu požadavku na mobilitu této platformy bylo ale nutné realizovat bezdrátové zapojení periferií. Možností, jak toto realizovat je celá řada.

Periferie umístěné na vozidle

Nabízí se použít např. TFT dotykový displej připevněný přímo na mobilní platformu, ze kterého by probíhalo řízení vozidla. Toto řešení ostatně využívá již zmíněná komerčně dostupná sada Yahboom ROSMaster X3. Výhodou tohoto řešení je bezesporu fakt, že

odpadá nutnost připojené klávesnice, myši, veškeré důležité informace a grafické prvky jsou zobrazeny přímo na platformě.

V případě použití tohoto vozidla v rozsáhlejších prostorech, kdy by platforma již nebyla v dohledu operátora, zcela přicházíme o informace poskytované programem pro řízení. Proto bude vhodnější použít bezdrátové spojení s vozidlem.

Bezdrátové připojení

Vzhledem k přítomnosti WiFi modulu na řídicím PC Nvidia lze využít některé způsoby vzdáleného ovládání. Zřejmě nejjednodušším a datově nejméně náročným způsobem vzdálené správy PC je SSH. Jedná se o nástroj pro příkazový řádek, kterým lze bezdrátově komunikovat mezi dvěma zařízeními, přenášet mezi nimi data, kopírovat soubory [33]. Nevýhodou tohoto řešení je nemožnost zobrazování grafických prvků. Proto by celé ovládání vozidla bylo omezeno na použití několika definovaných příkazů.
Požadavkem je mít co nejlepší kontrolu nad vozidlem, s čímž se pojí i snadné ovládání. Vhodnější se proto jeví použití některého z prostředků vzdálené správy s grafickým prostředím. Nabízí se využití programů třetích stran jako je TeamViewer, případně AnyDesk. V tomto případě ale nastává problém nekompatibility s procesory architektury ARM, kterou využívá řídicí počítač.
Dalším možným řešením může být využití VNC (Virtual Network Computing), tedy systému pro grafické sdílení obrazovky a kontrolu jiného zařízení. Přímo výrobce počítače doporučuje pro tyto účely využít program RealVNC Viewer. Na ovládaném zařízení poté stačí program stáhnout, povolit přihlášení do systému Ubuntu bez hesla, dále nastavit statickou IP adresu daného zařízení, aby bylo každé další připojení pohodlné a bez nutnosti hledat nově přidělenou IP adresu. Posledním krokem je nastavení hesla pro vzdálenou správu. Poté se nainstaluje totožný program do PC, ze kterého bude vozidlo ovládáno, v konfiguraci se nastaví požadovaná IP adresa, zadá heslo a nyní je již dostupný grafický přístup do řídicího PC vozidla.
Problém v podobě nezobrazení plochy nastal při zapnutí řídicího PC bez připojeného monitoru. Ta se zobrazila až po fyzickém připojení monitoru do vzdáleně ovládaného zařízení. Následně bylo zjištěno, že pro správné spuštění systému je vyžadována přítomnost monitoru.

Řešením bylo buď zakoupit tzv. dummy plug, tedy zařízení simulující přítomnost monitoru, nebo upravit konfiguraci při spouštění systému Ubuntu Linux tak, aby jako grafické výstupní zařízení byl nastaven fiktivní monitor, po jeho nastavení vše fungovalo správně. Bude-li v budoucnu třeba PC opět používat s fyzicky připojeným monitorem, bude nejprve nutno přes VNC nastavit zpět konfiguraci grafického výstupního zařízení.

3.4.2 Software

Po zprovoznění vzdáleného přístupu bylo nutné zvolit vhodné prostředky pro ovládání robota. Prvním krokem byl výběr jazyka. Vzhledem k tomu, že všechny obslužné programy pro snímače byly napsány v jazyce Python, a znalosti tohoto jazyka, byla použita právě tato alternativa. Dále byl vyhodnocen nejvhodnější způsob implementace algoritmů autonomní jízdy vozidla. Nabízelo se vytvořit vlastní algoritmy, nebo použít již existující řešení. Již v části 2.2 byl u několika produktů zmíněn ROS, což je balíček SW nástrojů, který usnadňuje vývoj a implementaci robotů.

Výhodou tohoto prostředí je přímá podpora OS Ubuntu Linux, dále, že tento projekt je open-source, tedy veškeré části tohoto projektu jsou volně k dispozici, libovolně upravitelné. Díky rozšířenosti tohoto programu i v průmyslovém prostředí vzniklo několik podpůrných balíčků s ovladači, implementací algoritmů, atd.

Knihovny

Aby bylo možné ovládat z PC Nvidia Jetson Orin Nano GPIO piny, bylo vhodné pro tento účel využít některou z již existujících knihoven pro jazyk Python. Přímo výrobce počítače doporučuje využití vlastní knihovny **Jetson.GPIO**, která vychází z knihovny **RPi.GPIO** pro ovládání totožného rozhraní na deskách Raspberry Pi. Ovládání jednotlivých pinů v programu je poté již poměrně jednoduché. V inicializační části bylo třeba rozhodnout, zda bude použito číslování pinů BCM (= Broadcom) či *BOARD*, které odpovídá číslování konektorů na již zmíněném 40pinovém GPIO headeru (Obr. 29). Následuje konfigurace konkrétního pinu, zda má být vstupem nebo výstupem. Poté jsou již stavy pinů řízeny logickými hodnotami typu *bool*.

Toto řešení se ukázalo jako vhodné pouze do chvíle, než bylo třeba k počítači připojit I2C modul pro rozšíření počtu PWM kanálů. Problém byl v I2C komunikaci modulu PCA9685 s počítačem. Z tohoto důvodu byla opuštěna knihovna **Jetson.GPIO**, kterou nahradilo prostředí **CircuitPython** vyvíjené společností Adafruit, jež je také výrobcem celé řady doplňkových modulů pro mikrokontroléry a jednodeskové počítače. Jedná se o implementaci jazyka Python v jazyce C umožňující programování mikrokontrolérů [34]. Další použitou knihovnou je **blinka**, v tomto případě jde o CircuitPython API knihovnu zprostředkující komunikaci s moduly. Výhodou použití tohoto prostředí byla i přítomnost knihovny pro modul PCA9685.

3.5 Napájení

Doposud bylo možné k napájení PC a snímačů využívat síťové adaptéry. Nicméně k tomu, aby mohlo být vozidlo mobilní, musí být jeho napájení vyřešeno bez přítomnosti vodičů vedoucích ven z vozidla. Proto bylo nutné vybrat vhodný alternativní napájecí zdroj.

Vzhledem k tomu, že nebude pro účely monitorování okolí při autonomní jízdě využit radar vyžadující samostatné napájení, bude třeba napájet pouze PC a H-můstky pro motory.

Dle technického listu od řídicího PC je rozmezí napájecího napětí mezi 9 V až 20 V [32]. Odebíraný proud při běhu počítače byl změřen pomocí ampérmetru laboratorního zdroje, přičemž hodnota byla okolo 2 A. Následně byl změřen proud potřebný pro napájení motorů v zátěži, který byl asi 1,5 A. Vzhledem k tomu, že je možné použít stejné napětí pro napájení H-můstků i PC, bylo rozhodnuto použít pouze jeden napájecí zdroj. Zdrojem je akumulátor přinášející výhodu opakovatelného nabíjení. Bylo třeba zvážit změřené parametry a na základě těchto informací vybrat vhodnou technologii akumulátoru.

Z běžně dostupných byly na výběr tři typy. NiMh, neboli nikl-metalhydridové, dále Li-Ion, tedy lithium-iontové a poslední variantou byly akumulátory Li-Pol, neboli lithium-polymerové. Nevýhodou akumulátorů technologie NiMh je jejich paměťový efekt, který lze potlačit pouze úplným vybíjením těchto akumulátorů. Navíc pro daný účel není vhodné příliš nízké jmenovité napětí jednoho článku 1,2 V.

Další variantou byl lithium-iontový akumulátor, který již netrpí paměťovým efektem, jmenovité napětí článku je 3,7 V, nicméně články s vyšší kapacitou jsou poměrně finančně nákladné. Navíc tato technologie neumožňuje přílišnou proudovou zátěž - vybíjení je průměrně 2C, nabíjení 0,5C [35]. To znamená, že pokud má baterie kapacitu např. 2200 mAh, lze ji vybíjet proudem $I_v = 2C = 2 \cdot 2200 = 4400\ mA$ a nabíjet maximálně $I_n = 1100\ mA$. Další nevýhodou Li-Ion technologie je samovybíjení akumulátorů, dále možnost jejich trvalého poškození při podbití.

Vzhledem k uvedenému zbývala poslední varianta, a tou bylo použití technologie lithium-polymerových akumulátorů. Tato technologie má výhradní využití v pohonu RC modelů zejména z důvodu velkých vybíjecích proudů a nízké hmotnosti. Ty se pohybují zpravidla okolo 25C, tedy pro akumulátor s kapacitou již zmíněných 2200 mAh vychází vybíjecí proud $I_v = 25C = 25 \cdot 2200 = 55000\ mA$. Jmenovité napětí jednoho článku je 3,7 V. Akumulátory často obsahují více článků řazených sériově. Napětí je dáno součinem počtu článků a napětí jednoho článku. V případě dané aplikace bylo třeba vyřešit kompromis mezi hmotností a dostatečnou kapacitou. Proto byl vybrán akumulátor s následujícími parametry:

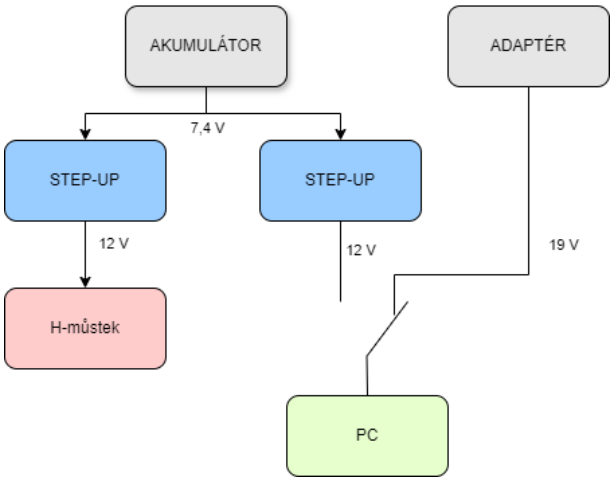
Technologie:	Li-Pol
Kapacita:	2250 mAh
Počet článků:	2S
Jmenovité napětí:	7,4 V
Vybíjecí proud:	25C
Nabíjecí proud:	3C

Obr. 33: Li-Pol akumulátor Power X6 [36]

Vzhledem k tomu, že napětí vybraného akumulátoru je nižší než minimální požadované počítačem, musel být použit zvyšující měnič napětí. Pro tento účel byl vybrán modul s integrovaným obvodem XL6009, který má nastavitelné výstupní napětí od 5 V do 35 V. Maximální výstupní proud má hodnotu 4 A [37]. Dle specifikace by měl jeden tento modul proudově vyhovovat.

Nicméně z důvodu přítomnosti nadproudové ochrany měniče docházelo při současném provozu PC a motorů k náhlému odpojení jeho výstupu, což způsobilo okamžité vypnutí PC. Z tohoto důvodu nakonec byly použity dva tyto měniče, přičemž jeden obstarává pouze napájení PC, druhý dodává výkon H-můstkům pro napájení motorů.

