

Návrh a vývoj autonomního vozidla s využitím v automatizaci

Tomáš Horák

Bakalářská práce
2025



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Ústav automatizace a řídicí techniky

Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Tomáš Horák
Osobní číslo:	A22704
Studijní program:	B0714A150006 Aplikovaná informatika v průmyslové automatizaci
Specializace:	Inteligentní systémy s roboty
Forma studia:	Prezenční
Téma práce:	Návrh a vývoj autonomního vozidla s využitím v automatizaci
Téma práce anglicky:	Design and Development of an Autonomous Vehicle for Use in Automation

Zásady pro vypracování

1. Zpracujte literární rešerši na téma autonomní vozidlo s využitím v automatizaci.
2. Analyzujte potřebné požadavky na návrh vozidla.
3. Navrhněte vozidlo s využitím platformy Arduino.
4. Realizujte navržené vozidlo
5. Otestujte funkčnost vozidla.

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

Seznam doporučené literatury:

1. BLUM, Jeremy, 2013. *Exploring Arduino: tools and techniques for engineering wizardry*. Indianapolis: Wiley. ISBN 978-1-118-54936-0.
2. ČERNÝ, David, 2022. *Automatizované řízení vozidel a autonomní doprava: technické a humanitní perspektivy*. Praha: Academia. ISBN 978-80-200-3358-1.
3. HRUŠKA, František, 2005. *Technické prostředky automatizace III : (senzory, jejich principy a funkce)*. 2. Zlín: Univerzita Tomáše Bati ve Zlíně. ISBN 807-31-8315-3.
4. KLOSKI, Liza Wallach a KLOSKI, Nick, 2017. *Začínáme s 3D tiskem*. Brno: Computer Press. ISBN 978-802-5148-761.
5. MARTINEK, Radislav, 2004. *Senzory v průmyslové praxi*. Praha: BEN – technická literatura. ISBN 80-730-0114-4.
6. PRŮŠA, Josef, 2019. *Základy 3D tisku s Josefem Průšou*. Praha: Prusa research. ISBN 978-80-907-7980-8.

Vedoucí bakalářské práce: **Ing. Václav Mach, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **13. prosince 2024**
Termín odevzdání bakalářské práce: **27. května 2025**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan



prof. Ing. Vladimír Vašek, CSc. v.r.
ředitel ústavu

Ve Zlíně dne 13. prosince 2024

PROHLÁŠENÍ AUTORA BAKALÁŘSKÉ PRÁCE

Beru na vědomí, že

- odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému a bude dostupná k nahlédnutí;
- jedno vyhotovení bakalářské práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – bakalářskou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na bakalářské práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.

Ve Zlíně, dne

.....
podpis autora

ABSTRAKT

Autonomní vozidla už dávno nejsou jen vizí budoucnosti – stávají se realitou, která mění způsob, jakým vnímáme dopravu, automatizaci i inteligentní řízení. Tato práce se věnuje návrhu vlastního autonomního vozidla, které se dokáže orientovat v prostoru, rozpoznávat překážky a rozhodovat se zcela bez lidského zásahu. Využitím moderních technologií, jako je vizuální sledování pomocí ArUco markerů, zpracování obrazu v reálném čase a komunikace mezi Python programem a Arduino mikrokontrolérem, vzniká komplexní systém spojující teorii s praktickou realizací. Cílem je ověřit, zda je možné navrhnout a postavit inteligentní autonomní vozidlo, které dokáže reagovat na okolní prostředí s dostatečnou přesností a spolehlivostí.

Klíčová slova: autonomní řízení, autonomní vozidlo, strojové vidění, ArUco, Arduino, automatizace

ABSTRACT

Autonomous vehicles are no longer merely a vision of the future—they are becoming a reality that transforms how we perceive transportation, automation, and intelligent control. This thesis focuses on the design of a custom-built autonomous vehicle capable of spatial navigation, obstacle detection, and independent decision-making without human intervention. By leveraging modern technologies such as visual tracking using ArUco markers, real-time computer vision, and communication between a Python program and an Arduino-based microcontroller, a complex system is created that bridges theoretical concepts with practical implementation. The aim is to explore whether it is possible to design and build an intelligent autonomous vehicle that can respond to its surroundings with sufficient precision and reliability.

Keywords: autonomous driving, autonomous vehicle, machine vision, ArUco, Arduino, automation

Rád bych poděkoval vedoucímu bakalářské práce Ing. Václavu Machovi, Ph.D. za odborné vedení, cenné rady a trpělivost při konzultacích v průběhu zpracování této práce. Dále děkuji své rodině, a to zejména své mamince a přítelkyni za podporu, která mi umožnila se plně věnovat studiu i praktické realizaci práce.

„Autonomní doprava není otázkou technické připravenosti, ale především společenské akceptace.“ – Václav Jirovský

OBSAH

ÚVOD.....	11
I TEORETICKÁ ČÁST	12
1 AUTONOMNÍ VOZIDLO A JEHO ROLE V AUTOMATIZACI.....	13
1.1 ÚROVNĚ AUTONOMNÍCH VOZIDEL	13
1.2 HISTORIE A VÝVOJ AUTONOMNÍCH VOZIDEL	14
1.2.1 Počátky a klíčové milníky autonomních vozidel	15
1.3 AUTONOMNÍ VOZIDLA A JEJICH VĚDOMÍ.....	15
1.4 DRUHY VYUŽITÍ AUTONOMNÍCH VOZIDEL	16
1.5 VÝHODY A RIZIKA AUTONOMNÍCH VOZIDEL	19
1.5.1 Přínosy.....	19
1.5.2 Rizika	20
2 KOMPONENTY AUTONOMNÍHO VOZIDLA.....	21
2.1 SENZORY A JEJICH FUNKCE.....	21
2.2 KAMEROVÉ SYSTÉMY	22
2.2.1 Druhy kamer.....	22
2.3 ELEKTRICKÉ POHONY	24
3 ARDUINO A JEHO VYUŽITÍ V AUTOMATIZACI.....	28
3.1 DRUHY ARDUINO DESEK	28
3.2 VÝVOJOVÉ PROSTŘEDÍ ARDUINO	32
3.3 OVLÁDÁNÍ A KOMUNIKACE POMOCÍ WeMos D1 MINI (ESP8266).....	35
II PRAKTICKÁ ČÁST	36
4 NÁVRH AUTONOMNÍHO VOZIDLA.....	37
4.1 VYBRANÉ KOMPONENTY	39
4.2 MONTÁŽ A KONSTRUKCE VOZIDLA.....	42
4.3 SCHÉMA ZAPOJENÍ.....	43
5 PROGRAMOVÉ VYBAVENÍ VOZIDLA	46
5.1 ARDUINO KÓD	46
5.2 PYTHON KÓD	50
5.3 KAMEROVÝ SYSTÉM.....	57
5.3.1 Umístění a konfigurace kamery	57
5.3.2 Význam horního pohledu pro navigaci	58
6 NÁVRH EXPERIMENTU	59
6.1 ARUCO MARKERY	60
6.2 PŘEKÁŽKY.....	61

6.3	ALGORITMUS.....	62
6.4	PLÁNOVANÉ TESTOVACÍ SCÉNÁŘE.....	62
7	TESTOVÁNÍ A VYHODNOCENÍ FUNKČNOSTI	64
7.1	SCÉNÁŘE TESTOVÁNÍ.....	64
7.2	VÝSLEDKY TESTOVÁNÍ.....	67
7.3	IDENTIFIKACE PROBLÉMŮ A JEJICH ŘEŠENÍ	68
	ZÁVĚR.....	70
	SEZNAM POUŽITÉ LITERATURY	71
	SEZNAM OBRÁZKŮ	78
	SEZNAM TABULEK.....	80
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	81
	SEZNAM PŘÍLOH.....	82

ÚVOD

V posledních letech dochází k výraznému pokroku v oblasti autonomního řízení, což se projevuje nejen v komerčním sektoru, ale také v akademickém prostředí. Autonomní vozidla, která byla dříve považována za futuristickou vizi, se postupně stávají realitou díky dostupnosti pokročilých technologií a otevřených vývojových platforem. Tento trend umožňuje realizaci projektů autonomního řízení i v rámci studentských prací, a to s využitím běžně dostupných komponent.

Cílem této bakalářské práce je návrh a realizace funkčního modelu autonomního vozidla schopného samostatného pohybu v definovaném prostoru. Model je sestaven z dostupného stavebnicového kitu a řízen mikrokontrolérem Arduino. Řídící algoritmy jsou implementovány v programovacím jazyce Python, přičemž zpracování obrazu z horní kamery umožňuje detekci pozice a orientace vozidla pomocí ArUco markerů. Komunikace mezi počítačem a mikrokontrolérem probíhá prostřednictvím protokolu WebSocket, což umožňuje reálné řízení pohybu vozidla.

Práce se zaměřuje na integraci různých technologií a jejich vzájemnou komunikaci s cílem vytvořit systém schopný autonomní navigace bez nutnosti přímého zásahu člověka. Důraz je kladen na využití cenově dostupných komponent a otevřených softwarových nástrojů, což demonstruje možnosti realizace takového systému i s omezeným rozpočtem.

Úvodní část bakalářské práce poskytuje obecný přehled autonomních systémů a technologií využívaných v této oblasti. Následně je podrobně popsán návrh a konstrukce modelu vozidla, včetně principu fungování kamery a metod zpracování obrazu. Závěrečná část se zaměřuje na testování systému v různých scénářích a vyhodnocení jeho chování, identifikaci funkčních aspektů a oblastí vyžadujících další optimalizaci.

I TEORETICKÁ ČÁST

1 AUTONOMNÍ VOZIDLO A JEHO ROLE V AUTOMATIZACI

Na úvod je potřeba definovat pojem *autonomní vozidlo* [1]. Pod pojmem autonomní nebo robotické vozidlo si lze představit technologii, která umožní vozidlu samostatně vnímat své okolí, zpracovávat získané informace o okolním prostředí a následně se pohybovat bez nutnosti zásahu člověka.

Ministerstvo dopravy definuje automatizované vozidlo jako „vozidlo, které díky pokročilým systémům a funkcím dokáže za stanovených podmínek jezdit autonomně, tedy bez zásahů řidiče.“ [2]

Dle autora [3] jsou za fascinující vynález označována vozidla, která automaticky přepravují lidi nebo věci z bodu A do bodu B, případně vykonávají určitou obecně prospěšnou činnost. Na druhou stranu dnes představují běžnou součást života, kterou již většina lidí nevnímá jako něco mimořádného. Autonomní vozidla lze rozdělit do dvou skupin. První skupina zahrnuje veškeré lidskou myslí představitelné stroje. Naopak do druhé skupiny patří roboti ve skladech a výrobních halách, metro, kolejová vozidla nebo domácí robotický vysavač. Na rozdíl od autonomních vozidel pohybujících se v běžném silničním provozu, jak jsou často k vidění v médiích, mají tyto stroje jednu obrovskou výhodu – pohybují se v prostředích, která jsou do velké míry předvídatelná a uzavřená. Když se něco pokazí, následky bývají malé, většinou jde jen o materiální škody. Fungují tedy v prostředích, kde je vše plánované a řízené, jejich provoz je centrálně koordinován a každá cesta je předem známá. Autonomní vozidla, která jsou dnes prezentována jako budoucnost dopravy, však čelí mnohem větší výzvě. Pohybují se totiž v prostředí, které je chaotické, nepředvídatelné a neustále se mění. Na rozdíl od systémů s pevně stanovenými pravidly a předvídatelným chováním, jako je metro nebo skladištní robotika, musí autonomní vozidla reagovat na nečekané situace – ať už jde o nepředvídatelné chování ostatních účastníků dopravy, měnící se povětrnostní podmínky nebo technické problémy.

1.1 Úrovně autonomních vozidel

Autonomní vozidla lze diferencovat do šesti úrovní [1]: vozidlo bez automatizace, vozidlo s asistencí řidiče, vozidlo s částečnou automatizací, vozidlo s podmíněnou automatizací, vozidlo s vysokou automatizací a vozidlo s plnou automatizací.

První úroveň zastupují vozidla bez automatizace, nejsou zde přítomné žádné asistenční systémy, které by mohly ovlivňovat řízení vozidla [4]. Vozidlo má pouze základní informativní kontrolky.

V druhé úrovni může vozidlo s asistencí řidiče obsahovat několik systémů, avšak ty mezi sebou nefungují propojeně a nejsou vzájemně kombinovány. Objevují se zde první systémy, které pomáhají udržet vozidlo v podélném nebo příčném směru, například adaptivní tempomat [5]. Tyto systémy při řízení pomáhají, ale za ovládání vozidla je stále zodpovědný řidič.

V rámci třetí úrovně je vozidlo schopné samo ovládat zrychlování, zpomalování i zatáčení. Řidič však musí mít ruce stále na volantu a sledovat okolnosti. Tento systém je výhodný především na rychlostních silnicích nebo dálnicích [4]. Vozidla s částečnou automatizací oproti vozidlům s asistencí řidiče mohou kombinovat různé funkce, například točení volantem u zpomalování [6].

Čtvrtou úrovní si vozidlo s podmíněnou automatizací přebírá kontrolu nad řízením, a to za určitých podmínek. Jako příklad lze uvést dálniční kolonu. Řidič se může věnovat jiným činnostem, ale musí být připraven převzít kontrolu nad vozidlem, pokud systém vyhodnotí, že to situace vyžaduje [7]. Vozidlo samostatně zrychluje, brzdí a vyhýbá se překážkám.

Pátá úroveň je charakteristická vysokou automatizací. Vozidlo je schopno zvládat řízení plně autonomně v určitých situacích, například při parkování. Řidič nemusí zasahovat do kontroly vozidla, protože dané vozidlo je schopné samo zaparkovat [4]. Tato úroveň však stále vyžaduje jasně definované prostředí a řidič může systém zapnout, je-li to bezpečné [7].

V šesté úrovni je vozidlo plně automatizované a zvládne všechny jízdní situace autonomně, bez jakékoliv pomoci řidiče. Taková vozidla nevyžadují volant ani jiné ovládací prvky [5]. Typickým příkladem je robotické taxi, které přijede a odveze cestující bez přítomnosti řidiče.

1.2 Historie a vývoj autonomních vozidel

Koncept autonomních vozidel sahá hluboko do historie. Už ve 20. letech 20. století se objevily první experimentální prototypy, které dokázaly jezdit bez přímého lidského zásahu, přičemž k jejich ovládání bylo využíváno rádiového řízení. Tyto systémy měly značná omezení a nebyly schopny adekvátně reagovat na složitější dopravní situace [8]. Pro

historický kontext bude pozornost zaměřena na počátky a technologické pokroky, které vedly k rozvoji samostatně řízených vozidel.

1.2.1 Počátky a klíčové milníky autonomních vozidel

Myšlenka vozidel schopných samostatné jízdy nebyla v tehdejší technologické kontextu považována za nereálnou. Důkazem toho byly experimenty organizace Radio Corporation of America (RCA), která v roce 1957 úspěšně otestovala autonomní pohyb vozidla na úseku dlouhém 120 metrů poblíž města Lincoln v Nebrasce. Další testování probíhalo i na okruhu u města Princeton v New Jersey, kde se pohybovalo více autonomních vozidel současně s vozidly ovládanými řidiči. Testovací vozidla byla vybavena přijímačem a malými anténami umístěnými na obou stranách předního nárazníku [9]. Pod vozovkou byl veden elektrický kabel, který byl propojený s elektrickým obvodem umístěným na kraji silnice. Při průjezdu vozidla docházelo ke změně proudu v obvodu, což umožňovalo nejen řízení vozidla, ale i ovládání frekvencí – jedna frekvence udržovala vozidlo v jízdním pruhu, další upozorňovala na překážky na silnici a jiná informovala řidiče prostřednictvím rádia o blížícím se sjezdu z elektrifikované dálnice, bylo tedy nutné řízení převzít manuálně.