Napájecí větev pro PC je zapojena přes kolébkový přepínač, který umožňuje zvolit, zda-li bude napájen z akumulátoru, nebo ze síťového adaptéru. Napájení adaptérem je vhodné využít ve stádiu programování, případně ladění programů a algoritmů.



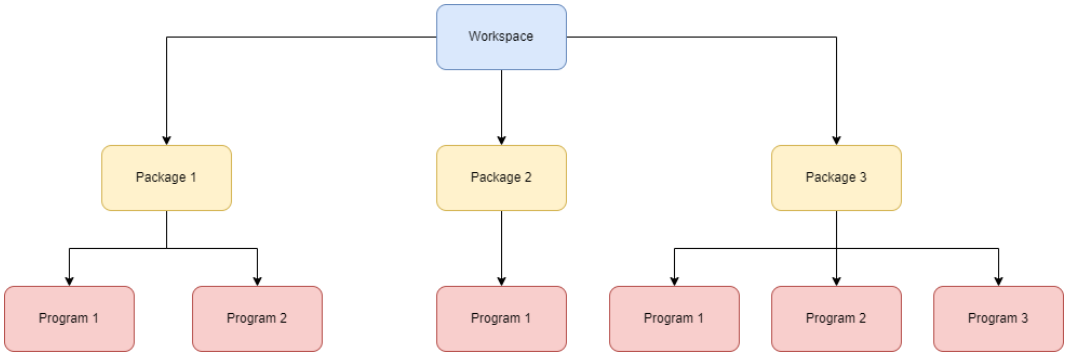
Obr. 34: Schéma napájení

3.6 ROS

S ohledem na všechny okolnosti bylo rozhodnuto o použití software ROS, který by měl usnadnit práci při implementaci algoritmů potřebných pro autonomní jízdu vozidla. Základním principem systému ROS je rozdělení systému do několika subsystémů, přičemž každý obsluhuje jednu konkrétní činnost. Tyto subsystémy mezi sebou potom komunikují, čímž je možno vytvořit i velmi složité struktury, které jsou ale vzájemně oddělitelné, a proto jednoduše spravovatelné. Další výhodou ROS je opakované vyžívání již existujících kódů, případně algoritmů tak, aby byl urychlen vývoj.

Toto prostředí je možné programovat v několika jazycích - Python, C++, Lisp [4]. Architektura systému ROS si také klade za cíl být přehledná a jednoduchá, proto základní struktura obsahuje tzv. workspace, neboli pracovní prostředí, kterých může být nesčetně. Uvnitř každého pracovního prostředí se poté nacházejí tzv. packages, neboli balíčky. Tyto

balíčky již obsahují obslužné programy pro jednotlivé části robotických platforem. V případě dané platformy bylo vytvořeno několik vlastních balíčků, naopak pro implementaci algoritmů navigace a autonomní jízdy byly využity již existující balíčky.



Obr. 35: Hierarchie systému ROS

3.6.1 Komunikace

Vzhledem k rozdělení systému do několika menších částí je třeba, aby mezi sebou jednotlivé uzly komunikovaly. V systému ROS jsou k tomuto účelu používány převážně dvě metody. První variantou je komunikace mezi uzly (nodes) pomocí témat (topics), kde předmětem komunikace je message (zpráva), viz Obr. 36. Do jednoho tématu může přispívat teoreticky neomezený počet uzlů, stejně tak je tomu se čtením z tématu.

Druhou variantou je architektura klient-server. Ta funguje analogicky k systému node to node s tím rozdílem, že klient pošle žádost (request) s určitými parametry, kterou přijme server, zpracuje ji a následně posílá odpověď (response). Dalším rozdílem v tomto případě je fakt, že mezi sebou vždy komunikují pouze dva uzly [4].

Node

Node, česky uzel, je část systému, která je schopna číst či publikovat zprávy do daného tématu. Uzly lze rozdělit dle způsobu manipulace s informacemi, zda je informace publikována - potom se jedná o tzv. publisher, neboli přispěvatel, nebo je daná informace získávána, poté jde o tzv. subscriber, neboli odběratel.

Konkrétní implementace poté obnáší tvorbu kódu v jazyce Python, který na pozadí vykonává např. vyhodnocování dat a následně jej pomocí uzlu přepošle jako přispěvatel do tématu. V inicializaci každého uzlu je třeba uvést, zda-li jde o přispěvatele, či odběratele, a do jakého tématu přispívá, resp. z něj čte. Dále je třeba uvést, jaký typ zprávy je očekáván. ROS totiž disponuje různými typy zpráv tak, aby bylo dosaženo lepší čitelnosti kódu a zjednodušení. Pro názornost je zde uveden obsah zprávy typu *Twist* z balíčku *geometry_msgs*. Tato zpráva je určena pro přenášení informace o rychlosti robota, takže obsahuje tři složky pro jednotlivé směry lineární rychlosti a tři složky pro úhlové rychlosti. Aby mohla být požadovaná

informace přenesena, je tedy třeba, aby dva uzly sdílely zprávu pomocí stejného tématu. Souhlasit musí také typ zprávy.

Topic

Topic, neboli téma, je jakýmsi prostředníkem mezi dvěma uzly, k němuž se připojí libovolný počet uzlů a může do něj přispívat, případně z něj číst data. Toto přináší výhodu anonymizace, tedy jednotlivé uzly na sebe navzájem „nevidí“, čímž je zajištěna bezpečnost. Pomocí příkazů v terminálu lze následně zobrazit seznam všech témat, případně číst jejich obsahy.

Message

Message, neboli zpráva, je informace předávaná mezi jednotlivými uzly prostřednictvím již zmíněných témat (topiců). ROS umožňuje využití několika již definovaných typů zpráv konkrétně určených pro robotiku - obsahují souřadnice robota, jeho rychlosti, dokáže přenášet data ze snímačů ve vhodném formátu, atd. Systém nabízí tvorbu vlastních zpráv, což může být výhodné v případě specifických podmínek použití.

Obr. 36: Struktura komunikace systému ROS s jedním přispěvatelem a dvěma odběrateli [38]

3.6.2 RViz

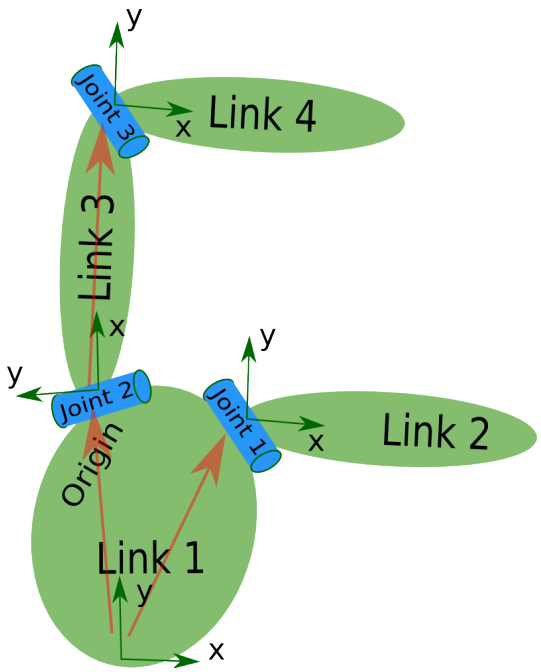
Jedná se o zkratku slov Robot Visualization a slouží k vizualizaci nejen robota, ale umožňuje zobrazit výstupy ze snímačů, souřadné systémy, trajektorie, atd. Vzhledem k tomu, že jde o nástavbu systému ROS, funguje na stejném principu, tedy má vytvořené uzly, které se připojují na jednotlivá témata, a z nich získávají data potřebná pro vizualizaci. Například balíček pro obsluhu 2D lidaru publikuje zprávu typu *laser_scan* do tématu *scan*, ke kterému se následně připojí nástroj RViz, a tyto hodnoty graficky zobrazí.

3.6.3 Model robota

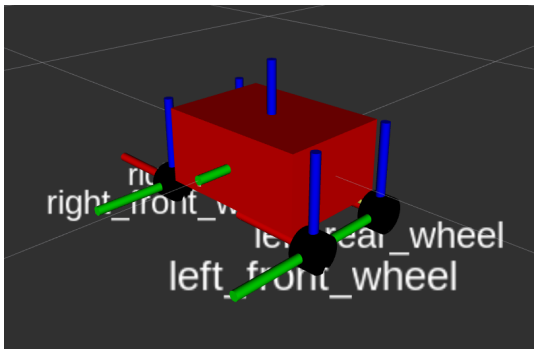
Nejen pro účely vizualizace, ale i z důvodu výpočtu kolizí robotického vozidla s okolními objekty, bylo nutno v dalším kroku vytvořit model robota. K jeho vytvoření se využívá tzv. URDF soubor ve formátu XML. Model robota byl tvořen jednoduchými příkazy pro generování základní geometrie - válce, kvádry, atd. Každé těleso je zpravidla definováno pravoúhlým souřadným systémem a následně charakteristickými rozměry daného tělesa.

Zmíněné souřadné systémy hrají roli v geometrických transformacích. Všechny roboty mají zpravidla více než jeden souřadný systém. Dle konvence je jeden systém hlavní a ostatní jsou následně popsány ve vztahu k tomuto systému. Základna robota se v dané terminologii označuje jako *base_link*, ostatní jsou poté *link_X*, kde X reprezentuje číslo tělesa. Jednotlivé linky jsou poté svázány klouby (joints) a dle typu kloubu je určeno, kolik bude danému tělesu odebráno stupňů volnosti. Pro kloub typu kolo je využit kloub nesoucí označení *continuous*, odebírající tělesu kola 5 stupňů volnosti a umožňující neomezenou rotaci v ose jediného stupně volnosti. Dále jsou v souboru URDF definovány vztahy mezi jednotlivými klouby. To znamená, který link je nadřazený (typu rodič) a který typu potomek, příslušící konkrétnímu rodiči. Vztah má omezení v té podobě, že rodič může mít více potomků, ale potomek pouze jednoho rodiče.

Použité souřadné systémy jsou pravoúhlé kartézské. Kladná orientace osy X směřuje dopředu, zatímco osa Y má kladný směr vlevo. Osa Z je poté binormálou k těmto dvěma osám s kladnou orientací vzhůru.



Obr. 37: Struktura geometrických transformací v URDF [39]



Obr. 38: Model robota v prostředí RViz

3.6.4 Spouštění souborů

Spouštění systému ROS a jednotlivých programů probíhá pomocí Linux terminálu (příkazového řádku). Spouštějí se jednotlivé uzly v balíčcích. Vzhledem k tomu, že pro každé další spuštění nového uzlu je třeba další instance terminálu, může dojít k velmi rychlému zahlcení obrazovky jednotlivými okny terminálu, která musí po dobu běhu těchto uzlů zůstat otevřená. Z tohoto důvodu přináší ROS řešení v podobě *launch* souboru, který umožňuje spouštění více uzlů najednou. Při spouštění projektu tedy lze zavolat pouze jeden *launch* soubor namísto mnohdy až desítek souborů. Stejně jako v případě URDF souborů popsaných výše, využívá i tento typ souboru jazyk XML.

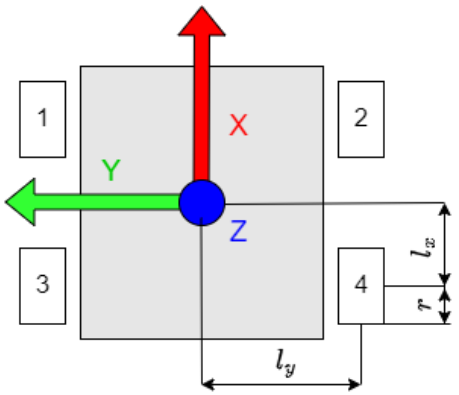
3.6.5 Manuální řízení

Po vytvoření modelu robota a seznámení se se základními principy ROS mohlo být přistoupeno k tvorbě programu pro manuální řízení modelu vozidla, které bylo později důležité pro mapování okolního prostředí.

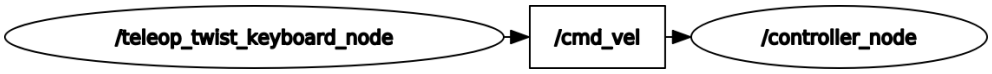
K tomuto účelu byl vytvořen balíček *motor_controller* a v něm program s názvem *jog*, který má za úkol zpracovávat přijatá data z klávesnice, sloužící k manuálnímu řízení. Část vyhodnocování informací o stisknutých klávesách na klávesnici vyhodnocuje program *teleop_twist_keyboard*, jež je součástí základní instalace ROS. Program obsluhující klávesnici navíc publikuje příkazy o žádané rychlosti do tématu nazvaného *cmd_vel* pomocí zprávy typu *Twist*. Z tohoto tématu následně odebírá informace program pro obsluhu motorů. Znázornění popsané komunikace pomocí nástroje *rqt_graph* v ROS je na Obr. 40. Aby bylo možné převést složky rychlostí $\{v_x, v_y, \omega_z\}$ z tématu *cmd_vel* na otáčky jednotlivých kol, je třeba kinematický model vozidla, který byl převzat z [40]. Konkrétně zde bude užita matice (12) pro výpočet úhlových rychlostí jednotlivých kol z požadavků na rychlosti ve třech zmíněných složkách.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & -1 & -(l_x + l_y) \\ 1 & 1 & (l_x + l_y) \\ 1 & 1 & -(l_x + l_y) \\ 1 & -1 & (l_x + l_y) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} \tag{12}$$

Tato matice popisuje vztahy pro výpočet úhlových rychlostí na základě vektoru $\{v_x, v_y, \omega_z\}$. Poloměr kola je označen r, vzdálenosti l_x a l_y odpovídají vzdálenostem kola od geometrického středu modelu vozidla - viz Obr. 39.



Obr. 39: Schéma vozidla se souřadným systémem



Obr. 40: Struktura komunikace manuálního řízení

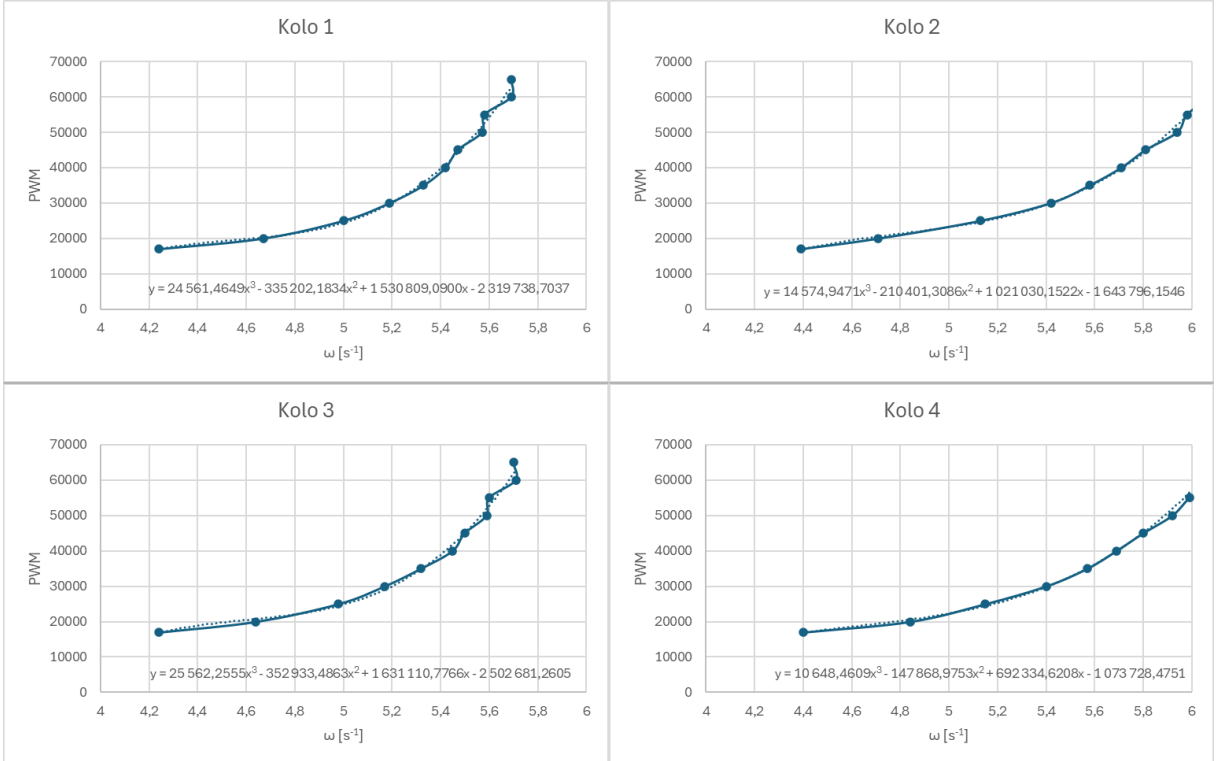
Vzhledem k tomu, že otáčky jsou řízeny pulzně šířkovou modulací z modulu PCA9685, a bylo zjištěno, že závislost otáček na PWM střídě není lineární, bylo třeba provést experiment s cílem zjistit tuto závislost. Na základě tohoto experimentu a z matice (12) lze poté určit konkrétní hodnotu PWM střídy pro dosažení žádaných otáček.

K experimentu byl vytvořen skript v jazyce Python, který umožňuje nastavit vzdálenost, jakou má vozidlo ujet, a zároveň měřit otáčky jednotlivých kol. Po skončení tohoto programu automaticky vypíše průměrné otáčky a dosaženou lineární rychlost v m/s. Postupně byla zvyšována PWM střída, dokud se motory neroztočily. PWM bylo zvyšováno téměř do maxima odpovídajícímu 16 bitům, tedy hodnotě 65536, přičemž tato hodnota je následně přemapována na skutečné rozlišení modulu PCA9685, tedy 12 bitů. Poté bylo provedeno několik měření s cílem zjistit závislost $\omega = f(PWM)$. Naměřené hodnoty jsou v Tab. 4.

Tab. 4: Naměřené závislosti $\omega = f(PWM)$

PWM	Kolo 1		Kolo 2		Kolo 3		Kolo 4	
	$n\ [min^{-1}]$	$\omega_1\ [s^{-1}]$	$n\ [min^{-1}]$	$\omega_2\ [s^{-1}]$	$n\ [min^{-1}]$	$\omega_3\ [s^{-1}]$	$n\ [min^{-1}]$	$\omega_4\ [s^{-1}]$
17000	40,5	4,24	41,9	4,39	40,5	4,24	42	4,4
20000	44,6	4,67	45	4,71	44,3	4,64	46,2	4,84
25000	47,7	5	49	5,13	47,6	4,98	49,2	5,15
30000	49,6	5,19	51,8	5,42	49,4	5,17	51,6	5,4
35000	50,9	5,33	53,3	5,58	50,8	5,32	53,2	5,57
40000	51,8	5,42	54,5	5,71	52	5,45	54,3	5,69
45000	52,2	5,47	55,5	5,81	52,5	5,5	55,4	5,8
50000	53,2	5,57	56,7	5,94	53,4	5,59	56,5	5,92
55000	53,3	5,58	57,1	5,98	53,5	5,6	57,2	5,99
60000	54,3	5,69	58	6,07	54,5	5,71	58,1	6,08
65000	54,3	5,69	58,2	6,09	54,4	5,7	57,6	6,03

Z hodnot v Tab. 4 byly vytvořeny grafy, které byly posléze proloženy polynomem 3. stupně, který dostatečně kvalitně reprezentoval naměřenou křivku, a byla zjištěna rovnice tohoto polynomu pro všechna čtyři kola - viz Obr. 41. V souladu s těmito rovnicemi a matice (12) bylo možné nastavit hodnotu PWM tak, aby všechna kola měla přibližně stejné otáčky.



Obr. 41: Průběhy závislostí $\omega = f(PWM)$ pro všechna kola

3.7 SLAM

Prvním krokem pro autonomizaci vozidla je schopnost orientovat se v prostoru. Aby tohoto bylo dosaženo, je nutné vytvořit mapu prostředí, v němž se má robot pohybovat. Na základě této mapy a při následné lokalizaci vozidla v této mapě bude posléze možné implementovat algoritmy obsluhující autonomní jízdu. Proces tvorby mapy je označovaný jako SLAM (= Simultaneous Localization and Mapping). Jak již vyplývá z názvu tohoto procesu, jde o současnou lokalizaci a mapování neznámého prostředí.

Aby mohl být zvolen nejvhodnější algoritmus pro SLAM, muselo být rozhodnuto o výběru snímače pro mapování prostředí. Vzhledem k tomu, že mapa představuje půdorys daného prostředí ve 2D prostoru, byl použit snímač RPLidar A2, který vytváří půdorys prostředí. Navíc ROS již obsahuje balíčky určené pro tvorbu statických map z tohoto typu snímače.

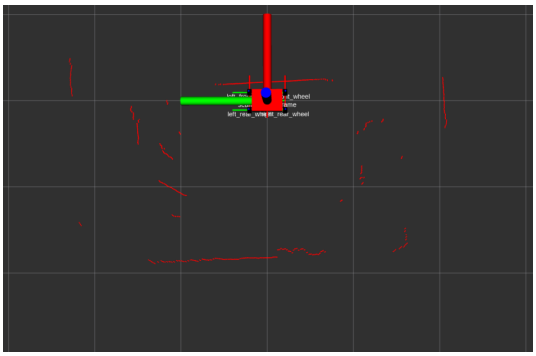
3.7.1 Tvorba mapy

Pro tvorbu mapy byl použit přístup tzv. grid mappingu spočívající v rozdělení prostoru do rovnoměrné mřížky, nejčastěji tvaru šestiúhelníku, či čtverce a následné vyplňování této mřížky. Mřížka je postupně na základě odrazů paprsků lidaru od objektů (překážek) zabarvována. Interpretace prostředí je poté velmi jednoduchá, neboť se využívají pouze odstíny šedi v 8bitovém rozlišení. Přičemž odstíny blíže černé znamenají překážku v prostoru, zatímco bělejší odstíny představují volný prostor. Na Obr. 44 je patrný ještě třetí odstín šedi, ležící mimo ohraničenou oblast. Tato barva označuje neznámé prostředí. Prahové hodnoty pro rozlišení mezi překážkou a volným prostorem lze v algoritmu libovolně nastavit. V software ROS se o tvorbu mapy stará přímo algoritmus SLAM. Vytvořená mapa je v binárním formátu PNG, lze ji následně v grafických programech dle potřeby upravovat. S každou mapou je následně asociován *yaml* konfigurační soubor, který popisuje umístění souřadného systému mapy, její měřítko pro přepočet do reálného světa a prahy mezi volnou oblastí a překážkou.