K výraznému posunu v oblasti autonomní mobility došlo až v 80. letech s rozvojem počítačového vidění a prvních algoritmů strojového učení [8]. V průběhu 90. let došlo k akceleraci technologického pokroku v oblasti autonomního řízení, a to především díky rostoucímu výpočetnímu výkonu a zlepšující se dostupnosti senzorických systémů, jako jsou kamery a GPS.

1.3 Autonomní vozidla a jejich vědomí

Technologie autonomních vozidel se stále více přibližuje plnému nahrazení řidiče pokročilými systémy schopnými samostatného rozhodování na vysoké úrovni. Aby bylo možné předcházet potenciálním problémům před jejich vznikem, musí tyto systémy rozumět běžnému chování vozidla a dokázat ho predikovat. Autonomní systém by měl být schopen průběžně se adaptovat na dynamicky se měnící prostředí a zároveň se učit řešit nové, doposud neznámé situace [10]. Vzhledem k tomu, že taková prostředí nejsou vždy dostupná pro trénování, je nezbytné, aby systém měl přehled o svých vlastních schopnostech a limitech, což je jev označovaný jako sebeuvědomění nebo ovládání vědomí.

Autonomní vozidla fungují díky percepčním systémům, které umožňují vnímat a analyzovat své okolí. Tyto systémy využívají informace získané z různých senzorů, aby

vozidlo mohlo vnímat své prostředí a na základě těchto vjemů přijímat rozhodnutí [11]. Senzorům je věnována samostatná podkapitola 2.1 *Senzory a jejich funkce*.

1.4 Druhy využití autonomních vozidel

Pokrok v technologii automatizovaných a autonomních vozidel je mnohem rychlejší, než by se dalo očekávat. Dnes už koncept samořídících vozidel není omezen pouze na osobní přepravu, ale zahrnuje i komerční aplikace, které jsou již blíže k realizaci než ty pro každodenní použití. Tato technologie, která mění automobilový průmysl, nachází své uplatnění v praxi na našich ulicích, ve skladech, na farmách, nebo dokonce i v letecké dopravě [12]. Kromě veřejně dostupných aplikací existují i specifické případy využití v různých odvětvích.

Mezi zásadní trendy moderní chytré mobility v osobní dopravě patří využití internetu věcí (IoT). Internet věcí je technologie, která přináší výrazné inovace v oblasti cestování, ať už jde o efektivnější veřejnou dopravu, rozvoj autonomních vozidel nebo inteligentní parkovací systémy [13]. Současná logistika a nákladní doprava jsou na těchto řešeních již závislé, což dokazuje, jak významný dopad má IoT na způsob, jakým se dnes pohybujeme a přepravujeme.

Autonomní osobní doprava

Koncept známý jako Mobility-as-a-Service (MaaS), neboli mobilita jako služba, propojuje různé dopravní možnosti do jednotné platformy, kterou lze snadno ovládat prostřednictvím mobilních aplikací. Díky tomu mohou uživatelé efektivně kombinovat taxi, sdílená vozidla a další dopravní prostředky podle svých aktuálních potřeb, což přispívá k vyšší flexibilitě a pohodlnějšímu cestování (Obrázek 1) [13]. Tento přístup nejenže zjednodušuje přepravu, ale také podporuje udržitelnější využívání dopravních prostředků v městském prostředí.

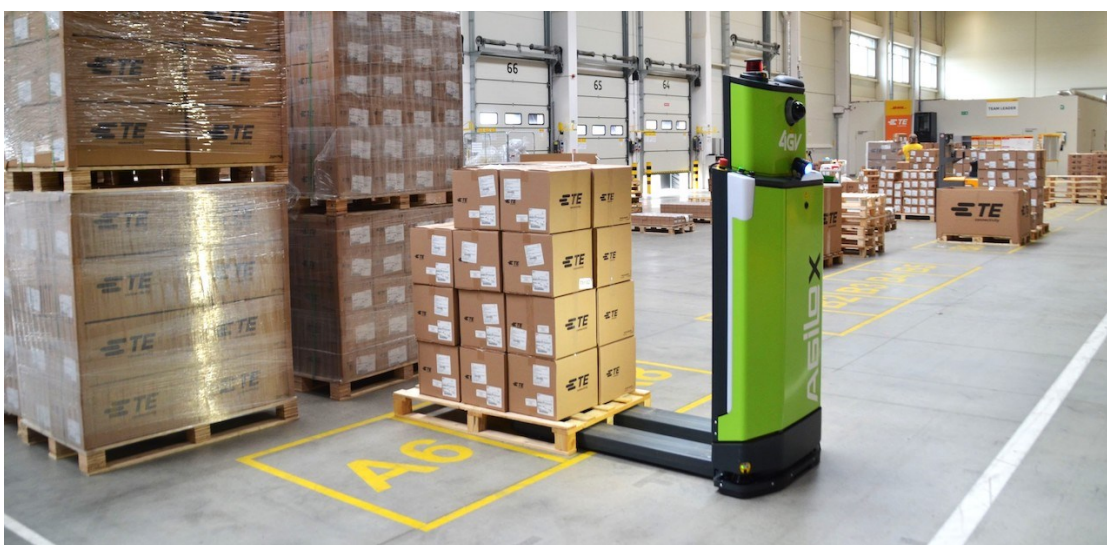


Obrázek 1: Autonomní autobus pro 15 osob [14]

Autonomní logistika a přeprava zboží

Digitalizace procesů představuje jednu z nejzásadnějších změn v oblasti logistiky, jelikož umožňuje efektivnější správu dodavatelských řetězců. Technologie jako internet věcí nebo umělá inteligence hrají hlavní roli při zvyšování transparentnosti a sledování zásob. Díky senzorům a chytrým RFID štítkům lze sledovat pohyb zboží v reálném čase, což napomáhá optimalizaci. Automatizace skladových a logistických center se dynamicky rozvíjí, protože robotické systémy dokáží manipulovat s nákladem efektivněji než lidská pracovní síla a snižují provozní náklady [15]. Automatizované skladovací technologie urychlují proces vyřizování objednávek, a zároveň zvyšují přesnost doručení, což je klíčové v prostředí, kde e-commerce (neboli elektronické obchodování) klade vysoké nároky na rychlost a kvalitu služeb (Obrázek 2).

Jan Flodрман z firmy Zebra Technologier Europe popsal „Firmy budou flexibilnější a lépe přizpůsobitelné dynamickým požadavkům zákazníků. Celkově automatizace zlepší efektivitu a konkurenceschopnost logistických operací a dodavatelským řetězcům umožní rychleji reagovat na změny na trhu“ [16].



Obrázek 2: Autonomní manipulační vozík [17]

Autonomní vozidla v průmyslu a zemědělství

Se změnou metod průmyslové výroby, kdy část tradiční pásové výroby ustupuje ostrovním pracovním stanicím, se objevují nové požadavky na intralogistiku a využívání autonomních vozítek. V prostředí továren, kde jsou přítomni lidé, je kladen důraz na vysokou úroveň bezpečnosti a prevenci kolizí. Antikolizní systémy se stávají nezbytnými nejen pro ochranu lidského zdraví, ale i majetku a pro prevenci střetů mezi vozítky [18]. Do budoucna se

očekává rostoucí míra využívání kolaborativních robotů, kteří se budou pohybovat na autonomních vozítkách a přemísťovat se mezi pracovními stanicemi podle aktuálních potřeb.

V sektoru stavebnictví a těžby přispívají autonomní vozidla ke zlepšení přepravy a manipulace s materiálem, přičemž se osvědčují v náročných provozních podmínkách. Tato vozidla jsou navržena tak, aby zvládla náročné úkoly vyžadující vysokou míru spolehlivosti a odolnosti – například přepravu těžkých nákladů v komplikovaném terénu nebo realizaci přesných zemních prací v extrémních podmínkách. V zemědělství představují autonomní traktory a sklízecí stroje zásadní změnu v přístupech k výsadbě a sklizni (Obrázek 3). Tyto moderní technologie umožňují dosahovat konzistentně vysokých výnosů díky pokročilým systémům sledujícím stav vegetace a zefektivňujícím procesy související s výsadbou a sklizní [19]. Nasazení precizních metod tak přispívá nejen ke zvýšení produktivity, ale i k lepšímu využití dostupných zdrojů.



Obrázek 3: Autonomní a plně elektrický traktor [20]

Autonomní vojenská vozidla

Robotická vozidla se širokou škálou funkcí mohou v moderních ozbrojených silách zastávat řadu klíčových úkolů. Mezi perspektivní technologie budoucnosti patří také pozemní autonomní vozidla, která jsou navržena nejen jako podpora pěchotních jednotek, ale i pro další vojenské operace. Tato vozidla usnadní přepravu materiálu a evakuaci zraněných a mohou být vybavena jak bojovými systémy, tak průzkumnými senzory. Vývoji autonomních vojenských prostředků se v posledních letech věnuje mnoho společností,

včetně českých firem [21]. Příkladem domácího řešení je robotické vozidlo TAROS, které již bylo testováno Armádou České republiky (Obrázek 4).



Obrázek 4: TAROS 6x6 [22]

1.5 Výhody a rizika autonomních vozidel

Hlavní motivací pro další podporu výzkumu, vývoje a následného zavádění autonomních vozidel do provozu je především zvýšení bezpečnosti. Snížení počtu dopravních nehod je považováno za jeden z nejvýznamnějších přínosů autonomního řízení. Až 95 % dopravních nehod je způsobeno nebo spoluzaviněno lidskou chybou. Postupné nasazování vozidel se automatizovanými systémy řízení na silnicích by mohlo zvýšit celkovou bezpečnost provozu a zajistit plynulejší dopravu. Autonomní vozidla navíc zvyšují komfort řidičů, protože část řidičských úkonů, a v budoucnu možná všechny, převzme autonomní systém místo člověka [23]. Toto usnadnění umožňuje efektivněji strávený čas cestováním. Dalším přínosem autonomní mobility je zpřístupnění dopravy pro osoby se zdravotním postižením, které v současné době nemohou řídit.

1.5.1 Přínosy

Autonomní minibusy jako součást efektivně organizované veřejné dopravy, schopné obsluhovat oblasti s nepravidelnou poptávkou a vytížením, si cestující budou moci objednat podle aktuální potřeby. Klíčové je identifikovat přínosy v konkrétních lokalitách a hledat vhodné způsoby začlenění této technologie do současných dopravních systémů, včetně spolupráce mezi veřejným a soukromým sektorem [23]. Podobné iniciativy již probíhají

v mnoha městech, nejen ve velkých jako Praha, Brno nebo Ostrava, ale i například v Ústí nad Labem, Plzni a dalších.

1.5.2 Rizika

Hlavní příčiny nehod autonomních vozidel a jejich rizika lze rozdělit do několika kategorií. Prvním a stále největším problémem je lidský faktor. I když jsou autonomní systémy stále vyspělejší, někteří řidiči stále přehlíží jejich omezení. Spoléhají na automatiku více, než by měli, a přestávají věnovat pozornost řízení. Na druhé straně existují situace, které mohou autonomní vozidlo překvapit, například nečekané chování chodců, cyklistů nebo jiných řidičů, které technologie není schopna vždy správně vyhodnotit. Tyto nepředvídatelné momenty stále představují problém. Dalším rizikem jsou softwarové chyby. Technologie není bezchybná a v minulosti došlo k nehodám, které byly způsobeny chybami v programu. Například v roce 2018 smrtelně srazil autonomní vůz Uberu chodkyni, protože software nedokázal správně posoudit její pohyb. Podobné problémy se objevily i u systému Autopilot u Tesly, kdy vozidla nezvládla složité dopravní situace. Další vážnou hrozbou jsou kybernetické útoky. Autonomní vozidla jsou v podstatě sofistikované počítače na kolech, a tak se mohou stát cílem hackerů. Už byly zaznamenány pokusy o nabourání těchto vozidel, což by mohlo mít fatální následky [24]. Představme si situaci, kdy by hacker převzal kontrolu nad autonomním vozem a nasměroval ho do kolize – takové scénáře jsou skutečně vážnou hrozbou.

2 KOMPONENTY AUTONOMNÍHO VOZIDLA

Automatizovaná vozidla mohou vznikat buď rozšířením stávajících vozidel o nové funkce, což je přístup, který volí zejména velké automobilky, nebo jako zcela nové konstrukční návrhy zahrnující i samotnou stavbu vozidla, což je typické zejména pro vozidla bez ovládacích prvků. Druhý přístup umožňuje optimalizovat konstrukci tak, aby vyhovovala specifickým požadavkům automatizace. Příkladem takového řešení je kategorie vozidel označovaná jako *peoplemover*, tedy autonomní vozidla určená k přepravě osob po silnicích, nebo doručovací vozidla sloužící pro rozvoz zásilek. Základní architektura automatizovaných vozidel se skládá z několika bloků, jako je například sledování okolí pomocí senzorů (*sensing*), lokalizace a porozumění okolnímu prostředí (*perception*), rozhodování o pohybu vozidla (*decision*) a blok aktuace (*actuation*), který tato rozhodnutí realizuje prostřednictvím hardwaru vozidla. Interní stavy vozidla jsou monitorovány senzory, které jsou součástí vozidla. Příkladem je stabilizační systém, který sleduje stav vozidla pomocí úhlů natočení volantů, akcelerometru a gyroskopu, přičemž porovnává požadovanou trajektorii se skutečnou dráhou vozidla. Externí stavy, tedy okolí vozidla, sledují senzory umístěné ve vozidle, například radary v oblasti zadního nárazníku, které zajišťují sledování mrtvého úhlu a monitorují situaci za vozidlem [25]. Okolí lze rovněž vnímat prostřednictvím senzorů umístěných mimo vozidlo, jako je tomu u kooperativního vnímání, kdy vozidlo přijímá informace o ostatních účastnících silničního provozu, dopravním značení nebo stavu semaforů.

2.1 Senzory a jejich funkce

Senzor představuje funkční prvek, který přichází do přímého styku s měřeným prostředím. Termín *senzor* lze chápat jako synonymem pro pojmy *snímač*, *detektor* nebo *převodník*. Citlivá část senzoru, nazývaná čidlo, je místem, kde probíhá snímání měřených veličin. Senzor je základní součástka pro získávání informací, která měří fyzikální, chemické nebo biologické veličiny a obvykle je převádí na elektrické signály, jako jsou napětí, proud nebo kapacita. Jeden z mnoha příkladů může být využití senzorů ve svařovně automobilového závodu, kde dopravník posunuje vyráběné předměty od jednoho stanoviště k druhému [26]. Na každém stanovišti je nutné dopravník zastavit pomocí zjištění náběžné hrany nebo jiné značky posunutého předmětu. Jedná se o funkci koncového spínače.

2.2 Kamerové systémy

Kamery hrají klíčovou roli při rozpoznávání objektů. Vozidla díky nim získávají zásadní údaje potřebné k určení objektů, například stojících nebo pohybujících se osob, a také dočasně umístěných pevných překážek, které se mohou nacházet jak přímo na trase, tak v jejím okolí. Mezi hlavní přednosti kamer patří jejich schopnost přesně zaznamenávat úhly, což napomáhá včasné identifikaci situací, jako je zatáčení jiného vozidla [27]. V hustě osídlených oblastech, kde je potřeba široký úhel pohledu pro snímání chodců a okolní dopravy, nebo při rychlém pohybu v definovaných koridorech, případně na dálnicích, kde je nezbytný dohled na vzdálenost až 300 metrů s omezeným zorným úhlem, což vyžaduje zapojení různých kamerových technologií.