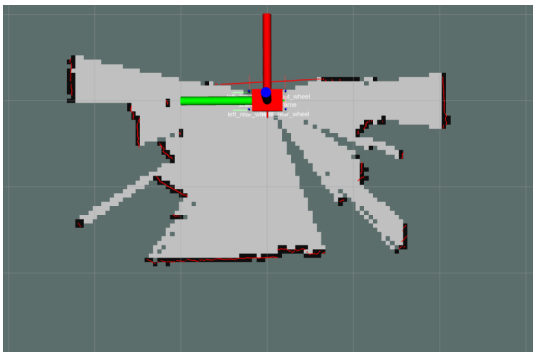
Princip tvorby mapy spočívá ve zmapování celého prostředí, které je uvažováno pro následné použití při autonomní jízdě. Vozidlo tedy musí projet tento prostor, přičemž je řízeno manuálně pomocí klávesnice operátorského PC. Obecně lze říci, že na kvalitu mapy má vliv lokalizace, rychlost pohybu robota a také použitý snímač.

3.7.2 Lokalizace

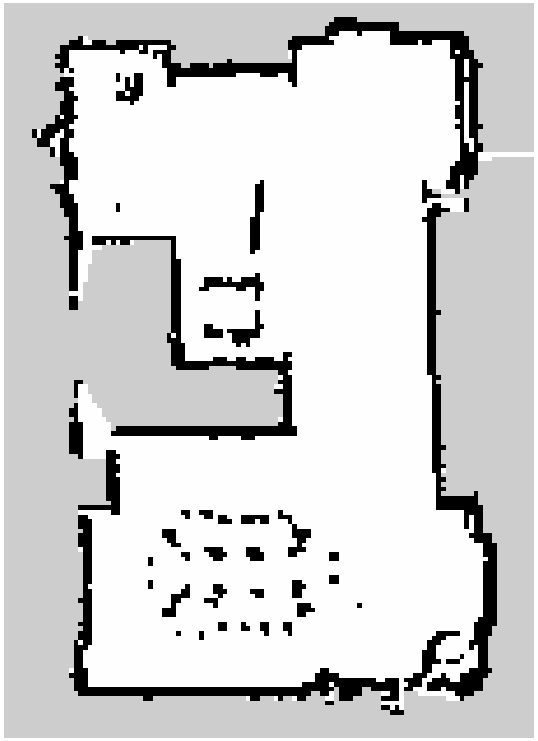
Nedílnou součástí SLAM je kromě mapování také lokalizace. Zřejmě nejjednodušší variantou je použití odometrie kol. Tento proces využívá měření otáček jednotlivých kol k zjišťování změny polohy robotického vozidla v čase. Z tohoto důvodu byla kola opatřena reflexními značkami umožňujícími odometrii. Aby bylo možné vyhodnocovat změnu polohy, je nutno vytvořit kinematický model tohoto vozidla, z nějž byl proveden přepočet na změnu



Obr. 42: Obraz dat z lidaru v RViz



Obr. 43: Proces tvorby mapy v RViz



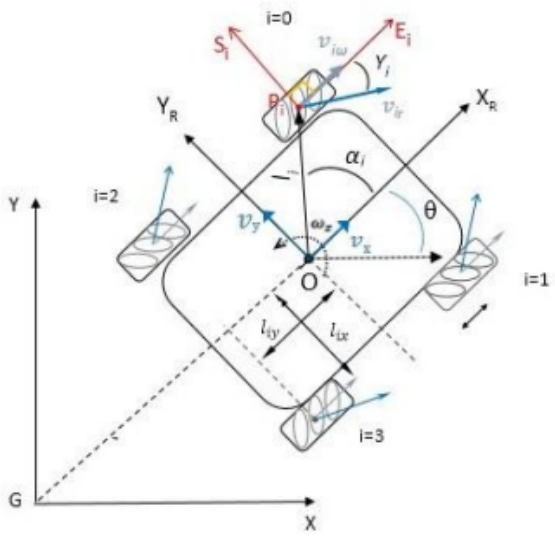
Obr. 44: Vytvořená konečná mapa místnosti

polohy. Ačkoliv tento způsob zjišťování polohy je poměrně jednoduchý, má i nevýhody. Ty vznikají z důvodu možného prokluzu kol a nepřesností matematického modelu. V důsledku popsaného může dojít ke zhoršení lokalizace. Navíc tato chyba s časem, resp. ujetou vzdáleností narůstá, proto je vhodné využít algoritmy sensor-fusion, tedy sloučení dat z několika nezávislých snímačů, a použít ta data, u nichž je očekávána vyšší přesnost.

Druhá varianta lokalizace při mapování je využití jiného balíčku ROS nevyžadujícího odometrii kol. Použitý balíček se jmenuje *hector_mapping* a umožňuje lokalizaci na základě porovnávání již vytvořeného segmentu mapy prostředí s aktuálním obrazem z lidaru. Tento balíček byl proto s výhodou použit k tvorbě mapy prostředí. Bližší informace k algoritmu lokalizace bohužel nebyly k nalezení, neboť dokumentace k tomuto balíčku je zpoplatněná.

3.7.3 Kinematický model

Bylo třeba vytvořit kinematický model robotické platformy, který umožňuje transformace z fixní souřadné soustavy do pohyblivé (konstrukce robota). Tento model byl převzat z [40].



Obr. 45: Kinematický model vozidla [40]

Z Obr. 45 je nyní třeba určit složky rychlosti v_x, v_y a ω_z. Po jednoduché integraci vzhledem k času těchto veličin se následně získá ujetá vzdálenost v obou směrech a informace o natočení. Jak je patrné z obrázku, poloha vozidla je v rovině určena pouze třemi souřadnicemi. Konkrétně vzdálenostmi x a y a úhlem natočení θ vůči kladné polorovině souřadného systému. Díky tomuto modelu lze poté pouze na základě znalosti otáček jednotlivých kol určit rychlost, resp. polohu vozidla. K tomuto účelu bude použita matice (13) převzatá z [40] v podobě

$$\begin{bmatrix} v_x \\ v_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} & -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \tag{13}$$

přičemž vektor na levé straně označuje hledané souřadnice lineárních rychlostí v_x, v_y a úhlové rychlosti otáčení kolem osy z ω_z. Po úpravě do podoby změny polohy se vektor na levé straně změní do podoby Δx, Δy a natočení $\Delta\theta$. Vektor úhlových rychlostí jednotlivých kol na pravé straně se změní na vektor pootočení, přičemž jednotlivé členy tohoto vektoru budou označovány jako *tick*, neboli dráha ujetá mezi dvěma pulsy enkodéru. Matice (14) pro výpočet změny polohy ve 2D prostoru má podobu

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} & -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} \end{bmatrix} \begin{bmatrix} ticks_1 \\ ticks_2 \\ ticks_3 \\ ticks_4 \end{bmatrix} \tag{14}$$

3.7.4 Souřadné systémy

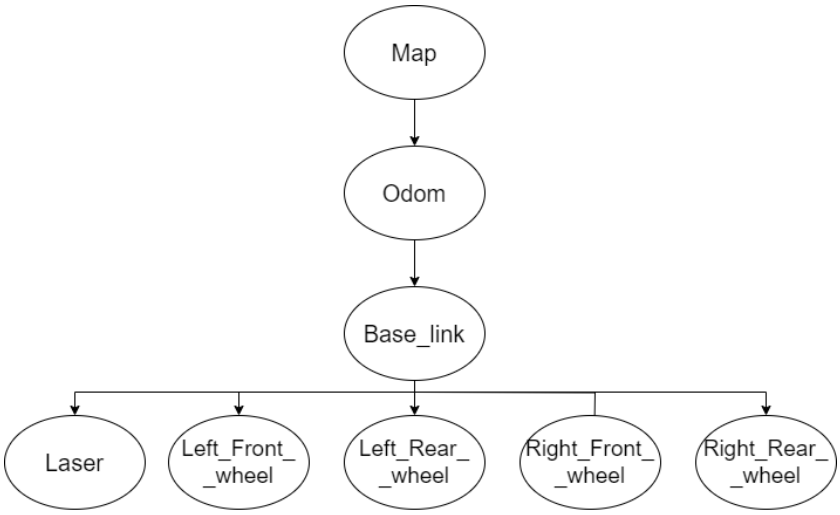
Transformace v ROS lze rozdělit na **statické** a **dynamické**. Statické transformace jsou v čase neměnné, jedná se například o transformaci souřadného systému lidaru do systému těla vozidla. Naopak dynamické transformace jsou ty, které se v čase mění, příkladem může být vzájemná poloha těla vozidla a natočení kol.

Kromě již popsaných souřadných systémů přímo na vozidle jsou používány ještě další dva. Prvním je *map* neboli mapa. Jedná se o pevný souřadný systém, který je nadřazený všemu. Tento souřadný systém reprezentuje reálný svět. Právě vůči tomuto souřadnému systému jsou následně popisovány všechny ostatní transformace. Přímo z mapy je následně určována transformace do systému nazvaného *odom* neboli odometrie. V případě, že by neexistovaly chyby měření odometrie, systémy *odom* a *map* by splynuly do jednoho. Podřazeným systémem odometrie je poté již tělo vozidla, označované jako *base_link*. Kaskáda transformací je patrná z Obr. 46.

Souřadný systém *odom* garantuje spojitost transformace z tohoto systému na *base_link* v čase, poskytuje relativně přesné transformace v horizontu krátkého času, nicméně jeho slabinou je již zmíněná narůstající chyba. Naopak transformační vztah mezi *map* a *base_link* již v čase nemusí být spojitý v důsledku skoků způsobených korekcí lokalizace vozidla vůči mapě [41].

K účelu výpočtu geometrických transformací je v ROS připraven balíček s názvem *tf*2, který publikuje do tématu *tf*. Výhodou tohoto balíčku je jednoduchá možnost realizace transformací pomocí příkazů obsahujících souřadnice transformovaného souřadného systému v podobě $\{x, y, z, \theta\}$. Další nezbytnou částí pro úspěšnou transformaci je publikování zprávy typu *Odometry* do tématu *odom*. Tato zpráva obsahuje informace o poloze transformovaného souřadného systému a jeho rychlosti ve všech směrech. Obě informace obsažené ve zprávě typu *Odometry* lze ještě doplnit kovarianční maticí, která reflektuje nejistotu a šum naměřených dat, případně jejich vzájemnou závislost.

Transformaci mezi částmi *odom* a *base_link* zajišťuje odometrie vozidla, tedy data získaná z výše popsaného kinematického modelu. Vzhledem k tomu, že *odom* je nadřazena *map*, je třeba vyřešit i tuto transformaci tak, aby byla minimalizována chyba způsobená výše popsanými nedostatky odometrie. O tento vztah se stará lokalizace, která má za úkol korigovat polohu souřadného systému *odom* v systému *map* tak, aby poloha vozidla na mapě odpovídala skutečnosti.



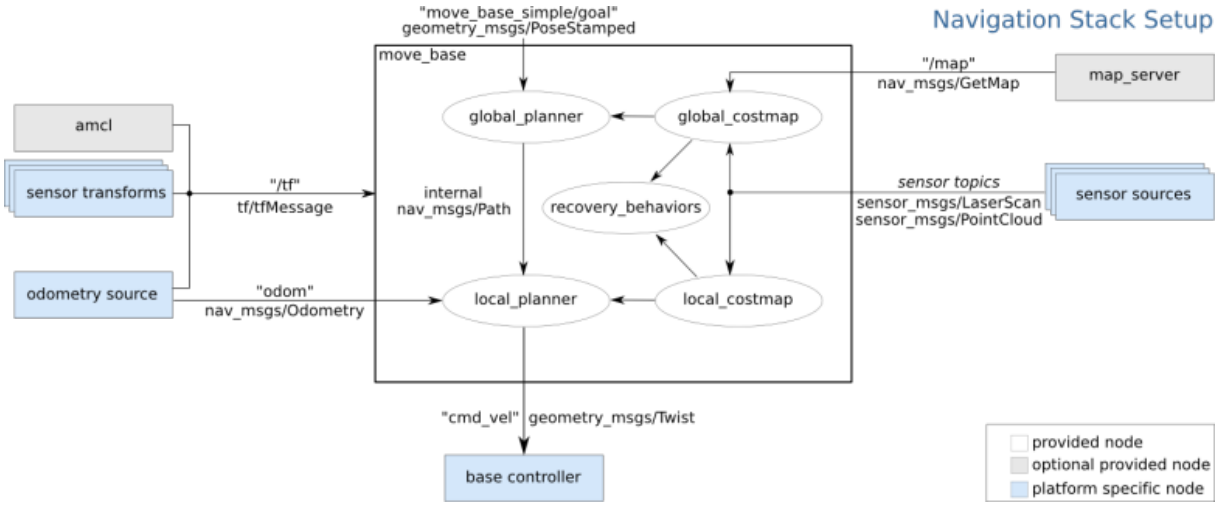
Obr. 46: Strom všech transformací robota vůči mapě

3.8 Navigace

Po vytvoření mapy a schopnosti určovat polohu vozidla vzhledem k statickému souřadnému systému pomocí odometrie bylo přistoupeno k dalšímu kroku potřebnému pro schopnost autonomní jízdy. Tímto krokem bylo vytvoření navigace. V principu se jedná o to, že je žádoucí vozidlo přesunout z počáteční polohy do nějaké jiné. Na základě dat odometrie je již možno určovat polohu a rychlost vozidla, manuálně jej ovládat. V případě využití navigace bude manuální ovládání nahrazeno příkazy pro žádanou rychlost a směr jízdy vozidla generovanými z plánovačů, které budou následně popsány. Výhodou v tomto případě je, že způsob řízení motorů zůstane totožný ať v případě manuálního řízení, tak i pomocí navigačních algoritmů, neboť je zachováno čtení a publikování z, resp. do tématu s názvem *cmd_vel*.

Kvalita autonomní jízdy je pochopitelně tím vyšší, čím jsou lepší odometrická data. Ta lze vylepšit, jak bylo již uvedeno, algoritmem sensor-fusion využívajícím Kalmánův filtr. Velmi častou kombinací snímačů pro zlepšení přesnosti odometrie je enkodér + IMU (akcelerometr a gyroskop). Pro tuto kombinaci snímačů existuje v ROS balíček nazvaný *robot_pose_ekf*. Pro čtení dat z IMU umístěného v 3D lidaru Intel L515 byl vytvořen program v jazyce Python, který tato data zpracovával a následně je publikoval pomocí zprávy typu *Imu* do tématu *imu_data*. Zpráva zmíněného typu obsahuje celkem tři vektory dat a ke každému vektoru kovarianční matici. Prvním vektorem je orientace ve 3D prostoru, druhý vektor obsahuje úhlovou rychlost, kterou měří gyroskop, a posledním je vektor lineární akcelerace. Bylo zjištěno, že balíček *robot_pose_ekf* vyžaduje vyplnění kovariančních matic, aby fungoval. Vzhledem k tomu, že výrobce neudává data o přesnosti IMU, nebylo možné tyto matice vyplnit odpovídajícími hodnotami rozptylů jednotlivých veličin na hlavní diagonále a jejich vzájemnou závislostí na ostatních pozicích v matici. Proto nebyl tento algoritmus využit.

Na Obr. 47 je patrná struktura navigačního balíčku v ROS. Uprostřed obdélníku jsou vyznačeny ty části struktury, které jsou zodpovědné za plánování trasy, vyhýbání se překážkám a generování příkazů pro rychlost vozidla. Z vnější strany obdélníku jsou poté rozmístěny ty části navigační struktury, které vstupují jako data ze snímačů, statické mapy a vystupují v podobě žádaných hodnot rychlostí. Z barevné legendy na Obr. 47 je patrné, že některé části jsou již implementovány v ROS, jiné je třeba ručně implementovat, neboť se liší v konkrétních případech užití. V následujících kapitolách budou popsány jednotlivé části zobrazené navigační struktury.



Obr. 47: Struktura navigace v ROS [4]

3.8.1 Amcl

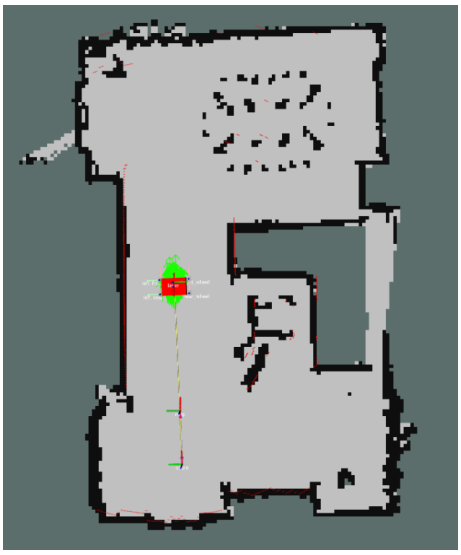
Tato písmena zkracují anglický výraz adaptive Monte Carlo localisation. Poslední slovo této zkratky napoví, že jde o algoritmus určený k lokalizaci robota ve statické mapě. Konkrétně má za úkol určit jeho polohu popsanou třemi již zmíněnými souřadnicemi $\{x, y, \theta\}$. Tento algoritmus má na vstupu mapu prostředí, monitoruje data z odometrie a v potaz bere také pozorování, v tomto případě z lidaru. Na základě těchto dat a znalosti kinematického modelu vozidla tento algoritmus určuje pravděpodobnou polohu, resp. množinu možných následujících stavů. Princip je založen na částicovém filtru, kdy každá částice v mapě reprezentuje možnou polohu robota. Při spuštění tohoto algoritmu nemá algoritmus dostatek dat pro určení polohy, proto jsou částice rovnoměrně rozložené po celé mapě. Pokud je zadána očekávaná počáteční poloha vozidla, nacházejí se částice v blízkém okolí vozidla. Toto chování je patrné z Obr. 48, kde zelené šipky reprezentují množinu odhadovaných stavů bez dostatečného počtu pozorování, přičemž směr šipky odpovídá odhadované orientaci vozidla.

Jak se vozidlo pohybuje po mapě, tím vznikají nová pozorování, jednotlivé částice dostávají váhy dle toho, jak je který stav (poloha) pravděpodobný. Adaptivita tohoto algoritmu spočívá v tom, že vzorky s nižší než prahovou hodnotou váhy nejsou brány v potaz, a nové

převzorkování se provede pouze se vzorky s dostatečnou váhou [42]. Tímto procesem se určení polohy robota v čase postupně zpřesňuje (viz. Obr. 49). Výstupem tohoto algoritmu je očekávaná pozice robota a její kovariance. Čím jsou menší jednotlivé prvky v kovarianční matici, tím je vyšší přesnost odhadu polohy vozidla. Balíček AMCL v ROS má celou řadu parametrů, které je třeba nastavit dle konkrétního případu užití. Lze nastavit minimální a maximální užitý počet částic filtru, dále očekávané chyby měření odometrie, lidaru, periodu převzorkování atd. Vhodným nastavením parametrů lze potlačit vliv chyb odometrie, měření lidaru na určení polohy robota. Algoritmus má následující kroky

1. Rozdělení částic do mapy dle aktuálního pozorování a kinematického modelu
2. Přidělení váhy částicím na základě pravděpodobnosti konkrétního stavu
3. Určení množiny možných poloh na základě částic s nejvyšší váhou
4. Převzorkování částic

Obr. 48: Odhad polohy dle AMCL na začátku



Obr. 49: Odhad polohy dle AMCL po ujetí cca 2,5 m

3.8.2 Sensor transforms

Dle barevné legendy v Obr. 47 je zřejmé, že jde o první část struktury, kterou je třeba implementovat, neboť je specifická pro každého robota. Konkrétně se jedná o provedení transformací ze souřadného systému lidaru do systému základny robota, dále pak ze systému označovaného jako *odom* opět do základny robota, neboli *base_link*. Toto je realizováno jednak pomocí již zmiňovaného URDF souboru, popisujícího geometrii a vztahy mezi jednotlivými částmi vozidla, zatímco dynamické transformace jsou získávány pomocí jednoduchého programu v jazyce Python, který řeší transformaci mezi systémy *odom* a *base_link*. Tyto transformace jsou následně publikovány do tématu *tf*, ze kterého tyto informace následně čte navigační balíček. Strom uvedených transformací je na Obr. 46.

3.8.3 Odometry source

Další částí vyžadující implementaci závislou na konkrétním případě užití je zdroj dat odometrie. Již bylo popsáno, že tímto zdrojem je kinematický model vozidla, resp. enkodéry jednotlivých kol. O publikování zprávy do tématu *odom* se opět stará kód v jazyce Python.

3.8.4 Map server

Tato část je již implementována a stará se o načtení uložené mapy prostředí, v němž se bude robot pohybovat. Na základě této mapy dochází k lokalizaci robota a plánování jeho trasy pohybu.

3.8.5 Sensor sources

Aby byla možná lokalizace a vytváření lokální mapy prostředí, je nutné číst aktuální data ze snímačů. V tomto případě jde o 2D lidar, který publikuje zprávy typu *LaserScan* se specifickým formátem přizpůsobeným standardizovanému výstupu z tohoto snímače. Z Obr. 47 je patrné, že jako zdroj dat lze mít i zprávy typu *PointCloud*, nicméně toto by platilo v případě snímačů měřících 3D prostor.

3.8.6 Base controller

Jediným výstupem z navigační struktury je uzel ovládající motory vozidla. Base controller část specifická pro konkrétní užití, proto ji bylo třeba implementovat. Díky tomu, že nejprve bylo vytvořeno manuální ovládání vozidla, je tato část již hotova. Konkrétně se jedná o uzel nazvaný *controller_node*, který je patrný z Obr. 40. Opět jde o jednoduchý Python program zpracovávající zprávy z tématu *cmd_vel*, které následně převádí na ovládání jednotlivých kol.

3.8.7 Global planner

Jedná se o plánovač, který pracuje se statickou, již uloženou mapou prostředí, a na základě tzv. costmap určuje nejvhodnější cestu robota k cíli. Jeho výhodou je rychlost reakce, neboť nemusí vytvářet dynamickou mapu. Na druhou stranu nevýhodou je jeho neschopnost reagovat na dynamické překážky [24]. Hierarchie návrhu trasy je většinou taková, že global_planner navrhne trasu na základě statické costmap, které se local_planner snaží držet, ale v případě, že narazí na dynamickou překážku, může tuto trasu částečně opustit. Rozdíl mezi těmito dvěma plánovači je tedy především v okně, ve kterém pracují. Zatímco globální plánovač pracuje s celou mapou, lokální má k dispozici pouze aktuální pozorování ze snímačů, tedy tzv. dynamické okno.

3.8.8 Local planner

V případě, že by byla zajištěna nepřítomnost dynamických překážek, nemusel by být tento plánovač vůbec použit, neboť by celá trasa od startu k cíli mohla být řízena pouze globálním plánovačem, který má výrazně vyšší dosah. V navigačním balíčku ROS lze pomocí textových souborů nakonfigurovat, jaký dosah má lokální plánovač mít, nicméně ten je omezen dosahem lidaru snímajícího okolí.