2.2.1 Druhy kamer

Webkamera

Webkamera je v podstatě zařízení podobné digitálnímu fotoaparátu, ale s menším rozlišením a silnější kompresí, které slouží k přenosu a zachycení obrazu (Obrázek 5). Světlo prochází čočkou a dopadá na senzorový čip. Webkamery nejčastěji využívají čipy typu CMOS (Complementary Metal Oxide Semiconductor), což je odlišné od běžných digitálních fotoaparátů, které obvykle využívají čip CCD (Charge Coupled Device) [28]. Webkamery prošly od svého vzniku výrazným vývojem. První z nich byla vyvinuta na Univerzitě v Cambridge v roce 1991, kde sloužila pouze k monitorování kávovaru v počítačové laboratoři. Kvalita videa byla v té době velmi nízká, černobílá a snímková frekvence pomalá. Dnes jsou webkamery schopné poskytovat ultra HD rozlišení, vysokou snímkovou frekvenci a širokou škálu funkcí, jako je automatické zaostřování, digitální zoom či automatické rámování [29]. Mezi nejznámější značky patří Logitech, Sony a Microsoft. Kvalita obrazu se výrazně zlepšila, a některé modely dnes dokonce podporují 4K video.



Obrázek 5: Webkamera [30]

IP digitální kamera

IP kamera je typ digitální bezpečnostní kamery, která přenáší videozáznamy přes IP síť (Obrázek 6). To znamená, že kamera může přenášet obraz i bezdrátově a zachytává obraz ve vysokém rozlišení. IP kamery se připojují k síťovému videorekordéru (NVR) prostřednictvím Wi-Fi, ethernetového kabelu nebo USB [31]. Pro svou funkčnost potřebují pouze připojení k síti. Tyto kamery nabízejí rozlišení od dvou do osmi MPx.



Obrázek 6: IP digitální kamera [32]

Analogová kamera

Analogové kamery převádějí snímáný obraz na analogový signál ve formátech PAL nebo NTSC (Obrázek 7) [33]. Signál z těchto kamer je přenášen prostřednictvím koaxiálního kabelu, který umožňuje propojení kamer se záznamovým zařízením nebo monitorem.



Obrázek 7: Analogová kamera [34]

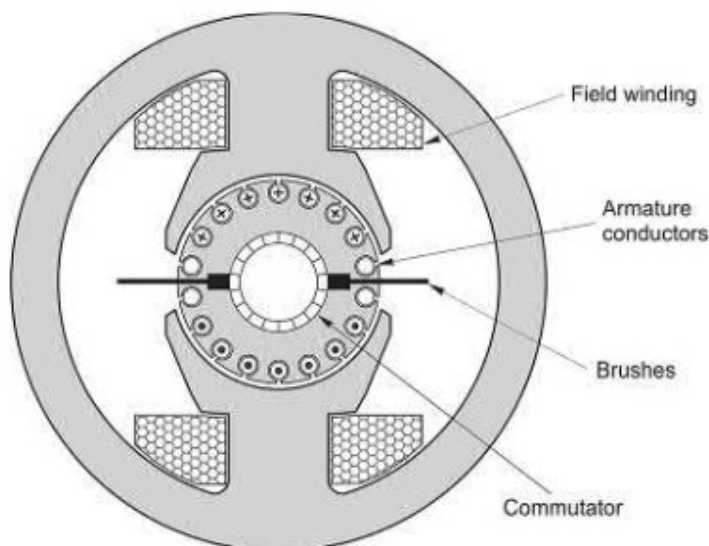
2.3 Elektrické pohony

Elektrické motory jsou natolik běžnou součástí našeho života, že jim často nevěnujeme zvláštní pozornost. Například při zapnutí starší elektrické vrtačky automaticky očekáváme, že se rychle rozběhne na požadovanou rychlost. Nezabýváme se však otázkou, jakým způsobem dosahuje této rychlosti, ani proč po jejím dosažení výrazně klesá spotřeba energie [35]. V okamžiku, kdy začne vykonávat práci, odběr energie se úměrně zvýší, a po jejím dokončení se spotřeba opět sníží, aniž by bylo nutné provádět jakýkoliv zásah.

Stejnoseměrné motory

Konstrukce klasického stejnosměrného motoru zahrnuje několik klíčových prvků, mezi něž patří stator tvořený permanentními magnety a rotor složený z vinutí navinutých na jádru z měděných plechů. Pohyb rotoru je zajištěn uhlíkovými kartáči, které jsou ve styku s komutátorem složeným z jednotlivých lamel (Obrázek 8). Po připojení napětí na rotorové vinutí jím začne procházet elektrický proud, jenž vytváří magnetické pole. Toto pole interaguje s magnetickým polem statoru a způsobuje otáčení rotoru. Jakmile se rotorová

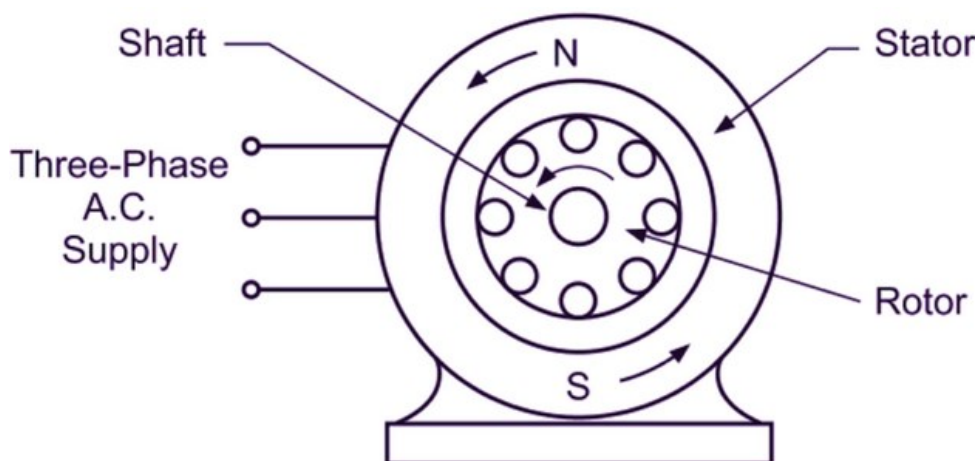
vinutí dostanou do polohy rovnoběžné se siločarami magnetického pole statoru, komutátor změni polaritu napájení, což vede k obrácení směru magnetického pole v rotoru [36]. Rotace však pokračuje stále stejným směrem, čímž je zajištěn nepřetržitý pohyb motoru a pohon připojených mechanických prvků.



Obrázek 8: Komutátorový motor [35]

Indukční motory

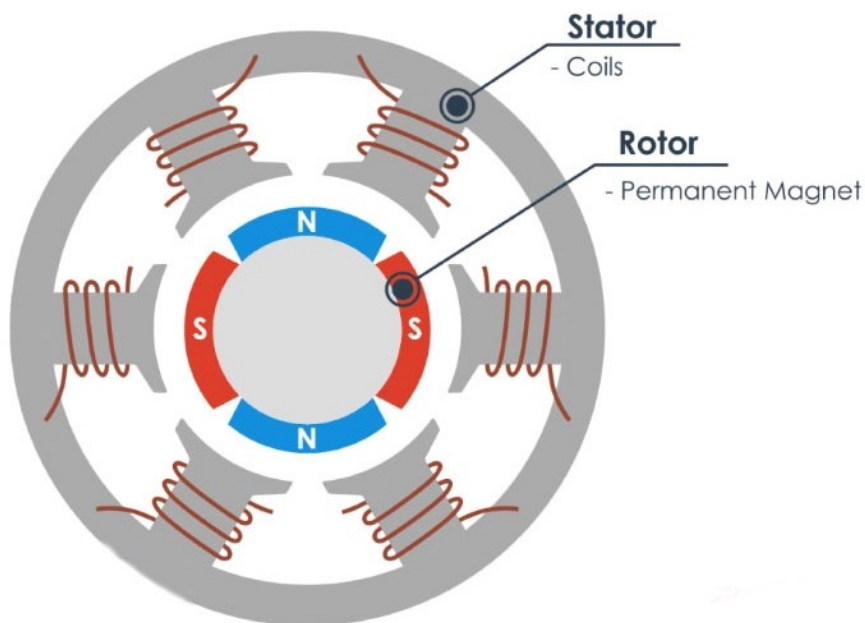
Asynchronní motor, často označovaný také jako indukční motor, patří mezi nejrozšířenější a nejvýznamnější elektrotechnické vynálezy. Jeho obliba je dána zejména spolehlivostí, jednoduchou konstrukcí a vysokou účinností, což z něj činí nepostradatelnou součást mnoha aplikací. Tento typ motoru funguje na principu elektromagnetické indukce a na rozdíl od motorů využívajících kartáče k přenosu elektrické energie pracuje zcela bezkontaktně. Díky tomu je méně náchylný k mechanickému opotřebení a vyznačuje se dlouhou životností. Základními prvky indukčního motoru jsou stator a rotor. Stator, tvořící pevnou vnější část motoru, obsahuje cívky napájené střídavým proudem, které vytvářejí rotující magnetické pole (Obrázek 9) [37]. Rotor umístěný uvnitř statoru se pak otáčí vlivem tohoto pole. Klíčovým faktorem pro správnou činnost motoru je interakce mezi rotujícím magnetickým polem statoru a indukovaným proudem v rotoru, která zajišťuje jeho pohyb a přenos mechanické energie.



Obrázek 9: Třífázový indukční motor [38]

BLDC motor

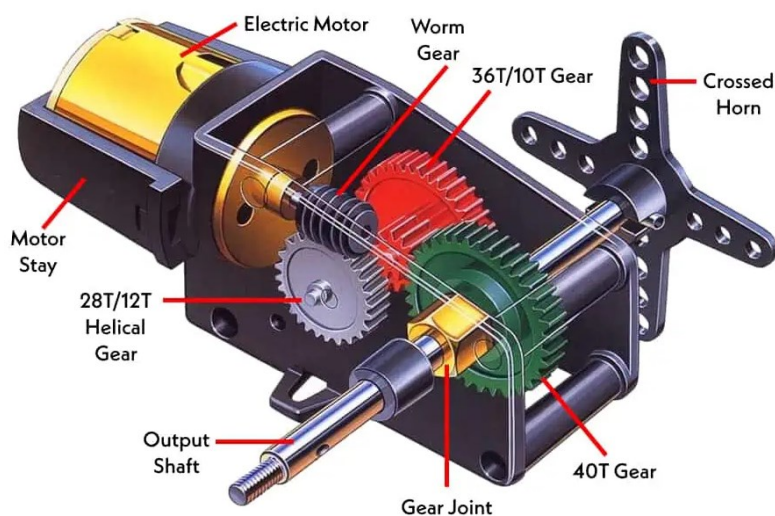
Motory typu BLDC (Brushless DC) – jak už jejich označení napovídá – nevyužívají kartáče. Díky tomu se rotor pohybuje prakticky bez mechanického kontaktu se statorem. Tato bezkartáčová konstrukce přináší řadu výhod, jako je například odstranění problémů spojených s třením a jiskřením, což výrazně zvyšuje celkovou spolehlivost motoru. Zároveň se prodlužuje jeho životnost a snižují se nároky na údržbu. Podobně jako u klasických stejnosměrných motorů vzniká pohyb u BLDC motorů na základě vzájemného působení dvou magnetických polí – jedno je vytvářeno elektromagnetem ve statoru, druhé permanentními magnety v rotoru. Tyto magnetické síly na sebe navzájem působí přitažlivě a odpudivě, čímž dochází k rotaci rotoru. Typická konstrukce BLDC motoru zahrnuje stator s několika vinutími, která generují elektromagnetické pole, a rotor, v němž jsou umístěny permanentní magnety (Obrázek 10). Magnetické pole statoru ovlivňuje rotor na základě přesně definované sekvence elektrických impulzů, které se aplikují na vinutí ve stanoveném pořadí. Díky těmto impulzům a interakci polí dochází k plynulému pohybu. Počet vinutí ve statoru i množství pólů (tj. permanentních magnetů) v rotoru se liší podle konkrétního typu motoru [39]. Obecně platí, že čím více vinutí a pólů motor obsahuje, tím větší točivý moment je schopen generovat.



Obrázek 10: BLDC motor [40]

Servomotor

Servomotor je elektromechanické zařízení, které generuje točivý moment a rychlost v závislosti na přivedeném proudu a napětí (Obrázek 11). Funguje v rámci systémů s uzavřenou smyčkou, kde zpětnovazební řízení a povely z nadřazené řídicí jednotky (servozesilovače) umožňují motoru přizpůsobit svůj točivý moment a rychlost konkrétním požadavkům. Zpětnovazební zařízení umístěné na servomotoru sleduje parametry jako proud, rychlost nebo poloha, které následně předává servozesilovači [41]. Ten na základě těchto informací upravuje akční veličinu motoru tak, aby odpovídala požadavkům zadaným v příkazech.



Obrázek 11: Servomotor [42]

3 ARDUINO A JEHO VYUŽITÍ V AUTOMATIZACI

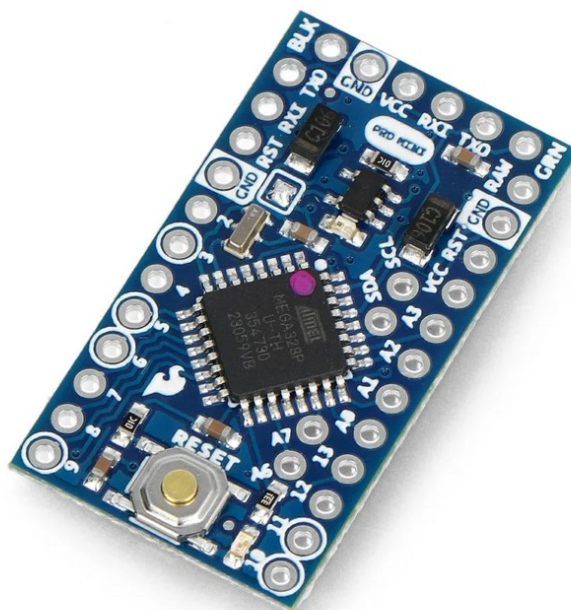
Arduino je univerzální prototypovací platforma, která může plnit různé úlohy podle potřeby. Lze jej využít například jako automatický systém zavlažování rostlin, webový server, nebo autopilot kvadrokoptéry. Jedná se o vývojovou platformu s mikrořadičem, která využívá intuitivní programovací jazyk a integrované vývojové prostředí Arduino (IDE). Připojením senzorů, akčních členů, světel, reproduktorů, rozšiřujících modulů (tzv. shieldů) a dalších integrovaných obvodů lze Arduino použít jako řídicí jednotku pro širokou škálu aplikací [43]. Možnosti využití Arduina jsou téměř neomezené a závisí především na kreativité uživatele.

3.1 Druhy Arduino desek

Vývoj první verze Arduina začal v roce 2005, kdy se tým z italského Interaction Design Institute ve městě Ivrea rozhodl vytvořit cenově dostupnou a jednoduchou vývojovou platformu pro studenty, kteří si nemohli dovolit tehdy rozšířené, ale drahé desky BASIC Stamp. Arduino se mezi studenty rychle uchýlilo, což vedlo tvůrce k rozhodnutí zpřístupnit ho celosvětově nejen prodejem vlastních desek, ale i sdílením veškeré dokumentace a schémat, jelikož se jedná o open-source projekt. Během osmi let vývoje vzniklo mnoho různých verzí Arduina. Díky tomu, že jde o open-source projekt, se vedle oficiálních verzí objevilo i množství neoficiálních variant, označovaných jako klony. Základním prvkem každé desky Arduino je procesor od společnosti Atmel, který je doplněn dalšími elektronickými součástkami. Většina desek se vyznačuje jednotným grafickým zpracováním, kde převažuje modrá barva. V internetových obchodech i na oficiálních stránkách Arduina lze narazit na desky, u jejichž názvu se nachází označení jako Rev3 nebo R3. Toto označuje konkrétní verzi dané desky. Mezi jednotlivými verzemi se mohou vyskytnout drobné rozdíly, například v rozložení součástek nebo designu. Nejde však o zásadní změny, které by vedly k vytvoření zcela nové desky. Na většině modelů se kromě hlavního čipu nachází také převodník umožňující komunikaci mezi počítačem a čipem. Existují však i desky, které převodník neobsahují. Důvodem může být buď úspora místa, což vyžaduje použití externího převodníku, nebo skutečnost, že je převodník již integrován přímo v čipu. Nyní budou jednotlivé desky postupně představeny, a to v pořadí od nejmenších po největší.