V konkrétním případě byl použit tzv. DWA lokální plánovač. Tato zkratka znamená dynamic window approach. Jeho úkolem je vytvářet příkazy pro rychlost ve třech souřadnicích $\{x, y, \theta\}$. Algoritmus pracuje s „cenami“ jednotlivých tras, přičemž překážky v trase mají výrazně vyšší cenu, než volný prostor. Hierarchie tohoto procesu je následující [4]:

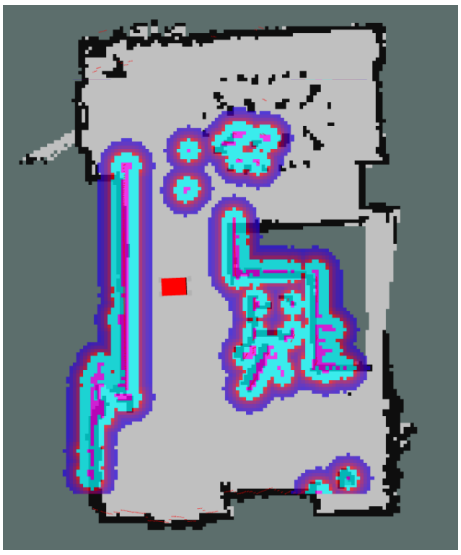
1. Vzorkování stavového prostoru robota v souřadnicích $\{\Delta x, \Delta y, \Delta\theta\}$
2. Provedení simulace jednotlivých navržených tras v krátkém čase
3. Vyřazení kolizních tras, ohodnocení jednotlivých navržených tras skóre
4. Výběr nejlepší trasy, generování příkazů pro řízení motorů
5. Opakování procesu

3.8.9 Local costmap

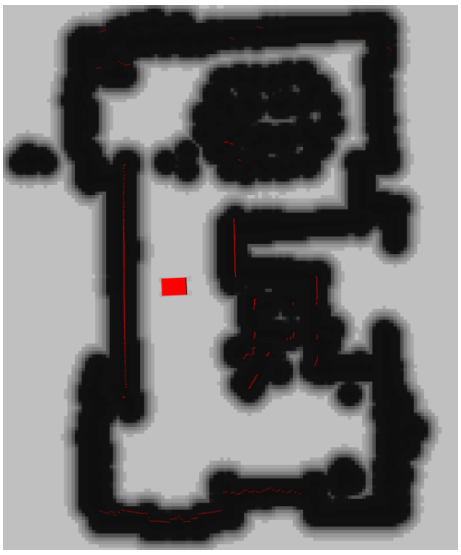
Tato mapa vzniká z aktuálních dat lidaru a pomáhá algoritmu plánování určit místa v prostoru, která jsou obsazena překážkami, a která jsou volná. Tato mapa respektuje rozměry robota a také žádanou vzdálenost - tzv. inflation radius, ve které se má robot pohybovat od překážky. Na základě těchto dat vytvoří tzv. local costmap. Zbarvení patrné na Obr. 50 vyjadřuje „cenu“, kterou stojí projetí danou barvou. Navigační algoritmus se snaží vytvořit vždy takovou trasu, aby byla tato „cena“ minimalizována [4]. Tato mapa vzniká tak, že aktuálně pozorované překážky jsou rozšířeny o daný rozměr a následně v ekvidistantních oblastech ohodnoceny cenami, které stojí projetí danou oblastí. Pochopitelně je vždy žádoucí projet co nejdále od překážky, ale zároveň minimalizovat celkovou ujetou dráhu.

3.8.10 Global costmap

Rozdíl mezi local costmap a global costmap je stejný jako mezi lokálním a globálním plánovačem. Tedy tato mapa je vytvořena již na začátku celého procesu navigace a je v čase neměnná, neboť je vytvořena ze statické mapy. Černé oblasti reprezentující překážky jsou opět zvětšeny o tzv. inflation radius. Výhodou této mapy je její nenáročnost na procesor počítače, neboť ji není třeba v čase cyklicky aktualizovat, na rozdíl od local costmap.



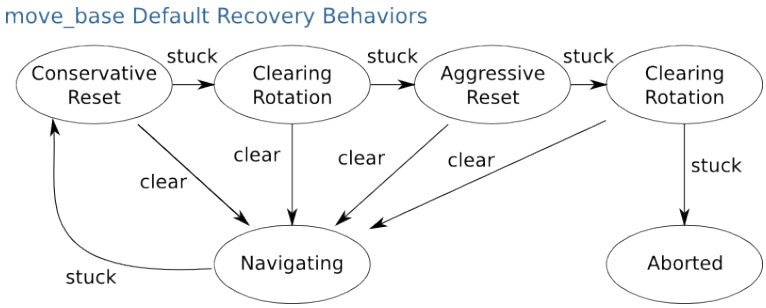
Obr. 50: Local costmap



Obr. 51: Global costmap

3.8.11 Recovery behaviors

Poslední doposud nepopsanou částí navigace a plánování je tzv. recovery behavior, neboli jak se má zachovat robot v případě, že se dostane do situace, kdy nemůže pokračovat v cestě v důsledku zaseknutí se o překážku. Výchozí nastavení je rotace o 360 °, která by měla robota uvolnit. V případě, že k uvolnění nedojde, dochází k dalším pokusům o pokračování v jízdě, jak je znázorněno na Obr. 52.



Obr. 52: Stavový diagram chování při zaseknutí [4]

3.9 Experiment

Posledním krokem této práce bylo ověřit funkčnost autonomní jízdy vozidla. Vozidlo bylo umístěno do předem zmapované místnosti a následně spuštěn program v ROS, který se stará o lokalizaci, plánování cesty, a řízení motorů. Cílem bylo, aby vozidlo dojelo do stanoveného bodu s přiměřenou tolerancí polohy $\{x, y, \theta\}$. Dále bylo testováno, zdali se model během jízdy dokáže vyhnout překážce, která není součástí statické mapy, čímž bude ověřena funkčnost lokálního plánovače trasy.

3.9.1 RViz

Jak již bylo vysvětleno v kapitole 3.6.2, jde o grafickou nástavbu software ROS umožňující vizualizovat množství dat. V tomto případě byla zobrazena statická mapa, global a local costmap, poté model vozidla společně se souřadnými systémy. Dalším bodem zájmu byla trajektorie generovaná lokálním a globálním plánovačem. Posledním sledovaným parametrem byl výstup z lidaru, který umožnil zhodnotit kvalitu lokalizace. V ideálním případě by měly jednotlivé body reprezentující odrazy od překážek mít v souřadném systému mapy pevnou polohu.

3.9.2 Odometrie

Velmi důležitou součástí lokalizace robota je určení polohy a rychosti na základě odometrie, resp. modelu z [40]. Rychlost vozidla byla změřena na 18 cm/s.

Pro ověření kvality odometrie byly porovnány vzdálenosti určené z modelu v ROS a skutečně ujetá vzdálenost. Naměřená data jsou uvedena v tabulce níže.

Tab. 5: Porovnání vypočtených a skutečných vzdáleností

Směr	ROS [cm]	Realita [cm]	Rozdíl [%]
X	94	98	4,1
Y	96	95	1,1

Poslední souřadnicí, která z důvodu nedostatku vybavení umožňujícího změření skutečného úhlu určena, byl úhel natočení θ. Lze konstatovat, že při natočení v realitě o cca 90 ° podobnou hodnotu ukazoval i software ROS.

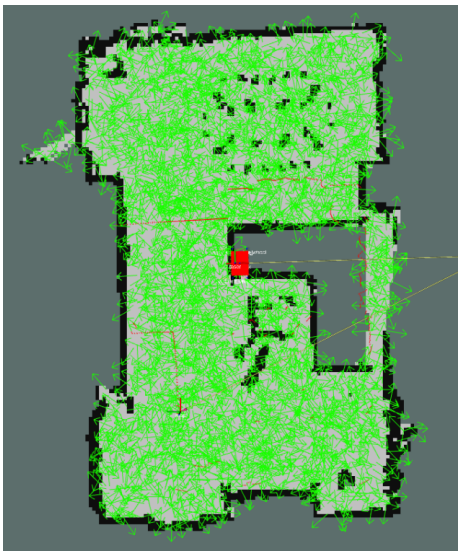
Je patrné, že odchylka mezi realitou a vypočítanou vzdáleností je zejména v ose X (předozadní) poměrně velká. Aby se tato chyba příliš neprojevila ve statické mapě, je využíván lokalizační balíček *amcl* využívající adaptivní Monte Carlo lokalizaci, jejíž snahou je vypočítat transformaci mezi souřadným systémem odometrie a mapou, tedy kompenzovat odchylku vzdáleností. I tento algoritmus byl testován v rámci tohoto experimentu a jeho výsledky jsou zmíněny dále.

3.9.3 Lokalizace

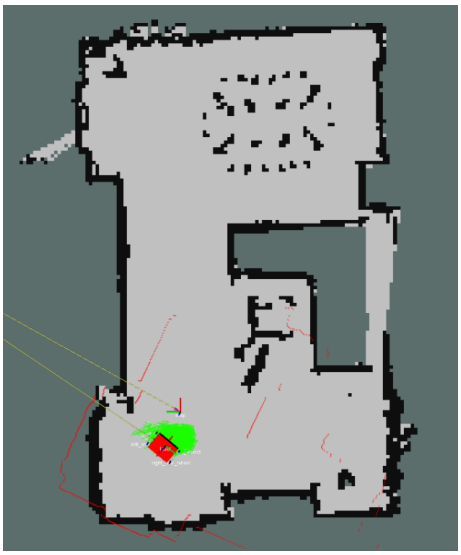
Další testovanou částí byla lokalizace vozidla, tedy určení jeho souřadnic ve statické mapě. Algoritmus umožňuje provést lokalizaci buď při odhadu počáteční polohy, případně lze vynutit lokalizaci globální, která se využije v případě, kdy není možné počáteční polohu ani přibližně určit. V obou případech by měl algoritmus po určité ujeté vzdálenosti určit polohu vozidla s patřičnou odchylkou.

Bez odhadu počáteční polohy

První lokalizační test byl za použití globální lokalizace. Množina možných poloh vozidla v mapě byla rovnoměrná v celém tomto prostoru, což je patrné z Obr. 53. Dále Obr. 54 poté ukazuje, že algoritmus *amcl* byl schopen určit polohu vozidla, neboť zelené šipky reprezentující možnou polohu jsou více koncentrované v jednom místě. Červené body ale ukazují, že natočení neodpovídá realitě, neboť by tyto body měly být zarovnané s černými okraji mapy reprezentující zdi místnosti.



Obr. 53: Odhad polohy na počátku globální lokalizace

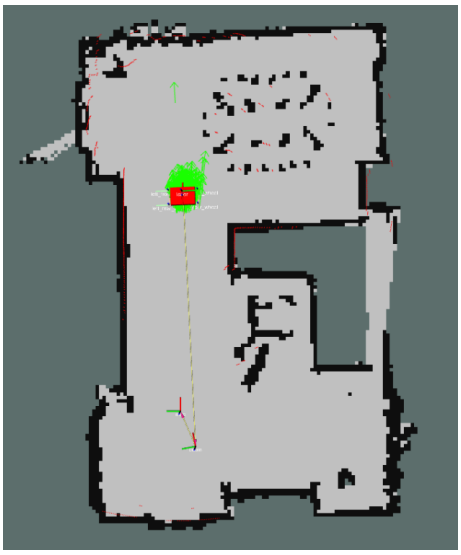


Obr. 54: Odhad polohy z globální lokalizace po cca 5 m

S odhadem počáteční polohy

Při zadání počátečních souřadnic $\{x, y, \theta\}$ je z Obr. 55 zřejmé, že částice jsou již po spuštění tohoto algoritmu koncentrovány v okolí vozidla. Po ujetí přibližně 3 m byl tento algoritmus schopen již poměrně spolehlivě určit polohu vozidla, což je patrné jednak z koncentrace zelených šipek na Obr. 56, jednak ze vzájemné polohy červených bodů a okrajů mapy.