Arduino Mini

Arduino Mini je jednou z nejmenších oficiálních verzí Arduina, navržená s důrazem na úsporu místa (Obrázek 12). Malé rozměry však znamenají absenci USB portu, a proto je k programování nutné použít externí USB 2 sériový převodník. I přesto výkon této desky nijak nezaostává za většími verzemi [44]. Používá procesor ATmega328 s taktem 16 MHz. Díky svým kompaktním rozměrům je ideální pro použití v aplikacích, jako jsou chytré vypínače nebo dálkové ovladače.



Obrázek 12: Arduino Mini [45]

Arduino Wemos D1 Mini

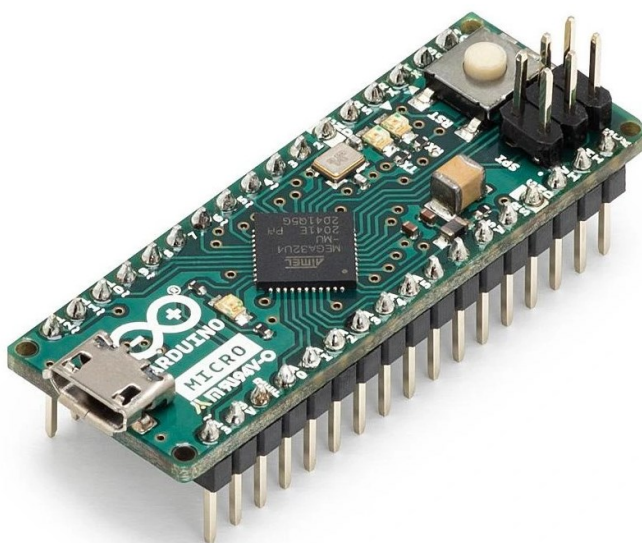
Modul Wemos D1 Mini je kompaktní vývojová deska postavená na 32bitovém mikrokontroleru ESP8266, který má integrovaný WiFi modul (Obrázek 13). Díky kompatibilitě s platformou Arduino je vývoj a implementace projektů výrazně jednodušší. Pro připojení k WiFi síti je nutné napájet desku 5 V přes micro USB port. Deska disponuje 11 digitálními vstupně/výstupními piny, přičemž všechny kromě D0 podporují funkce jako PWM (Pulzně šířková modulace) a přerušení [46]. Kromě toho je k dispozici jeden analogový vstup s rozsahem napětí 0-3,3 V. Frekvenci procesoru lze nastavit na 80 MHz nebo 160 MHz podle požadavku konkrétní aplikace, což poskytuje flexibilitu při návrhu projektu.



Obrázek 13: Arduino Wemos D1 Mini [47]

Arduino Micro

Mikrokontrolerová deska Micro je postavena na čipu ATmega32U4 a vznikla ve spolupráci s firmou Adafruit (Obrázek 14). Nabízí 20 digitálních vstupně – výstupních pinů, přičemž 7 z nich lze využít jako PWM výstupy a 12 jako analogové vstupy. Dále je vybavena 16Mhz krystalovým oscilátorem, micro USB konektorem, ICSP headerem a resetovacím tlačítkem. Obsahuje veškeré komponenty potřebné pro běh mikrokontroléru. Díky svým rozměrům je kompatibilní s nepájivým polem, což usnadňuje její zapojení [48]. Funkčně se Micro podobá Arduino Leonardo, protože čip ATmega32U4 disponuje integrovanou USB komunikací, čímž odpadá nutnost využívání sekundárního procesoru.



Obrázek 14: Arduino Micro [49]

Arduino Uno

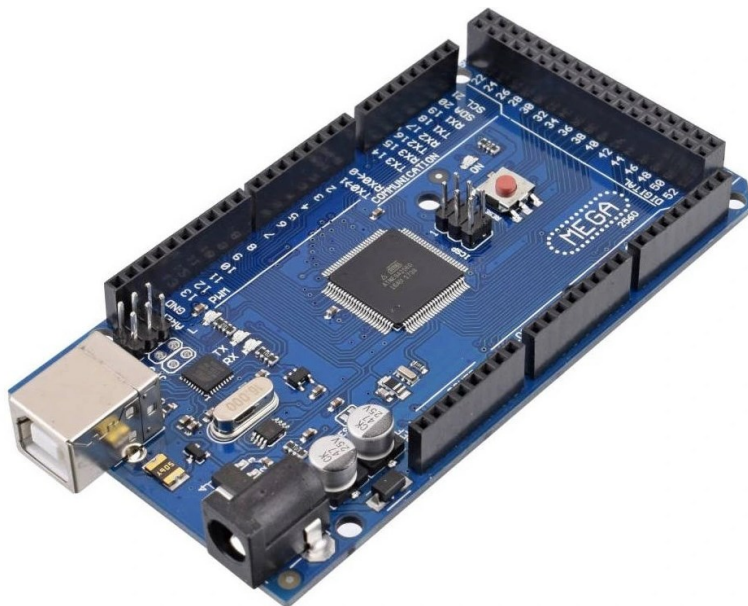
Arduino Uno představuje vývojovou platformu s mikrokontrolérem, která využívá čip ATmega328P (Obrázek 15). Má 14 digitálních pinů, které slouží pro vstup i výstup (z toho 6 lze použít jako PWM výstupy), 6 analogových vstupů, keramický rezonátor s frekvencí 16 MHz. Využívá port USB pro připojení, napájecí konektor, ICSP rozhraní a resetovací tlačítko. Název *Uno*, což v italštině znamená *jeden*, byl zvolen k označení vydání verze 1.0 Arduino Software (IDE) [50]. Deska Uno a verze 1.0 Arduino Software (IDE) byly původními referenčními verzemi platformy Arduino, která se od té doby vyvinula do novějších verzí. Arduino Uno je první v sérii USB desek a představuje referenční model pro celou platformu Arduino.



Obrázek 15: Arduino Uno [51]

Arduino Mega2560

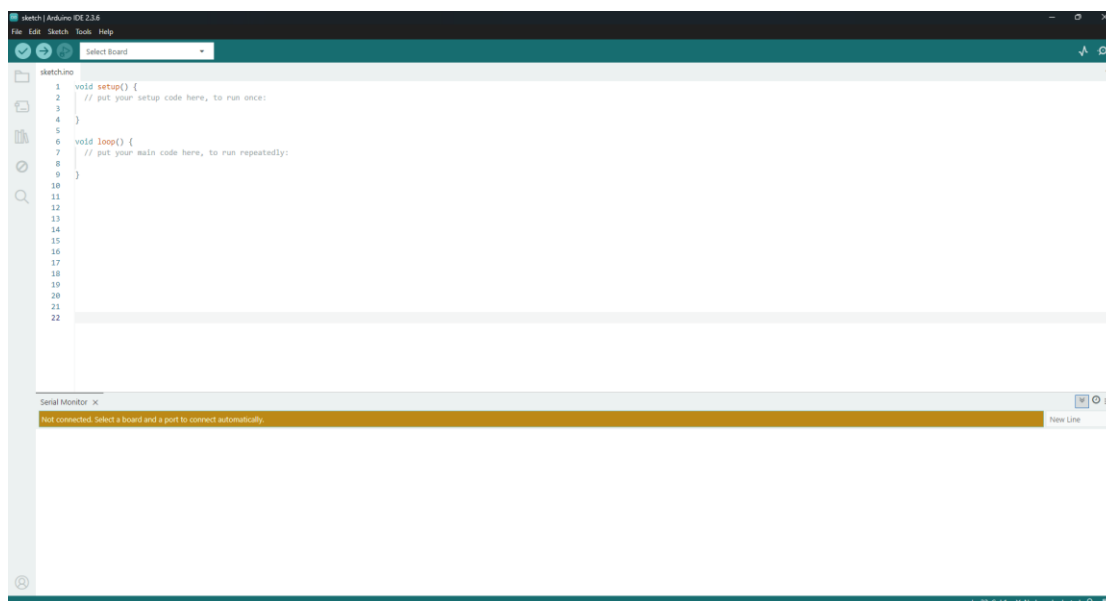
Vývojová deska MEGA 2560 je plně kompatibilní s platformou Arduino a je vybavena mikrokontrolérem ATmega2560 (Obrázek 16). Všechny vstupy a výstupy jsou rozmístěny kolem okraje desky. Obsahuje USB čip CH340, což umožňuje programování desky pomocí USB kabelu. Deska disponuje 54 digitálními vstupy/výstupy (z toho 15 lze použít jako PWM výstupy), 16 analogovými vstupy, 4 UART porty (hardwarové sériové porty), 16 MHz krystalovým oscilátorem, USB připojením, napájecím konektorem, ICSP záhlavím a tlačítkem pro resetování [52]. V porovnání s Arduino Uno má tato deska výhodu v podobě vyššího počtu vstupů/výstupů a 256 kB flash paměti (ve srovnání s 32 kB u Uno), což umožňuje ukládat i složitější programy.



Obrázek 16: Arduino Mega2560 [53]

3.2 Vývojové prostředí Arduino

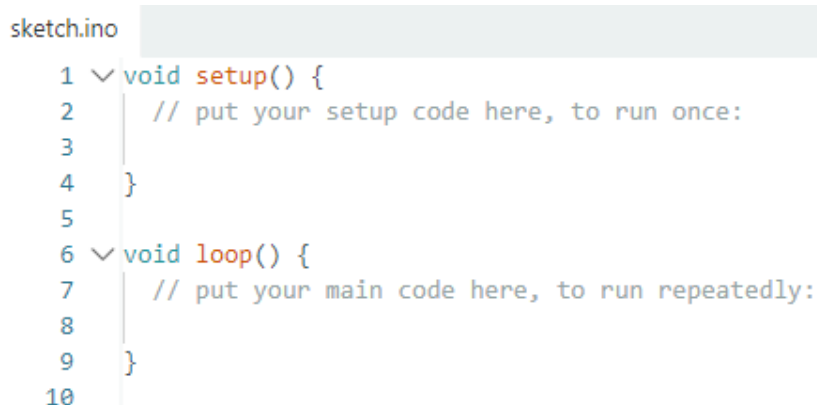
Integrované vývojové prostředí IDE (Integrated Development Environment) obsahuje řadu nástrojů, které usnadňují programování Arduina (Obrázek 17). Nejvýraznější součástí je textový editor umístěný v bílém okně, kde lze psát kód. K dispozici jsou také různé nabídky, například pro volbu typu Arduina, sériového portu, ke kterému je zařízení připojeno, nebo pro přístup k ukázkovým příkladům. Mezi často využívané nástroje patří také okno zobrazující zprávy ze sériové linky.



Obrázek 17: Arduino IDE

Psaní sketches

Program, psaný pro Arduino, se nazývá sketch (v překladu návrh nebo náčrt). Tento kód bývá vytvářen v textovém editoru integrovaného vývojového prostředí. Soubory obsahující Arduino sketch mají příponu .ino (Obrázek 18). Textový editor zabírá hlavní část obrazovky, přičemž se zobrazuje jako velká bílá plocha. Pod ním se nachází konzole (černá oblast), která poskytuje informace o průběhu nahrávání programu do Arduino a případné chybové hlášky. V pravém dolním rohu se zobrazují údaje o aktuálně vybrané desce a sériovém portu, ke kterému je připojena.



```
sketch.ino
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
10
```

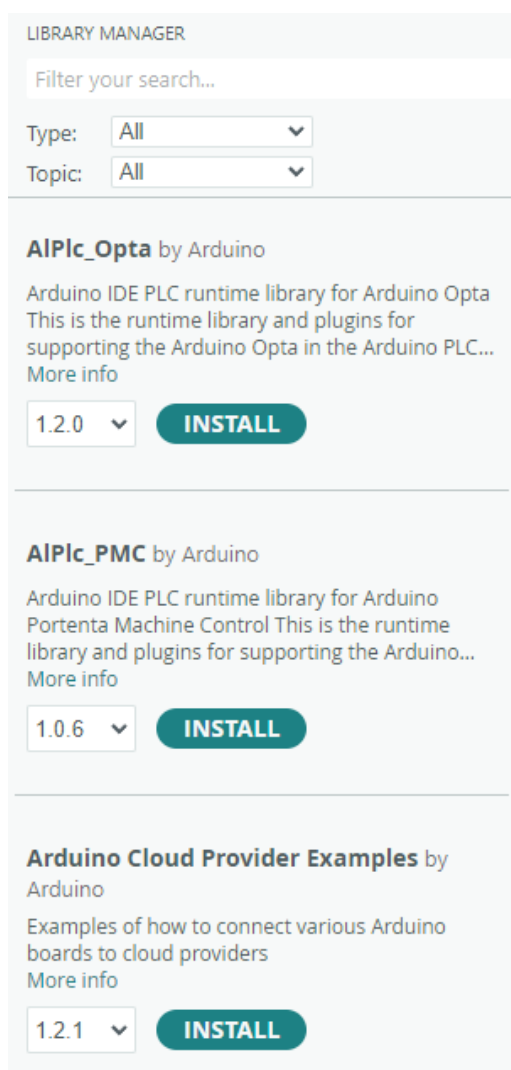
Obrázek 18: Skecth v Arduino IDE

Nahrávání kódu

Před nahráním sketche do Arduino je nutné zvolit správný sériový port v nabídce Tools > Serial Port a odpovídající desku v Tools > Board. Po správném nastavení sériového portu stačí kliknout na tlačítko Upload v nástrojové liště nebo vybrat možnost Upload v nabídce File. Novější verze desek Arduino se automaticky resetují a zahájí proces nahrávání. U starších modelů, například před verzí Decimila, které nemají funkci automatického restartu, je nutné před nahráváním manuálně stisknout tlačítko Reset. Během nahrávání na většině desek blikají LED diody RX a TX. Po dokončení procesu vývojové prostředí zobrazí zprávu o úspěšném nahrání nebo upozorní na případnou chybu. Při nahrávání programu se využívá Arduino bootloader, což je malý software, který je do mikrokontroléru nahrán již při výrobě desky [54]. Tento bootloader umožňuje nahrávání kódu bez potřeby dodatečného hardwaru. Aktivuje se krátce po restartu desky, kdy několikrát blikne LED dioda připojená k pinu 13, a poté začne vykonávat nahraný program.

Knihovny

V řadě návodů pro práci s Arduinem se často objevuje doporučení importovat příslušnou knihovnu, pokud chceme rozšířit funkcionalitu našeho projektu. Knihovna představuje soubor již hotových funkcí, které výrazně usnadňují samotné programování. Využíváme tak řešení, která někdo jiný předem připravil, čímž šetříme nejen čas, ale i rozsah potřebného kódu. Jelikož Arduino funguje jako open-source platforma, jsou i jeho knihovny volně dostupné online (Obrázek 19) [55]. Mnohé jsou k nalezení například na platformě GitHub.



Obrázek 19: Výběr knihoven

Programovací jazyk Arduino

Programovací jazyk používaný na platformě Arduino je založený na jazyce Wiring. Přesněji řečeno, Arduino z Wiringu vychází, i když rozdíly mezi nimi jsou spíše drobné. Autoři Wiringu jej popisují jako jazyk podobný C++, nicméně ve skutečnosti jde o rozšíření C++ v podobě sady maker a funkcí, které usnadňují práci s mikrokontroléry [56]. Jazyk Wiring

byl původně navržen pro vývojovou desku připomínající Arduino a jeho základy vycházejí z open source projektu Processing.

3.3 Ovládání a komunikace pomocí WeMos D1 Mini (ESP8266)

ESP8266 poskytuje vysoce integrované Wi-Fi SoC (System on a Chip) řešení, které vychází vstříc neustálým požadavkům uživatelů na efektivní spotřebu energie, kompaktní design a spolehlivý výkon v oblasti internetu věcí [57]. ESP8266 podporuje režim klienta a režim stanice. Kromě těchto funkcí WeMos D1 také podporuje funkci OTA (On The Air), která zcela eliminuje potřebu nahrávání kódu přes kabel. Připojení k WeMos D1 je možné prostřednictvím standardního USB kabelu s konektorem Micro USB [58]. Deska Arduino navíc obsahuje konektor pro napájení z externího adaptéru.

ESP WiFi webserver

ESP WiFi webserver je užitečná knihovna pro desky ESP32 a ESP8266, která umožňuje vytvářet nejen měřicí, ale i řídicí bezdrátová zařízení. Může být tedy využita například pro ovládání osvětlení, spínání zátěže pomocí relé a další podobné úkoly [59]. Webový server lze jednoduše popsat jako místo, kde jsou uloženy a zpracovávány webové stránky. K serveru se nejčastěji přistupuje prostřednictvím webového prohlížeče, který je také označován jako webový klient. V praxi to funguje tak, že prohlížeč odesílá požadavek na server, který požadavek nějakým způsobem zpracuje a vrátí výsledek zpět do prohlížeče [60]. Komunikaci mezi klientem a serverem zajišťuje protokol Hypertext Transfer Protocol, zkráceně HTTP nebo HTTPS.

WebSocket Server

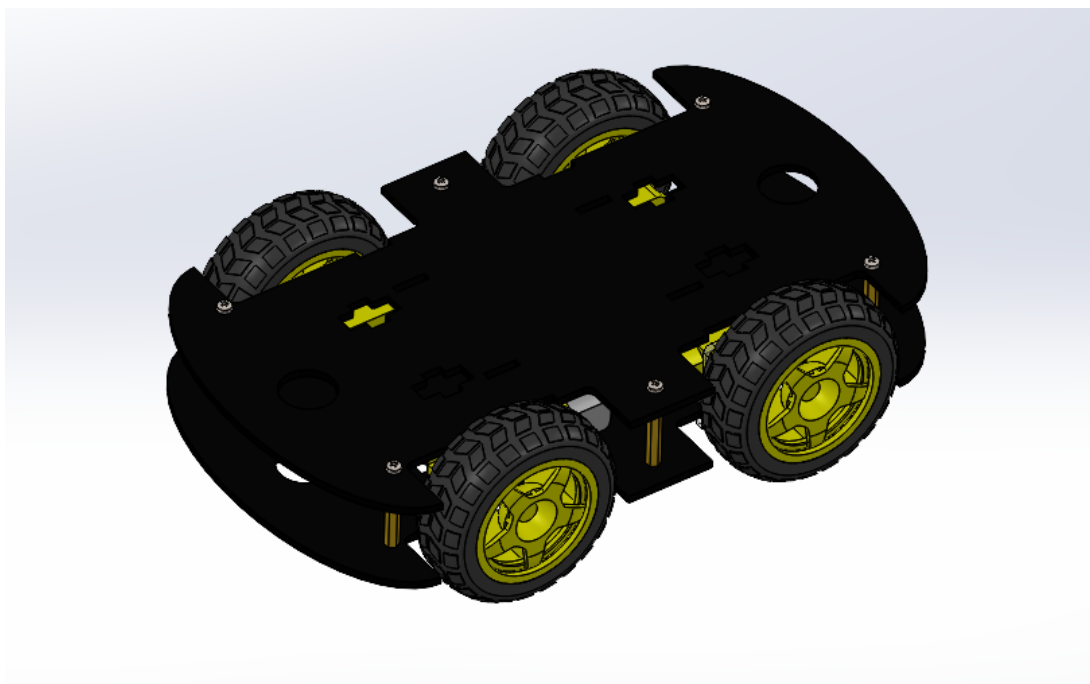
Knihovna Websockets slouží k vytváření WebSocket serverů a klientů v Pythonu a zaměřuje se na správnost, jednoduchost, robustnost a výkon. Podporuje různé způsoby síťového I/O a řízení toku. Výchozí implementace je založena na knihovně asyncio, která je součástí Pythonu a poskytuje asynchronní I/O. Elegantní řešení nabízí API postavené na korutinách, což je ideální pro servery obsluhující velké množství klientů. Alternativní implementace využívající vlákna je vhodná pro klientské aplikace, zejména pokud uživatel není obeznámen s asyncio, a může se také využít na serverech s malým počtem klientských připojení [61]. Implementace bez I/O je určena pro integraci do knihoven třetích stran, přičemž ji používá i samotná knihovna Websockets pro interní potřeby.

II PRAKTICKÁ ČÁST

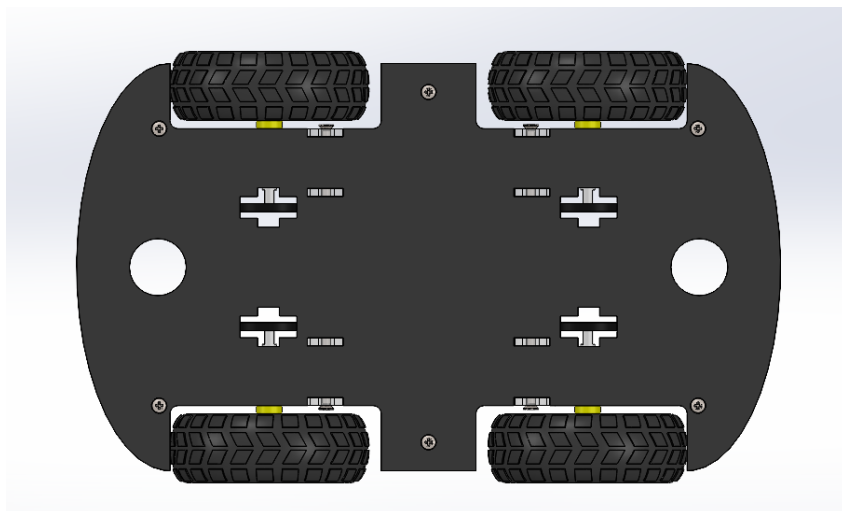
4 NÁVRH AUTONOMNÍHO VOZIDLA

V tomto projektu bylo navrženo autonomní vozidlo, které je schopné samostatně přejíždět z bodu A do bodu B. Vozidlo využívá webkameru umístěnou vysoko nad sebou k monitorování okolí a detekci překážek na cestě. Na základě těchto informací se vozidlo dynamicky rozhoduje o směru pohybu, aby se vyhnulo jakýmkoliv objektům a bezpečně dorazilo do cíle. Tato autonomní navigace využívá algoritmy pro zpracování obrazu a řízení pohybu, které umožňují efektivní a bezpečný pohyb bez zásahu uživatele.

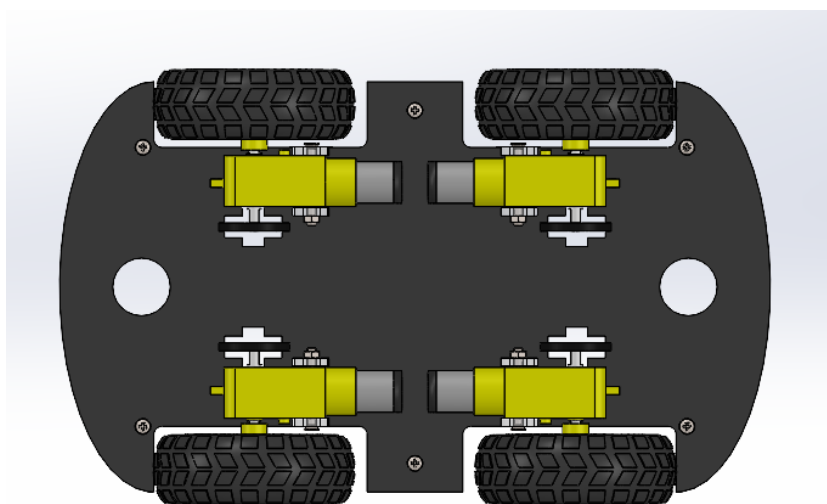
Pro realizaci svého autonomního vozidla byl využit kit „UNO Smart Robot Car Kit“ pro Arduino, který je kompatibilní s Arduino IDE a nabízí jednoduchý základ pro stavbu robotických vozidel. Sada obsahuje všechny nezbytné komponenty, jako jsou motory, kola a desky pro upevnění, které jsou klíčové pro pohyb vozidla. Pro řízení vozidla byl zvolen modul WeMos D1 mini, který je ideální pro bezdrátovou komunikaci díky své schopnosti připojení k WiFi. Tento modul umožňuje ovládání vozidla pomocí Pythonu přes WebSocket komunikaci. Vytvoření modelů pro projekt bylo zvoleno prostředí SOLIDWORKS. Vizualizace projektu je znázorněna na obrázcích níže (Obrázek 20, 21, 22, 23):



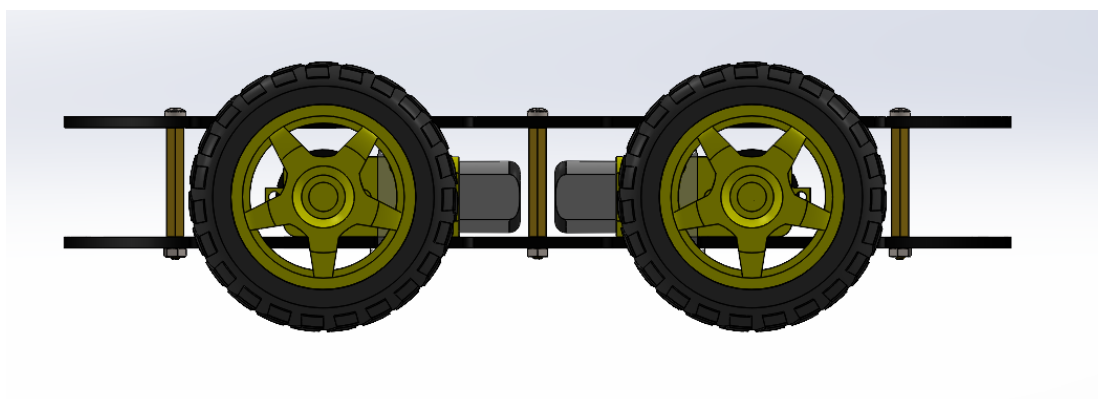
Obrázek 20: Izometrický pohled 3D modelu vozidla



Obrázek 21: Spodní pohled 3D modelu vozidla



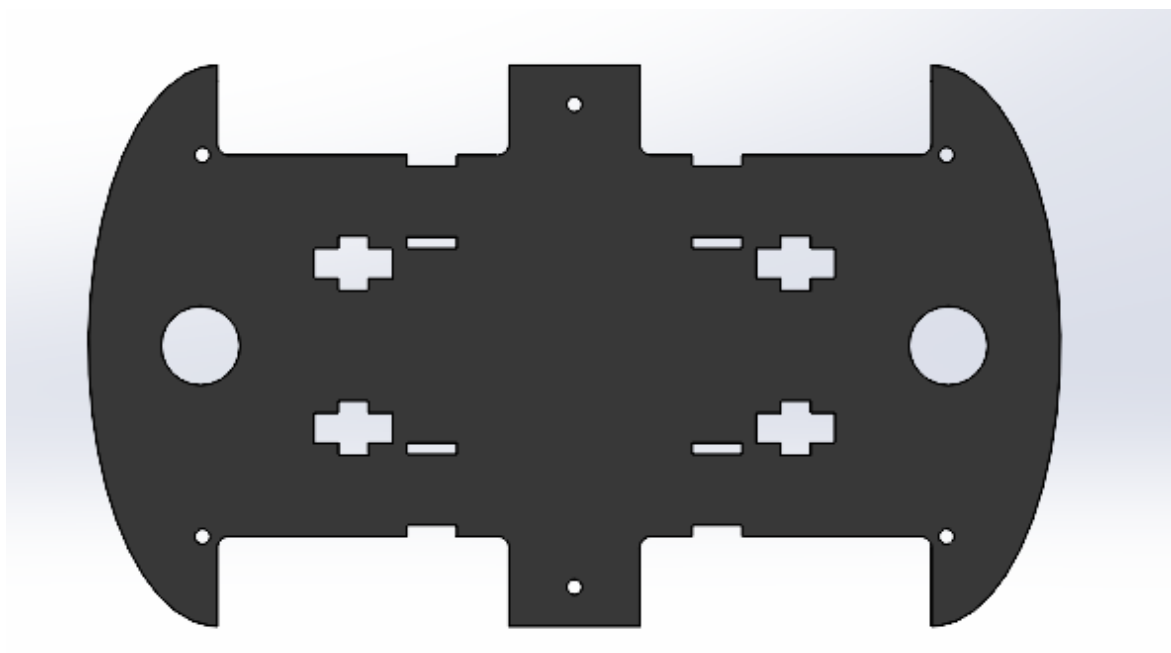
Obrázek 22: Horní pohled 3D modelu vozidla bez vrchní konstrukce



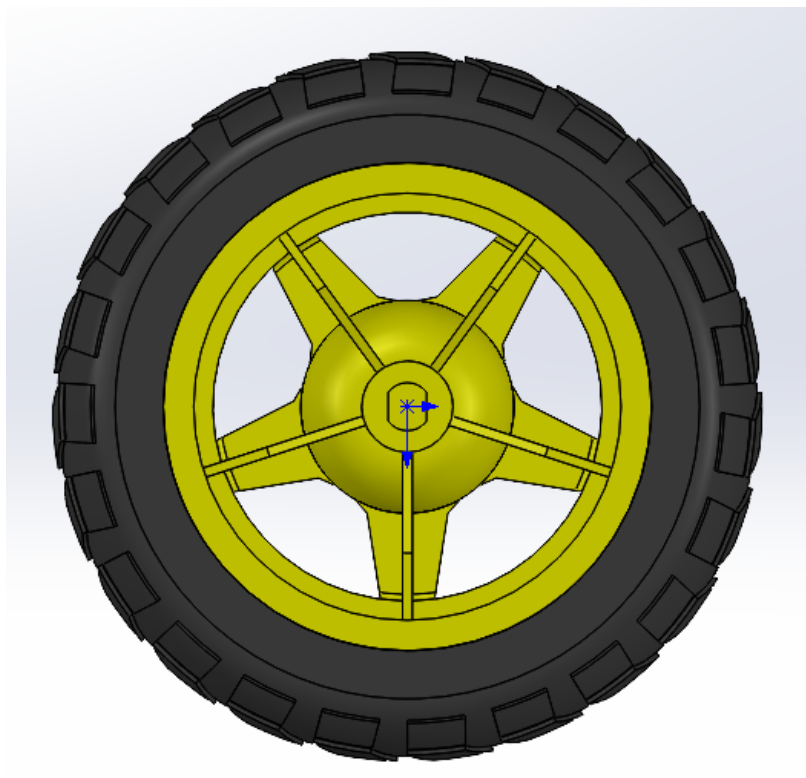
Obrázek 23: Boční pohled 3D modelu vozidla

4.1 Vybrané komponenty

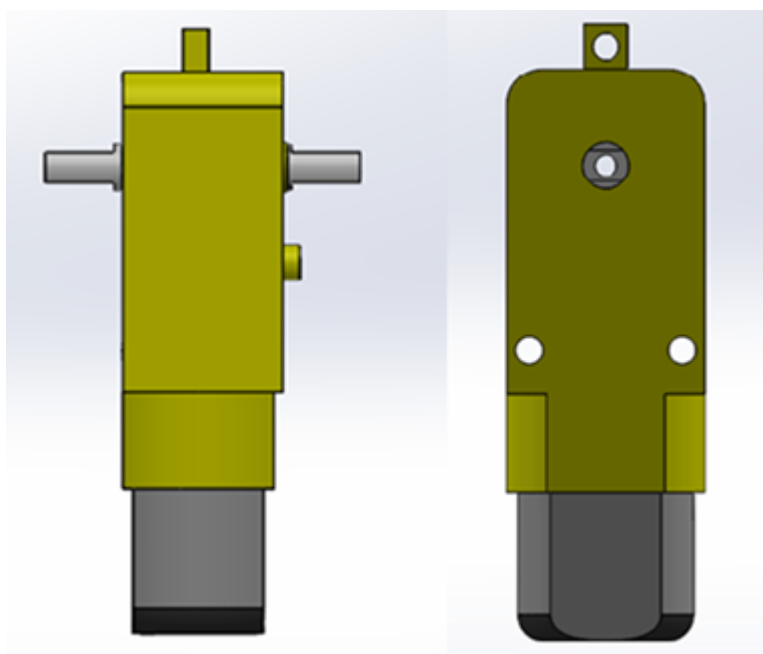
Pro konstrukci autonomního vozidla byly použity klíčové komponenty, kterými je zajištěna správná funkce a pohyb vozidla (Obrázek 24, 25). Pohon je zajišťován čtyřmi stejnosměrnými (DC) motory, pomocí nichž je umožněn pohyb a otáčení vozidla (Obrázek 26). Tyto motory jsou řízeny H-můstkem, což poskytuje flexibilitu v řízení směru otáčení a regulaci rychlosti (Obrázek 27). Dále je do systému integrován vypínač, který slouží k zapnutí a vypnutí vozidla, čímž je zajištěna bezpečnost a efektivní využívání energie (Obrázek 28). Napájení je zajištěno držákem na baterie, který poskytuje potřebnou energii pro chod celého vozidla (Obrázek 29). Jako centrální řídicí jednotka slouží WiFi modul WeMos D1 Mini, který komunikuje s počítačem prostřednictvím WebSocket připojení (Obrázek 30). Toto připojení je napojeno na Python kód, prostřednictvím kterého je realizováno řízení vozidla a interakce s dalšími komponentami systému. Modul je navržen tak, aby umožňoval snadné rozšíření o další senzory a zařízení, čímž zvyšuje flexibilitu celého systému.