Obr. 55: Odhad polohy na počátku lokalizace s určením výchozí polohy



Obr. 56: Odhad polohy z lokalizace s určením výchozí polohy po cca 3 m

3.9.4 Navigace

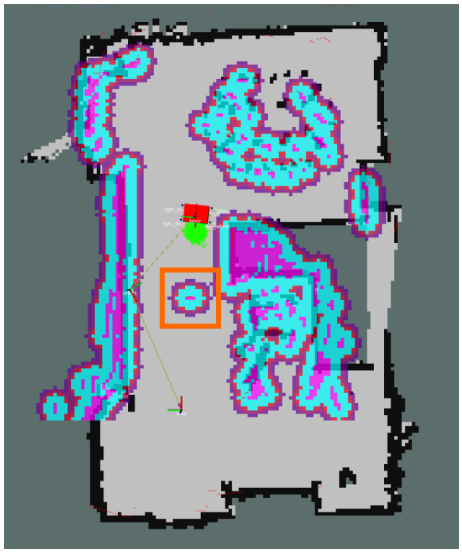
Posledním krokem bylo ověřit schopnost vozidla vyhnout se překážce v režimu autonomní jízdy. Před zahájením autonomní jízdy byla co nejpřesněji provedena lokalizace vozidla ujetím několika metrů v manuálním módu řízení. Když byla lokalizace dostatečně přesná, byla testována samotná navigace. V prostředí RViz k určení cílové pozice slouží tlačítko, jímž se v mapě zvolí konkrétní místo a orientace, kam má vozidlo dojet.

Při prvním pokusu se vozidlo rozjelo směrem dopředu dle naplánované trasy, ale ve chvíli, kdy mělo zahájit jízdu do boku, bylo slyšet pouze bzučení motorů, ale kola se neroztočila. Posléze bylo zjištěno, že za tímto problémem stojí nedostatečný moment pro překonání pasivních odporů kol. Z tohoto důvodu byla v konfiguračním souboru navýšena minimální rychlost vozidla, což tento problém vyřešilo.

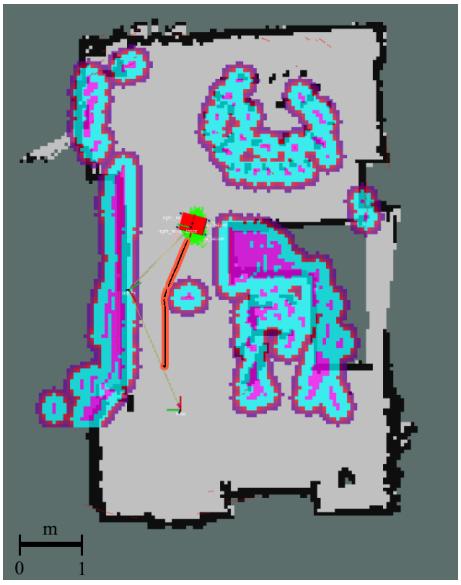
Po vyřešení tohoto problému se již bylo vozidlo schopno dostat do cíle, ale v tomto případě nastal problém s dosažením žádané orientace θ. Vozidlo se začalo otáčet na místě ve snaze dosáhnout zadaného úhlu. Problémem byla výchozí hodnota úhlové tolerance pro úspěšné dosažení cíle, která činila 0,05 rad, tedy asi 2,8 °. Vzhledem k nastavené minimální rychlosti otáčení kol a dynamice vozidla nebylo možné takto malou odchylku splnit, proto byla navýšena na 0,2 rad, tedy asi 11,5 °. S tímto nastavením již bylo vozidlo schopno dosáhnout cíle.

Aby byla ověřena funkčnost lokálního plánovače, byla zobrazena local costmap, která reprezentuje aktuální stav prostředí, konkrétněji překážky na základě dat z lidaru. Do dráhy vozidla byla umístěna překážka v podobě PET lahve (viz Obr. 60). Tato překážka je označena oranžovým čtvercem na Obr. 57. Následně byl vybrán na mapě takový cíl, aby byl nedosažitelný přímou drahou z důvodu přítomnosti překážky. Lokální plánovač tuto

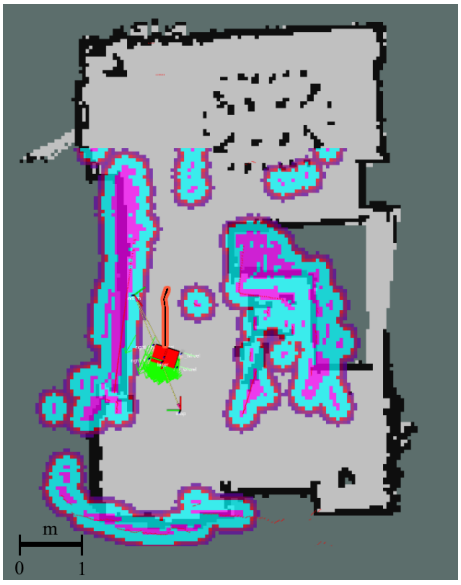
překážku zaregistroval a patřičně upravil trasu tak, aby nedošlo ke kolizi vozidla s PET lahví. Tato trasa je patrná z Obr. 58 (červená křivka). Na základě takto naplánované trasy plánovač vygeneroval příkazy pro pohyb, kterými se vozidlo řídilo, a jak je patrné z Obr. 59 a Obr. 61, bylo schopno se vyhnout překážce v dostatečné vzdálenosti.



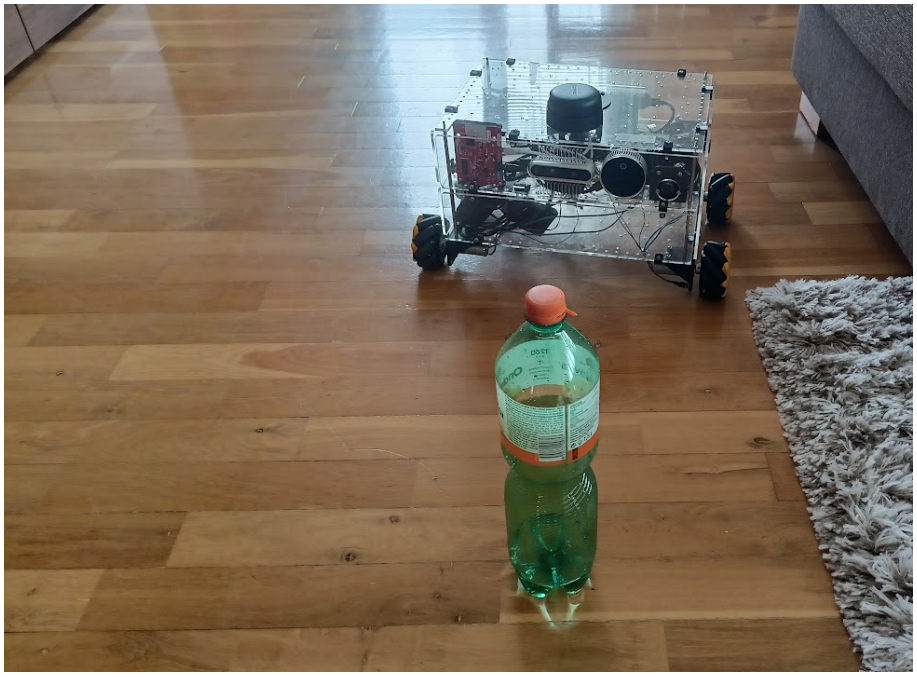
Obr. 57: Zobrazení překážky v local costmap



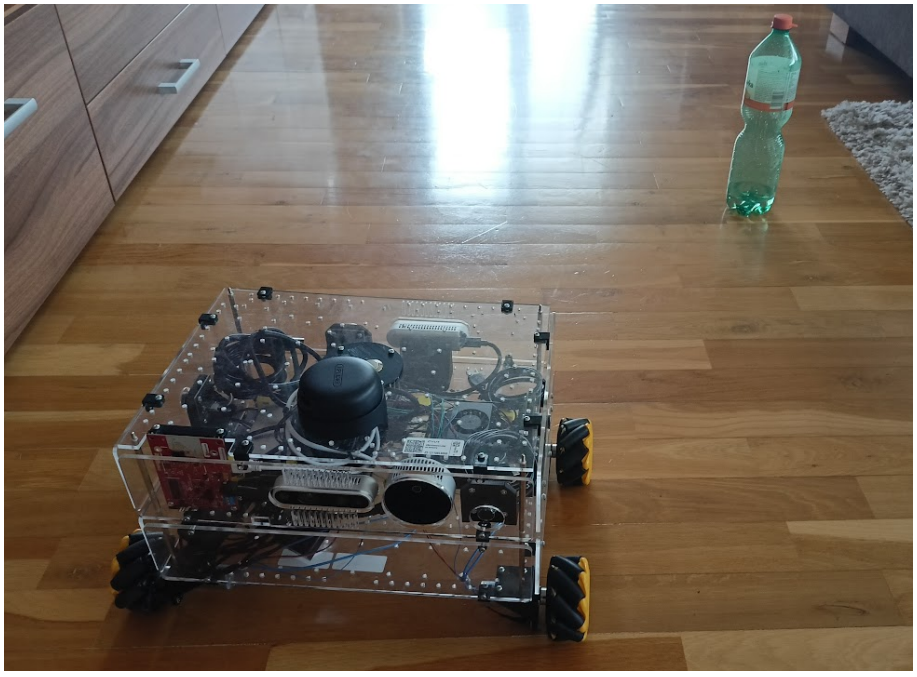
Obr. 58: Naplánovaná trasa před vyhnutím se překážce



Obr. 59: Naplánovaná trasa po vyhnutí se překážce a dosažení cíle



Obr. 60: Model vozidla před vyhnutím se překážce



Obr. 61: Model vozidla po vyhnutí se překážce a dosažení cíle

4 Závěr

Tato diplomová práce popsala proces tvorby modelu autonomního vozidla od výběru snímačů, přes návrh mechanické konstrukce, až po zprovoznění navigace umožňující autonomní jízdu s vyhnutím se překážkám. V úvodu teoretické části bylo popsáno, jaké snímače se používají v reálných autonomních vozidlech, jejich výhody a nevýhody. Dále byly popsány komerčně prodávané produkty. Tato část byla zároveň inspirací při výběru jednotlivých komponent pro sestavení vlastního modelu autonomního vozidla.

Poté, co byly vybrány všechny komponenty, bylo třeba vyřešit mechanickou stránku tohoto vozidla. Zajímavostí je použití tzv. mecanum kol umožňujících platformě všesměrový pohyb. Byl vytvořen 3D model v CAD software, do nějž byly staženy jednotlivé modely snímačů a ostatních komponent, aby vznikla představa o rozměrech konstrukce. Požadavkem bylo, aby model sloužil i jako vzdělávací pomůcka. Z tohoto důvodu byl materiálem vybraným pro konstrukci transparentní PMMA. Jednotlivé desky použité v konstrukci jsou laserové výpalky, ostatní díly byly vytištěny na FDM 3D tiskárně z materiálu PLA.