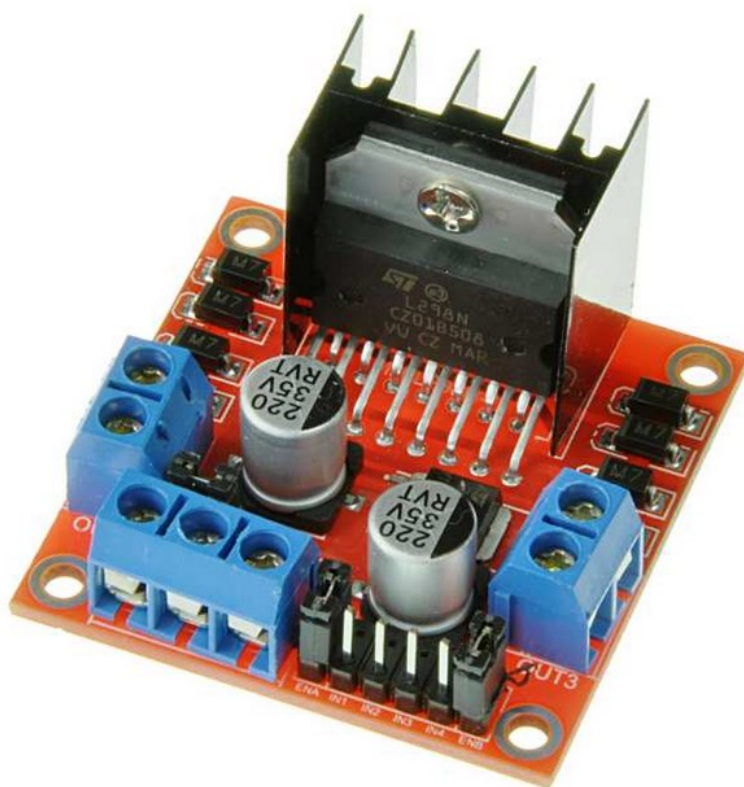
Obrázek 24: 3D model základní konstrukční desky



Obrázek 25: 3D model kola



Obrázek 26: Boční a čelní pohled 3D modelu DC motoru



Obrázek 27: H-můstek [62]



Obrázek 28: Vypínač [63]



Obrázek 29: Držák na baterie [64]

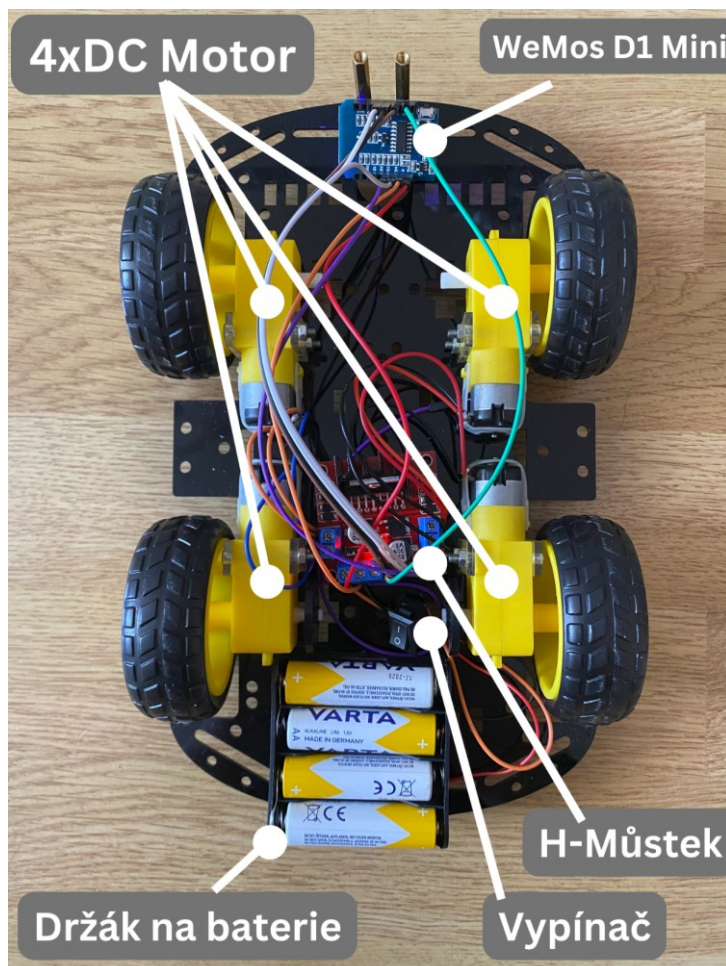


Obrázek 30: Arduino Wemos D1 Mini [47]

4.2 Montáž a konstrukce vozidla

Po výběru a přípravě všech potřebných komponent bylo přistoupeno k montáži samotného vozidla. Základ tvoří dvoupatrová konstrukční platforma z robotického kitu, na kterou byly postupně připevněny všechny klíčové části. Ve spodní části podvozku jsou umístěny čtyři stejnosměrné (DC) motory, zajišťující pohon všech kol. Mezi nimi je uchycen držák na baterie, který slouží jako zdroj napájení pro celý systém. Na horní část konstrukce byla instalována řídicí elektronika, konkrétně modul WeMos D1 mini, H-můstek pro řízení motorů a vypínač pro spouštění a vypínání celého zařízení. Jednotlivé komponenty byly vzájemně propojeny dle zapojovacího schématu s důrazem na přehlednost a funkčnost vedení kabeláže (Obrázek 31). Na vrchní straně vozidla je umístěn ArUco marker, který

slouží k určování polohy a orientace vozidla prostřednictvím kamery snímající prostor shora. Marker je umístěn přibližně ve středu vozidla, nikoliv v jeho přední části, což zajišťuje přesnější určení jeho pozice při detekci v obrazu.



Obrázek 31: Horní pohled – rozložení komponent

4.3 Schéma zapojení

Na níže vloženém obrázku je znázorněno kompletní zapojení elektronických součástek autonomního vozidla s mikrokontrolerem WeMos D1 mini, motorovým driverem L298N, čtyřmi stejnosměrnými motory (DC motory) a napájením z držáku se čtyřmi AA bateriemi (1,5 V) (Obrázek 32). Návrh byl vytvořen v online nástroji Cirkuit Designer, který umožňuje přehlednou vizualizaci a návrh obvodových schémat.

Napájecí část tvoří čtyři sériově zapojené baterie, které poskytují napětí přibližně 6 V. Kladný pól bateriového výstupu je připojen na mechanický spínač, který slouží zapnutí a vypnutí celého zapojení. Výstup ze spínače pokračuje do vstupu 12V motorového driveru L298N. Záporný pól baterií je připojen přímo na svorku GND motorového modulu. Modul

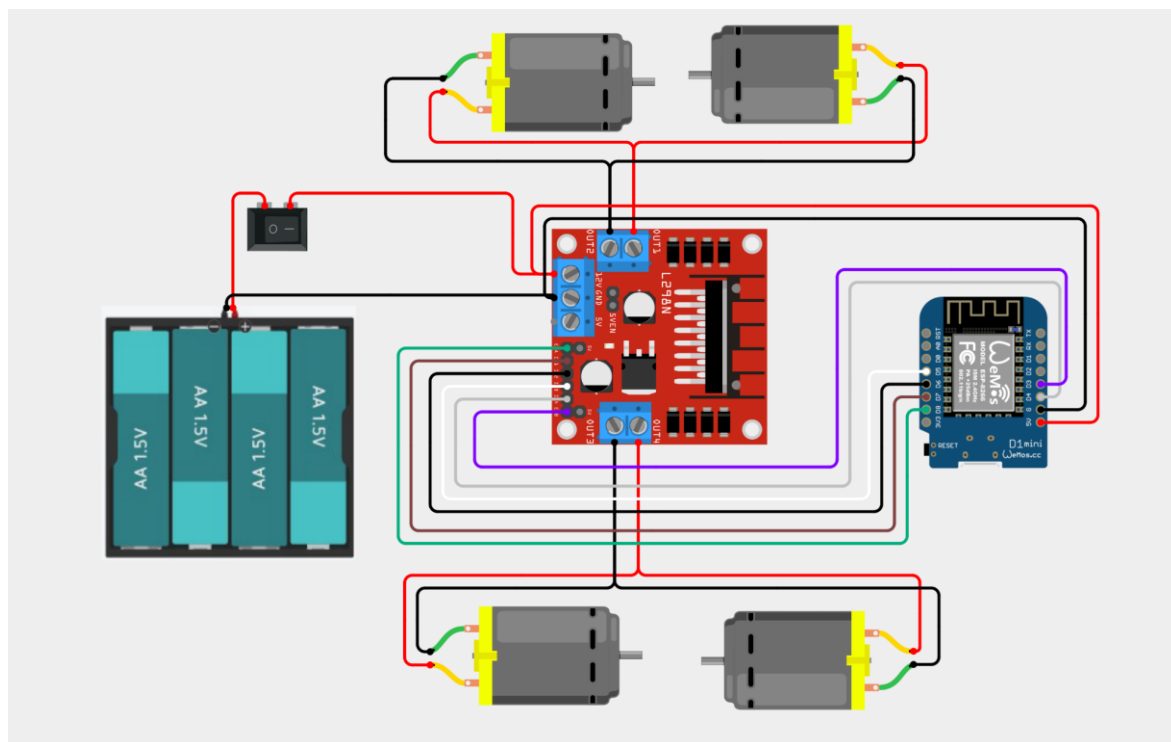
L298N slouží jako můstek (tzv. H-můstek), kterým je řízen směr otáčení motorů. Obsahuje čtyři výstupní kanály (OUT1 až OUT4), přičemž každý z těchto kanálů je připojen k jednomu z motorů. V tomto zapojení jsou vždy dva motory připojeny paralelně k jednomu výstupu, čímž je zajištěn pohyb obou kol na dané straně (např. levé přední a levé zadní kolo k OUT1 a OUT2, pravé přední a zadní kolo k OUT3 a OUT4).

Řídící část tvoří mikrokontroler WeMos D1 mini, založený na čipu ESP8266, který je zároveň vybaven WiFi modulem pro bezdrátovou komunikaci. Z mikrokontroleru jsou odesílány řídicí signály na vstupy IN1 až IN4, a dále také na piny ENA a ENB motorového driveru. Zapojení konkrétních pinů je následující:

- pin D3 je připojen k ENB (povolení motoru B),
- pin D4 k IN4,
- pin D5 k IN3,
- pin D6 k IN2,
- pin D7 k IN1,
- pin D8 k ENA (povolení motoru A).

Napájení WeMos modulu je zajištěno vedením z hlavního napětí (ze společného vedení s driverem), kde je připojen na piny 5 V a GND. Díky tomu je mikrokontroler napájen ze stejného zdroje jako zbytek systému, čímž je celková konstrukce zjednodušená.

Takto navržené zapojení umožňuje řízení směru a rychlosti otáčení jednotlivých motorů na základě signálů posílaných přes WebSocket z webového rozhraní, které běží na PC nebo mobilním zařízení připojeném ke stejné síti.



Obrázek 32: Schéma zapojení komponent

5 PROGRAMOVÉ VYBAVENÍ VOZIDLA

Při programování vozidla byl celý proces rozdělen do dvou samostatných částí. První část se týká samotného modulu WeMos D1 mini, který byl naprogramován v prostředí Arduino IDE. Pomocí tohoto kódu je realizováno přímé ovládání hardwarových prvků vozidla, jako jsou motory nebo řízení směru jízdy. Druhá část je tvořena aplikací psanou v programovacím jazyce Python, která běží na počítači a komunikuje s modulem WeMos přes WebSocket. Tímto způsobem jsou vozidlu předávány pokyny potřebné pro autonomní jízdu.

5.1 Arduino kód

V této podkapitole je uveden konkrétní kód první části. Pro větší přehlednost jsou rozděleny do sekcí s popisem i obrázkem kódu. Celý kód je uveden v přílohách této práce (Příloha P I: Arduino kód).

Na snímku (Obrázek 33) je znázorněn začátek kódu, kde jsou načteny knihovny potřebné pro připojení k WiFi síti, vytvoření webového serveru a WebSocket komunikaci. Dále jsou zde definovány proměnné pro SSID, heslo WiFi sítě a nastavení pinů, které slouží k řízení motorů vozidla. V závěru je vytvořen WebSocket server na portu 81 a webový server na portu 80.

```
1  #include <ESP8266WiFi.h>
2  #include <ESP8266WebServer.h>
3  #include <WebSocketsServer.h>
4
5
6  const char* ssid = "*****";
7  const char* password = "*****";
8
9  #define ENB D3
10 #define MOTORB_1 D4
11 #define MOTORB_2 D5
12 #define MOTORA_1 D7
13 #define MOTORA_2 D6
14 #define ENA D8
15
16 WebSocketsServer websocket = WebSocketsServer(81); // WebSocket server na portu 81
17 ESP8266WebServer server(80); // WebServer na portu 80
```

Obrázek 33: Inicializace knihoven a nastavení pinů

Po inicializaci knihoven následuje spuštění funkce setup, v této části je nejprve aktivována sériová komunikace a zahájeno připojování zařízení k bezdrátové síti (Obrázek 34). Proces připojení je monitorován pomocí výpisů do sériového monitoru. Po navázání spojení je spuštěn HTTP server, který slouží k obsluze webového rozhraní,

a WebSocket server, prostřednictvím něhož je realizován přenos ovládacích příkazů v reálném čase.

```
19 void setup() {
20     pinMode(ENA, OUTPUT);
21     pinMode(ENB, OUTPUT);
22     pinMode(MOTORA_1, OUTPUT);
23     pinMode(MOTORA_2, OUTPUT);
24     pinMode(MOTORB_1, OUTPUT);
25     pinMode(MOTORB_2, OUTPUT);
26
27     digitalWrite(ENA, LOW);
28     digitalWrite(ENB, LOW);
29
30     Serial.begin(115200);
31     WiFi.begin(ssid, password);
32
33     Serial.print("Connecting to WiFi...");
34     while (WiFi.status() != WL_CONNECTED) {
35         delay(500);
36         Serial.print(".");
37     }
38     Serial.println("Connected!");
39     Serial.print("IP Address: ");
40     Serial.println(WiFi.localIP());
41
42     server.on("/", HTTP_GET, [](){
43         server.send(200, "text/html", getHTML());
44     });
45
46     websocket.begin(); // Start WebSocket server
47     websocket.onEvent(webSocketEvent); // Při událostech WebSocketu
48     server.begin();
49 }
```

Obrázek 34: Inicializace pinů, připojení k WiFi síti a spuštění serverů

Na snímku (Obrázek 35) je znázorněna funkce `webSocketEvent`, která slouží k obsluze událostí v rámci WebSocket komunikace. Program detekuje připojení a odpojení klienta, případně přijímá textové zprávy. Na základě přijatého příkazu jsou následně vyvolány funkce pro pohyb vozidla dopředu, dozadu, vlevo, vpravo nebo jeho zastavení.

```

56 void websocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {
57     switch(type) {
58         case WStype_DISCONNECTED:
59             Serial.println("Disconnected from WebSocket");
60             break;
61         case WStype_CONNECTED:
62             Serial.println("Connected to WebSocket");
63             break;
64         case WStype_TEXT:
65             String message = String((char*)payload);
66             if (message == "forward") {
67                 moveForward();
68             } else if (message == "backward") {
69                 moveBackward();
70             } else if (message == "left") {
71                 turnLeft();
72             } else if (message == "right") {
73                 turnRight();
74             } else if (message == "stop") {
75                 stopMotors();
76             }
77             break;
78     }
79 }

```

Obrázek 35: Zpracování příkazů přijatých přes WebSocker komunikaci

Na obrázcích níže (Obrázek 36, 37, 38, 39, 40) jsou znázorněny pohybové funkce vozidla. Pomocí těchto funkcí je realizován pohyb dopředu (moveForward), dozadu (moveBackward), vlevo (turnLeft), vpravo (turnRight) a zastavení motorů (stopMotors). Informace o aktuálním stavu jsou současně vypisovány do sériového monitoru pro účely ladění.

```

81 void moveForward() {
82     digitalWrite(ENA, HIGH);
83     digitalWrite(ENB, HIGH);
84     digitalWrite(MOTORA_1, HIGH);
85     digitalWrite(MOTORA_2, LOW);
86     digitalWrite(MOTORB_1, LOW);
87     digitalWrite(MOTORB_2, HIGH);
88     Serial.println("Moving forward");
89 }

```

Obrázek 36: Funkce pro pohyb vozidla vpřed

```
91 void moveBackward() {  
92     digitalWrite(ENA, HIGH);  
93     digitalWrite(ENB, HIGH);  
94     digitalWrite(MOTORA_1, LOW);  
95     digitalWrite(MOTORA_2, HIGH);  
96     digitalWrite(MOTORB_1, HIGH);  
97     digitalWrite(MOTORB_2, LOW);  
98     Serial.println("Moving backward");  
99 }
```

Obrázek 37: Funkce pro pohyb vozidla vzad

```
101 void turnLeft() {  
102     digitalWrite(ENA, HIGH);  
103     digitalWrite(ENB, HIGH);  
104     digitalWrite(MOTORA_1, HIGH);  
105     digitalWrite(MOTORA_2, LOW);  
106     digitalWrite(MOTORB_1, HIGH);  
107     digitalWrite(MOTORB_2, LOW);  
108     Serial.println("Turning left");  
109 }
```

Obrázek 38: Funkce pro pohyb vozidla vlevo

```
111 void turnRight() {  
112     digitalWrite(ENA, HIGH);  
113     digitalWrite(ENB, HIGH);  
114     digitalWrite(MOTORA_1, LOW);  
115     digitalWrite(MOTORA_2, HIGH);  
116     digitalWrite(MOTORB_1, LOW);  
117     digitalWrite(MOTORB_2, HIGH);  
118     Serial.println("Turning right");  
119 }
```

Obrázek 39: Funkce pro pohyb vozidla vpravo

```
121 void stopMotors() {  
122     digitalWrite(ENA, LOW);  
123     digitalWrite(ENB, LOW);  
124     digitalWrite(MOTORA_1, LOW);  
125     digitalWrite(MOTORA_2, LOW);  
126     digitalWrite(MOTORB_1, LOW);  
127     digitalWrite(MOTORB_2, LOW);  
128     Serial.println("Stopped");  
129 }
```

Obrázek 40: Funkce pro zastavení vozidla

Poslední část pro kód psaný v Arduino IDE je funkce `getHTML`, pomocí které je generována jednoduchá webová stránka pro ovládání vozidla (Obrázek 41). Stránka obsahuje JavaScript, kterým jsou pomocí WebSocket připojení odesílány příkazy na základě stisknutí šipek na klávesnici. Při stisknutí příslušné klávesy je odeslán příkaz pro pohyb vozidla, při uvolnění klávesy je odeslán příkaz k zastavení. Tato část projektu byla využita

jako důležitý mezikrok při vývoji plně autonomního vozidla. Pomocí manuálního ovládání bylo v této fázi umožněno testování správné funkce motorů, odezvu vozidla na příkazy a stabilitu bezdrátové komunikace přes WiFi síť. Ovládání pomocí šipek bylo využito jako pokračování testování, které původně začínalo jednodušším řízením přes mobilní aplikaci Blynk.

```

131 String getHTML() {
132     return "<!DOCTYPE html>"
133         "<html><head><title>Ovladani auta</title>"
134         "<script>"
135         "var ws = new WebSocket('ws://192.168.1.x:81');" // konkrétní IP adresa Wemos zařízení
136         "ws.onopen = function() {"
137         "    console.log('WebSocket connection established');"
138         "};"
139         "ws.onerror = function(error) {"
140         "    console.log('WebSocket Error: ' + error);"
141         "};"
142         "document.addEventListener('keydown', function(event) {"
143         "    if(event.key === 'ArrowUp') ws.send('forward');"
144         "    if(event.key === 'ArrowDown') ws.send('backward');"
145         "    if(event.key === 'ArrowLeft') ws.send('left');"
146         "    if(event.key === 'ArrowRight') ws.send('right');"
147         "});"
148         "document.addEventListener('keyup', function(event) {"
149         "    ws.send('stop');"
150         "});"
151         "</script></head><body>"
152         "<h1>Ovladani auta</h1>"
153         "<p>&#8593; - dopredu, &#8595; - dozadu, &#8594; - doprava, &#8596; - doleva</p>"
154         "</body></html>";
155     }

```

Obrázek 41: Generování HTML stránky pro ovládání vozidla

5.2 Python kód

Zpracování obrazu z kamery a následné rozhodování o pohybu vozidla je zajišťováno pomocí Python skriptu, který představuje logickou řídicí vrstvu celého systému. Tento skript interpretuje vizuální informace, které jsou snímány kamerou, a převádí je na konkrétní akce – jako je zatáčení, jízda vpřed nebo zastavení. Pro detekci objektů v obraze jsou využívány ArUco markery, které umožňují přesné určení polohy vozidla, navigačních bodů a překážek. Na základě jejich vzájemné polohy a orientace je určován způsob pohybu vozidla v daném okamžiku. Komunikace s vozidlem probíhá prostřednictvím WebSocket rozhraní, jehož prostřednictvím jsou v reálném čase odesílány řídicí příkazy. V kódu je dále implementován systém pro detekci překážek v definované oblasti před vozidlem a také logika pro objíždění těchto překážek. Součástí systému je rovněž měření času jízdy mezi checkpointem a cílem, které slouží k následnému testování a vyhodnocování výkonu systému.

V této podkapitole je uveden konkrétní kód první části. Pro větší přehlednost je rozdělen do jednotlivých sekcí, přičemž každá z nich obsahuje popis funkcionality a doprovodný obrázek s příslušnou částí kódu. Celý kód je uveden v přílohách této práce (Příloha P II: Python kód).

V úvodní části Python skriptu jsou importovány potřebné knihovny, vytvořena třída pro WebSocket komunikaci a inicializovány všechny důležité proměnné. Pomocí třídy WebSocketCar je navázáno spojení s vozidlem, čímž je umožněno odesílání řídicích příkazů během jízdy. Dále jsou načteny různé ArUco slovníky s jejich konkrétními ID – pro označení vozidla, navigačních bodů a překážek. Jsou nastaveny parametry, jako je minimální vzdálenost k cíli (`STOP_DISTANCE`), tolerance směru (`ANGLE_TOLERANCE`), a inicializovány proměnné používané během celého běhu programu (například trajektorie, stav objížďky nebo kolizní zóna) (Obrázek 42).

```

1  import cv2
2  import numpy as np
3  import websocket
4  import time
5  from cv2 import aruco
6
7  class WebSocketCar:
8      def __init__(self, url):
9          self.ws = websocket.WebSocket()
10         self.ws.connect(url)
11
12         def send(self, command):
13             print(f"[SEND] {command}")
14             self.ws.send(command)
15
16 car = WebSocketCar("ws://192.168.137.35:81")
17
18 aruco_dict_auto = aruco.getPredefinedDictionary(aruco.DICT_5X5_250)
19 aruco_dict_nav = aruco.getPredefinedDictionary(aruco.DICT_4X4_50)
20 aruco_dict_obstacles = aruco.getPredefinedDictionary(aruco.DICT_5X5_100)
21 parameters = aruco.DetectorParameters()
22
23 AUTO_ID = 0
24 CHECKPOINT_ID = 39
25 GOAL_ID = 14
26 OBSTACLE_ID = 11
27
28 cap = cv2.VideoCapture(1)
29
30 STOP_DISTANCE = 50
31 ANGLE_TOLERANCE = 15
32 step = 1
33 trajectory = []
34
35 last_nav_pos = None
36 nav_pos = None
37 obstacle_positions = []
38
39 detour_active = False
40 detour_point = None
41
42 ZONE_LENGTH = 150
43 ZONE_WIDTH_FRONT = 100
44 ZONE_WIDTH_BACK = 40
45
46 zone_polygon = None
47
48 start_time = None
49 travel_time = None

```

Obrázek 42: Inicializace knihoven, komunikace a základních parametrů

Následující část kódu zobrazená na snímku níže (Obrázek 43) obsahuje sadu pomocných funkcí, jimiž jsou vypočítávány hodnoty potřebné pro řízení vozidla. Funkce `get_marker_center` a `get_marker_angle` slouží ke zjištění středu a orientace detekovaného ArUco markeru.

Další z funkcí je `angle_difference`, prostřednictvím které je určován minimální úhlový rozdíl mezi aktuálním směrem vozidla a požadovaným směrem k cíli. Nejrozsáhlejší z funkcí je `is_in_collision_zone`, jejímž úkolem je výpočet tzv. kolizní zóny na základě aktuální polohy a směru vozidla. Touto zónou je míněná oblast před vozidlem, ve které je kontrolován výskyt překážek. Pokud se v této oblasti nachází

detekovaný objekt, je funkcí vrácena hodnota `True` a následně je aktivována logika vyhýbání.

```

51 def get_marker_center(corners):
52     x = int(np.mean(corners[:, 0]))
53     y = int(np.mean(corners[:, 1]))
54     return (x, y)
55
56 def get_marker_angle(corners):
57     top_left, top_right = corners[0], corners[1]
58     dx = top_right[0] - top_left[0]
59     dy = top_right[1] - top_left[1]
60     angle = np.arctan2(dy, dx) * 180 / np.pi
61     return angle % 360
62
63 def angle_difference(a, b):
64     diff = (b - a + 360) % 360
65     if diff > 180:
66         diff -= 360
67     return diff
68
69 def is_in_collision_zone(auto_pos, auto_angle, obstacle_pos):
70     global zone_polygon
71
72     angle_rad = np.deg2rad(auto_angle)
73     dir_vec = np.array([np.cos(angle_rad), np.sin(angle_rad)])
74     side_vec = np.array([-dir_vec[1], dir_vec[0]])
75
76     a = np.array(auto_pos)
77     b1 = a + dir_vec * ZONE_LENGTH + side_vec * (ZONE_WIDTH_FRONT / 2)
78     b2 = a + dir_vec * ZONE_LENGTH - side_vec * (ZONE_WIDTH_FRONT / 2)
79     a1 = a + side_vec * (ZONE_WIDTH_BACK / 2)
80     a2 = a - side_vec * (ZONE_WIDTH_BACK / 2)
81
82     pts = np.array([a1, a2, b2, b1], dtype=np.int32)
83     zone_polygon = pts.copy()
84
85     obstacle_float = tuple(map(float, obstacle_pos))
86     is_inside = cv2.pointPolygonTest(pts, obstacle_float, False) >= 0
87     print(f"[DEBUG] Kolizní test □ obstacle {obstacle_pos} in zone: {is_inside}")
88     return is_inside

```

Obrázek 43: Pomocné funkce pro výpočet pozice, úhlů a kolizní zóny

Na snímku níže je znázorněna část hlavní smyčky programu, v jejímž rámci je zajišťováno zpracování obrazu z kamery a detekce ArUco markerů (Obrázek 44). Nejprve je obraz převeden do odstínů šedi, poté je provedena detekce markerů představujících vozidlo, navigační body a překážky. Na základě rozpoznaného ID je vyhledáno vozidlo a určeny jeho aktuální souřadnice a směr. Tyto informace jsou využívány pro vykreslování pozice vozidla do obrazu a pro zaznamenávání jeho trajektorie. V závislosti na fázi jízdy (směr k checkpointu nebo k cíli) je nastaven příslušný cílový marker, jehož pozice je zjišťována v samostatném bloku níže.

```

90 while True:
91     ret, frame = cap.read()
92     if not ret:
93         break
94
95     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
96
97     corners_auto, ids_auto, _ = aruco.detectMarkers(gray, aruco_dict_auto, parameters=parameters)
98     corners_nav, ids_nav, _ = aruco.detectMarkers(gray, aruco_dict_nav, parameters=parameters)
99     corners_obs, ids_obs, _ = aruco.detectMarkers(gray, aruco_dict_obstacles, parameters=parameters)
100
101     auto_pos, auto_angle = None, None
102     target_id = CHECKPOINT_ID if step == 1 else GOAL_ID
103     zone_polygon = None
104     obstacle_positions = []
105
106     if ids_auto is not None:
107         for i, id in enumerate(ids_auto.flatten()):
108             if id == AUTO_ID:
109                 corners = corners_auto[i][0]
110                 auto_pos = get_marker_center(corners)
111                 trajectory.append(auto_pos)
112                 auto_angle = (get_marker_angle(corners) + 90) % 360
113
114                 cv2.circle(frame, auto_pos, 6, (255, 0, 0), -1)
115
116                 arrow_len = 60
117                 angle_rad = np.deg2rad(auto_angle)
118                 end_x = int(auto_pos[0] + arrow_len * np.cos(angle_rad))
119                 end_y = int(auto_pos[1] + arrow_len * np.sin(angle_rad))
120                 cv2.arrowedLine(frame, auto_pos, (end_x, end_y), (0, 255, 0), 2)
121
122
123     if ids_nav is not None:
124         for i, id in enumerate(ids_nav.flatten()):
125             corners = corners_nav[i][0]
126             if id == target_id:
127                 nav_pos = get_marker_center(corners)
128                 if not detour_active:
129                     last_nav_pos = nav_pos