Poměrně problematické se ukázalo snímání otáček kol, neboť povrch mecanum kol vykazoval natolik velkou odrazivost, že nebylo možné pomocí IR snímače detekovat pulzy k určení rychlosti a natočení kol. Nakonec byl problém vyřešen nanesením černé matné barvy na povrch kol, která způsobila výrazné snížení odrazivosti tohoto povrchu. Pulzy byly vytvořeny pomocí kontrastních plošek realizovaných stříbrnou barvou.

Dalším krokem bylo sestavit vozidlo jako celek, s čímž souviselo zprovoznění řídicího PC, a posléze použitých snímačů. Z vyrobených desek bylo sestaveno vozidlo, které bylo osazeno snímači, počítačem, a ostatními prvky. Dále bylo navrženo a realizováno zapojení jednotlivých prvků za použití kabelového svazku vlastní výroby.

V další části byl vybrán vhodný software, kterým se stal program ROS přímo určený k vývoji robotických platforem. Obsahuje celou řadu již implementovaných algoritmů, a je open-source. Na základě kinematického modelu vozidla s mecanum koly byl vytvořen v prostředí tohoto SW program pro manuální řízení modelu vozidla. Aby bylo možno vyhodnocovat polohu vozidla v prostoru z kinematického modelu, bylo třeba získávat data z optických enkodérů. Tato data byla následně transformována na změnu polohy. Tím byla vytvořena zpětná vazba o poloze vozidla.

Po realizování těchto kroků byla vytvořena mapa místnosti, ve které se následně vozidlo mělo pohybovat. Tato mapa byla vytvořena pomocí 2D lidaru. Pro následnou orientaci v mapě byla využita adaptivní Monte Carlo lokalizace.

Další částí byl experiment zabývající se několika aspekty. První testovanou částí byla odometrie. V této části byla zjištěna poměrně velká odchylka (asi 4 %) mezi očekávanou a skutečnou polohou vozidla. Zároveň byla změřena maximální rychlost tohoto vozidla

$v = 18\ cm/s$. Druhým cílem experimentu bylo ověřit funkčnost lokalizace vozidla v mapě pomocí algoritmu adaptivní Monte Carlo lokalizace. Nejprve byla testována globální lokalizace bez udání počáteční polohy vozidla. Po několika ujetých metrech byl algoritmus schopen určit polohu vozidla, ovšem s nevyhovující přesností natočení. Při druhém kroku lokalizace byla zadána očekávaná počáteční poloha vozidla. V tomto případě byl algoritmus schopen vozidlo lokalizovat již poměrně přesně včetně natočení. Poslední částí experimentu bylo otestovat navigaci umožňující autonomní jízdu. Při tomto testu se objevilo několik problémů týkajících se dynamiky modelu znemožňujících jízdu. Ty se povedlo vyřešit přenastavením konfiguračních parametrů. Po zadání cílového bodu se vozidlo bylo schopno překážce vyhnout a dojet do očekávaného místa.

Pro zlepšení odometrie by bylo vhodné použít algoritmus sensor-fusion a využít data nejen z enkodérů kol, ale také z IMU. Po stránce software by bylo možné využít sofistikovanější plánovací algoritmy pro vytvoření více optimalizovaných tras. Dále by bylo vhodné přepracovat konstrukci kol, neboť vlivem hmotnosti vozidla se jejich držáky prohnuly. Toto způsobilo ne zcela hladkou jízdu z důvodu natočení os jednotlivých kol.

Seznam použité literatury a zdrojů

1. *Levels of driving automation* [online]. 2024. [cit. 2024-03-05]. Dostupné z: https://www.synopsys.com/automotive/what-is-autonomous-car.html.
2. *Intel certified Developer Kit,* [online]. 2024. [cit. 2024-04-05]. Dostupné z: https://up-board.org/up-xtreme-i11-up-squared-6000-robotic-development-kits/.
3. *Intel DepthCamera D435i* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://www.intelrealsense.com/depth-camera-d435i/.
4. *ROS - Robot Operating System* [online]. 2024. [cit. 2024-03-05]. Dostupné z: https://ros.org/.
5. *iRobot Education* [online]. 2024. [cit. 2024-03-05]. Dostupné z: https://edu.irobot.com/what-we-offer/create3.
6. *iRobot Create 3* [online]. 2024. [cit. 2024-03-05]. Dostupné z: https://edu.irobot.com/shop/coding-robots/create?variant=269697.
7. *Yahboom ROSMaster X3* [online]. 2024. [cit. 2024-03-05]. Dostupné z: https://category.yahboom.net/collections/ros-series/products/rosmaster-x3?variant=39684316594260.
8. *Nvidia Robotics and Edge Computing* [online]. 2024. [cit. 2024-03-05]. Dostupné z: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetbot-ai-robot-kit/.
9. *SparkFun JetBot AI Kit* [online]. 2024. [cit. 2024-03-05]. Dostupné z: https://www.sparkfun.com/products/18486.
10. PISAROV, Jelena; MESTER, Gyula. The Future of Autonomous Vehicles. *FME Transactions*. 2020, roč. 49, s. 29–35. Dostupné z DOI: 10.5937/fme2101029P.
11. *Úvod do technologie LiDAR* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://cz.farnell.com/introduction-to-lidar-technology.
12. *Introduction to mmwave Sensing: FMCW Radars* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://www.ti.com/content/dam/videos/external-videos/2/3816841626001/5415528961001.mp4/subassets/mmwaveSensing-FMCW-offlineviewing_0.pdf.
13. *What are the sensors used in self-driving cars?* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://www.engineersgarage.com/what-are-the-sensors-used-in-self-driving-cars/.
14. *Arducam 2Mpx IMX291 USB Low Light 100° Camera Module V2.0* [online]. 2024. [cit. 2021-03-05]. Dostupné z: https://rpishop.cz/usb-kamerove-moduly/3621-arducam-2mpx-imx291-usb-low-light-100-camera-module-v20.html.

15. *RPLIDAR A2* [online]. 2024. [cit. 2024-04-08]. Dostupné z: https://www.slamtec.ai/product/slamtec-rplidar-a2/.
16. *360stupňový laserový skener RPLidar A2M8* [online]. 2024. [cit. 2024-04-09]. Dostupné z: https://botland.cz/stazene-produkty/7036-360stupnovy-laserovy-skener-rplidar-a2m8-5904422361433.html.
17. *Intel Lidar Camera L515* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://www.intelrealsense.com/lidar-camera-l515/.
18. *AWR1642BOOST* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://www.ti.com/tool/AWR1642BOOST.
19. *New Galileo service set to deliver 20 cm accuracy* [online]. 2024. [cit. 2024-04-14]. Dostupné z: https://www.esa.int/Applications/Satellite_navigation/New_Galileo_service_set_to_deliver_20_cm_accuracy.
20. *Raspberry Pi 4 Model B - 8GB RAM* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://rpishop.cz/raspberry-pi-4/2611-raspberry-pi-4-model-b-8gb-ram.html.
21. *NVIDIA Jetson Orin Nano Developer Kit* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://www.sparkfun.com/products/22098.
22. *USB Charger FAQs* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://www.cmd-ltd.com/advice-centre/usb-chargers-and-power-modules/usb-and-power-module-product-help/usb-charger-faqs/.
23. *Waveshare Průmyslový USB HUB, 7x USB 2.0 port* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://rpishop.cz/rozbocovace/5586-waveshare-prumyslovy-usb-hub-7x-usb-20-port.html.
24. LYNCH, Kevin M.; PARK, Frank C. *Modern Robotics: Mechanics, planning, and Control*. University Press, 2021.
25. *Motor JGA25-370 12V s převodovkou* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://www.laskakit.cz/motor-jga25-370-12v-s-prevodovkou-/.
26. NOVÁK, Jaroslav. *Elektromechanické systémy v dopravě a ve strojírenství*. Praha: Vydavatelství ČVUT, 2002. ISBN 80-010-2457-1.
27. *H můstek pro krokový motor L298N Dual H Most DC* [online]. 2021. [cit. 2022-03-11]. Dostupné z: https://dratek.cz/arduino/877-arduino-h-mustek-pro-krokovy-motor-l298n-dual-h-most-dc.html.
28. *PCA9685 16-Channel 12-bit PWM/Servo Driver* [online]. 2024. [cit. 2024-04-07]. Dostupné z: https://electropeak.com/16-channel-12-bit-pwm-servo-driver.

29. *SDK Manager* [online]. 2024. [cit. 2024-04-13]. Dostupné z: https://developer.nvidia.com/sdk-manager.
30. *USB: Port Types and Speeds Compared* [online]. 2024. [cit. 2024-04-13]. Dostupné z: https://tripplite.eaton.com/products/usb-connectivity-types-standards.
31. *Interpreted vs Compiled Programming Languages: What's the Difference?* [online]. 2024. [cit. 2024-05-04]. Dostupné z: https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/.
32. *Jetson Orin Nano Developer Kit Carrier Board* [online]. 2024. [cit. 2024-04-13]. Dostupné z: https://developer.nvidia.com/downloads/assets/embedded/secure/jetson/orin_nano/docs/jetson_orin_nano_devkit_carrier_board_specification_sp.pdf.
33. *Secure Shell* [online]. 2024. [cit. 2024-05-02]. Dostupné z: https://en.wikipedia.org/wiki/Secure_Shell.
34. *The Easiest way to program microcontrollers* [online]. 2024. [cit. 2024-04-24]. Dostupné z: https://circuitpython.org/.
35. *Jak vybrat správnou baterii* [online]. 2024. [cit. 2024-05-13]. Dostupné z: https://www.bighobby.cz/modelarska-poradna/jak-vybrat-spravnou-baterii/.
36. *Li-pol baterie 2250 mAh 2S 25C (50C) Power X6* [online]. 2024. [cit. 2024-05-12]. Dostupné z: https://www.bighobby.cz/li-pol-baterie-2250-mah-2s-25c--50c--power-x6/.
37. *Step-up boost měnič s XL6009, modrá* [online]. 2024. [cit. 2024-05-12]. Dostupné z: https://www.laskakit.cz/step-up-boost-menic-s-xl6009--modra/.
38. *Exchange Data with ROS Publishers and Subscribers* [online]. 2024. [cit. 2024-04-27]. Dostupné z: https://es.mathworks.com/help/ros/ug/exchange-data-with-ros-publishers-and-subscribers.html.
39. *XML Robot Description Format (URDF)* [online]. 2024. [cit. 2024-04-27]. Dostupné z: https://wiki.ros.org/urdf/XML/model.
40. TAHERI Qiao, Ghaeminezhad. Kinematic Model of a Four Mecanum Wheeled Mobile Robot. *International Journal of Computer Applications*. 2015, s. 4.
41. *Coordinate Frames for Mobile Platforms* [online]. 2024. [cit. 2024-05-12]. Dostupné z: https://www.ros.org/reps/rep-0105.html.
42. *Adaptive Monte Carlo Localization* [online]. 2024. [cit. 2024-05-13]. Dostupné z: https://roboticsknowledgebase.com/wiki/state-estimation/adaptive-monte-carlo-localization/.

Seznam obrázků, grafů a tabulek

Seznam obrázků

Seznam grafů

Seznam tabulek

Seznam použitého SW

- Texmaker
- Solidworks 2019
- Microsoft Visual Studio Code
- MS Excel
- Paint.Net
- Draw.IO
- Ultimaker Cura
- ROS
- GIMP