```

Obrázek 44: Detekce markerů a určení pozice vozidla

Na obrázku 45 je znázorněn úsek programu zaměřený na rozpoznávání překážek a jejich zpracování. Jakmile jsou v obraze detekovány markery se specifickým ID určeným pro překážky, jejich pozice jsou uloženy a vizuálně zvýrazněny v aktuálním snímku. Následně je prováděn test, zda se některá z překážek nachází v předem definované kolizní zóně před vozidlem. V případě pozitivního výsledku je aktivován proces výpočtu objízdného bodu. Tento bod je určen pomocí vektorového výpočtu kolmého směru vůči přímce pohybu, a vozidlo je následně přesměrováno na nový bod mimo překážku. Po dokončení objížd'ky je detekce překážek deaktivována a vozidlu je umožněno pokračovat v jízdě k původnímu cíli.

```

131     if ids_obs is not None:
132         for i, id in enumerate(ids_obs.flatten()):
133             if id == OBSTACLE_ID:
134                 corners = corners_obs[i][0]
135                 pos = get_marker_center(corners)
136                 obstacle_positions.append(pos)
137                 cv2.circle(frame, pos, 6, (0, 0, 255), -1)
138                 cv2.putText(frame, "PREKAZKA", (pos[0]+10, pos[1]),
139                             cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
140
141     if not detour_active and auto_pos and auto_angle and obstacle_positions:
142         for pos in obstacle_positions:
143             if is_in_collision_zone(auto_pos, auto_angle, pos):
144                 offset = 150
145                 angle_rad = np.deg2rad(auto_angle)
146                 dir_vec = np.array([np.cos(angle_rad), np.sin(angle_rad)])
147                 side_vec = np.array([-dir_vec[1], dir_vec[0]])
148
149                 rel_vec = np.array(pos) - np.array(auto_pos)
150                 side_sign = np.sign(np.cross(dir_vec, rel_vec))
151                 if side_sign == 0:
152                     side_sign = 1
153
154                 detour_point = tuple((np.array(pos) + side_vec * offset * side_sign).astype(int))
155                 last_nav_pos = detour_point
156                 detour_active = True
157                 print(f"[INFO] PREKAZKA V ZONE ☐ OBJÍZDIM PREKOZKU NA BOD: {detour_point}")
158                 break
159
160     if zone_polygon is not None:
161         cv2.polylines(frame, [zone_polygon], True, (0, 255, 255), 2)
162
163     if detour_active and auto_pos and detour_point:
164         if np.linalg.norm(np.array(auto_pos) - np.array(detour_point)) < STOP_DISTANCE:
165             print(f"[INFO] Objízdka hotova ☐ pokračuju na cíl")
166             last_nav_pos = nav_pos
167             detour_active = False
168

```

Obrázek 45: Detekce překážek a výpočet objízdného bodu

Část skriptu vyobrazená níže (Obrázek 46) se zaměřuje na výpočet směrového úhlu mezi aktuální pozicí vozidla a cílovým bodem, kterým může být checkpoint nebo koncový bod trasy. Výsledný úhel je porovnáván s aktuální orientací vozidla a na základě vzniklého rozdílu je určován další směr pohybu. V dalších částech kódu jsou postupně vykreslovány různé vizuální prvky – například šipka znázorňující směr jízdy, trajektorie ujetého úseku, směrové úhly, vzdálenost k cíli nebo souřadnice jednotlivých objektů. Tyto informace slouží k vizuální kontrole chování systému během provozu. Testovací blok je určen k měření a zobrazení aktuálního času jízdy, přičemž časomíra je spuštěna po projetí checkpointem a zastavena při dosažení cíle. Tento údaj je využíván pro účely experimentu popsaného v následující kapitole.

```

168 | if last_nav_pos is not None and auto_pos and auto_angle is not None:
169 |     dx = last_nav_pos[0] - auto_pos[0]
170 |     dy = last_nav_pos[1] - auto_pos[1]
171 |     target_angle = (np.arctan2(dy, dx) * 180 / np.pi) % 360
172 |     angle_diff = angle_difference(auto_angle, target_angle)
173 |     distance = np.linalg.norm(np.array(last_nav_pos) - np.array(auto_pos))
174 |
175 |     cv2.arrowedLine(frame, auto_pos, last_nav_pos, (255, 0, 0), 2)
176 |     if len(trajjectory) > 1:
177 |         cv2.polylines(frame, [np.array(trajjectory, dtype=np.int32)], False, (0, 0, 255), 2)
178 |
179 |     cv2.putText(frame, f"Step: {step}", (10, 15), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 255, 100), 2)
180 |     cv2.putText(frame, f"Car angle: {auto_angle:.2f}", (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (200, 200, 0), 2)
181 |     cv2.putText(frame, f"Target angle: {target_angle:.2f}", (10, 70), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (200, 200, 0), 2)
182 |     cv2.putText(frame, f"Angle diff: {angle_diff:.2f}", (10, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (200, 200, 0), 2)
183 |     cv2.putText(frame, f"Distance: {distance:.2f}", (10, 130), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (200, 200, 0), 2)
184 |
185 |     if auto_pos:
186 |         cv2.putText(frame, f"AUTO: {auto_pos}", (10, 160), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
187 |     if nav_pos:
188 |         nav_label = "CHECKPOINT" if step == 1 else "CIL"
189 |         cv2.putText(frame, f"{nav_label}: {nav_pos}", (10, 190), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
190 |     if obstacle_positions:
191 |         for i, pos in enumerate(obstacle_positions):
192 |             cv2.putText(frame, f"PREKAZKA: {pos}", (10, 220 + i*25), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)
193 |
194 |     #Zobrazení času
195 |     if travel_time is not None:
196 |         cv2.putText(frame, f"Cas: {travel_time:.2f}s", (10, 280), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 255, 255), 2)
197 |     elif start_time is not None:
198 |         elapsed_time = time.time() - start_time
199 |         cv2.putText(frame, f"Cas: {elapsed_time:.2f}s", (10, 280), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 255, 255), 2)

```

Obrázek 46: Výpočet směru a zobrazení informací v obraze

Závěrečná část algoritmu určuje konkrétní akci vozidla na základě aktuální vzdálenosti a směrového rozdílu vůči cílovému bodu. Pokud je rozdíl úhlu větší než stanovená tolerance, je aktivováno otáčení vozidla. V opačném případě je vozidlu umožněno pokračovat v přímém pohybu, nebo se zastavuje, pokud se nachází v dostatečné blízkosti cíle. Po dosažení checkpointu je systém automaticky přepnut do další fáze jízdy směrem k cílovému bodu a zároveň je spuštěna časomíra. Při příjezdu do cíle je časomíra zastavena a aktuální hodnota je vypsána do výstupu. Skript zároveň zajišťuje zobrazení okna s náhledem a umožňuje jeho ukončení pomocí stisku klávesy q. Tato část uzavírá celý cyklus řízení a vizuálního zobrazení vozidla (Obrázek 47).

```

201         if abs(angle_diff) > ANGLE_TOLERANCE:
202             car.send("right" if angle_diff > 0 else "left")
203         else:
204             if distance < STOP_DISTANCE:
205                 car.send("stop")
206                 if step == 1:
207                     print("[INFO] Dosahl checkpointu ☐ pokračuji na cil")
208                     step = 2
209                     start_time = time.time()
210                 elif step == 2 and not detour_active:
211                     if travel_time is None and start_time is not None:
212                         travel_time = time.time() - start_time
213                         print(f"[INFO] CIL DOSAŽEN! Cas od checkpointu: {travel_time:.2f} s")
214                     print("[INFO] Auto zastaveno, pripraveno na dalsi akci.")
215                     start_time = None
216                 else:
217                     car.send("forward")
218             else:
219                 car.send("stop")
220                 cv2.putText(frame, "Hledam auto nebo cil...", (10, 250), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
221
222             cv2.imshow("Auto View", frame)
223             if cv2.waitKey(1) & 0xFF == ord('q'):
224                 break
225
226         cap.release()
227         cv2.destroyAllWindows()

```

Obrázek 47: Řízení jízdy, měření času a ukončení programu

5.3 Kamerový systém

V této podkapitole je popsán kamerový systém, jehož úkolem je vizuální sledování prostoru a poskytování klíčových dat pro navigaci autonomního vozidla. Pro snímání scény je použita standardní webkamera připojená k počítači, která zaznamenává dění z horního pohledu. Pomocí knihovny OpenCV a technologie ArUco markerů jsou z obrazu extrahovány informace o poloze a orientaci vozidla, stejně jako o pozici navigačních bodů.

Na základě těchto údajů je v reálném čase vypočítáván potřebný směr pohybu a vozidlo je ovládáno prostřednictvím WebSocket komunikace. Detekovaný úhel orientace vozidla je porovnáván s úhlem směrem k cílovému bodu, a podle vzniklého rozdílu je určeno, zda má být vozidlo natočeno, nebo zda má pokračovat v přímém směru. V rámci systému je rovněž implementováno sledování trajektorie pohybu vozidla a logika zajišťující postupné dosažení jednotlivých navigačních bodů ve správném pořadí.

5.3.1 Umístění a konfigurace kamery

Pro účely vizuálního sledování prostoru byla kamera umístěna nad sledovanou plochu tak, aby byl snímán celý prostor, ve kterém se vozidlo pohybuje. Kamera je připevněna k pevné konstrukci, čímž je zajištěno stabilní uchycení bez nežádoucího pohybu či vibrací během provozu. Stabilita obrazu je klíčová pro správné rozpoznávání ArUco markerů i spolehlivou detekci překážek v prostoru.

Použitá webkamera Eternico ET201 disponuje dostatečným rozlišením pro detekci značek a objektů, a to i při menších velikostech nebo při jejich větší vzdálenosti od kamery. Technické parametry kamery jsou následující:

- rozlišení: 1920×1080 px (Full HD),
- snímková frekvence: 30 snímků za sekundu (fps),
- typ připojení: USB 2.0 (Plug & Play),
- zorné pole: 106° .

Kamera byla nastavena tak, aby byl minimalizován obrazový šum a zároveň zachována dostatečná ostrost a kontrast potřebné pro spolehlivé zpracování obrazu pomocí algoritmů knihovny OpenCV. Osvětlení scény bylo rovněž upraveno s cílem eliminovat přesvětlené nebo naopak příliš tmavé oblasti, které by mohly negativně ovlivnit kvalitu detekce.

5.3.2 Význam horního pohledu pro navigaci

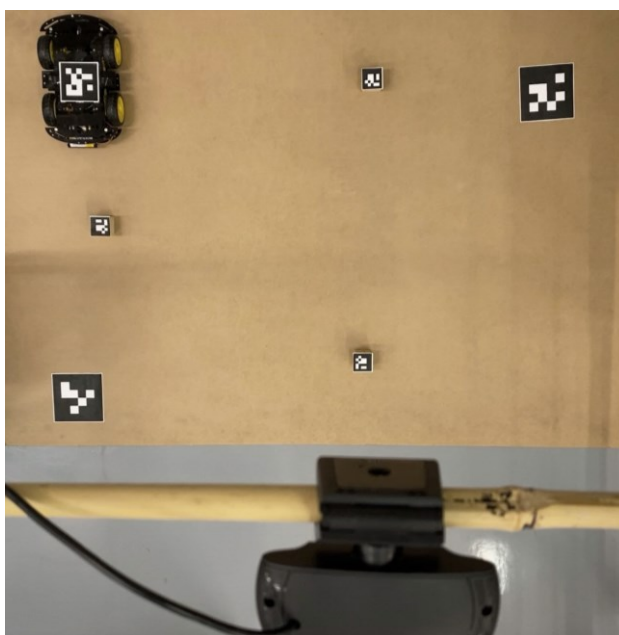
Použití horního pohledu při snímání prostoru se ukázalo jako velmi výhodné pro zajištění správné navigace vozidla. Tento způsob snímání výrazně zjednodušuje vyhodnocování polohy vozidla a překážek, neboť všechny objekty jsou zachyceny v jedné rovině bez zkreslení způsobeného perspektivou. Díky tomu mohou být jednoduše určeny aktuální souřadnice i orientace vozidla a prováděny potřebné výpočty pro jeho navigaci.

Významnou výhodou horního pohledu je také vysoká stabilita obrazu. Na rozdíl od řešení, při němž je kamera umístěna přímo na vozidle, nedochází v tomto případě k žádnému kolísání ani natáčení obrazu během pohybu. Tím je eliminována nutnost složité kompenzace vibrací a usnadněna detekce markerů i překážek. Horní pohled se v rámci tohoto systému osvědčil jako praktický a efektivní přístup k zajištění plynulé jízdy bez rušení obrazových dat.

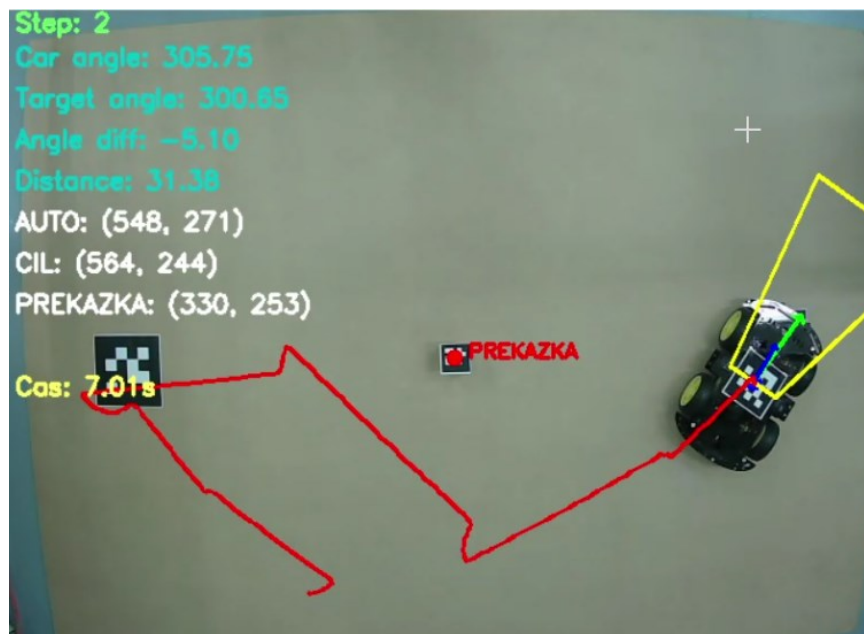
6 NÁVRH EXPERIMENTU

Cílem experimentu je ověření funkčnosti a spolehlivosti navrženého systému autonomního řízení vozidla v reálných podmínkách testovacího prostředí. Experiment byl navržen tak, aby byly prověřeny všechny klíčové části systému – od detekce polohy vozidla, přes rozpoznávání překážek, až po plánování a realizaci trasy k cílovému bodu.

Testovací scénář je založen na sledování prostoru z horního pohledu pomocí pevně uchycené kamery a na vyhodnocování obrazu v reálném čase prostřednictvím knihovny OpenCV (Obrázek 48). Vozidlo je vybaveno ArUco markerem, který umožňuje přesné určení jeho aktuální polohy a orientace. Pro účely navigace byly rovněž využity ArUco kódy sloužící k označení kontrolních bodů (tzv. checkpointů) a cílového místa. Kontrolní bod je označen jako bod A a cílový bod jako bod B. Překážky v prostoru jsou opatřeny ArUco kódem, na jehož základě je prováděna jejich detekce a následně plánovány vyhýbací manévry. Během jízdy jsou překážky detekovány, jejich poloha je analyzována vzhledem k plánované trase a trajektorie vozidla je upravována tak, aby se překážkám bylo možné bezpečně vyhnout. V rámci vizualizace je současně zaznamenávána trajektorie pohybu vozidla, která je průběžně vykreslována v podobě červené čáry zobrazující již ujetou trasu (Obrázek 49). Navigační algoritmus využívá princip porovnávání směru vozidla k cíli a inteligentního rozhodování na základě aktuální situace v prostředí. V této kapitole je popsáno využití ArUco kódů pro navigaci, charakteristika překážek, se kterými vozidlo interaguje, a princip činnosti hlavního algoritmu, jenž řídí pohyb vozidla k cílovému bodu.



Obrázek 48: Horní pohled – vizualizace projektu

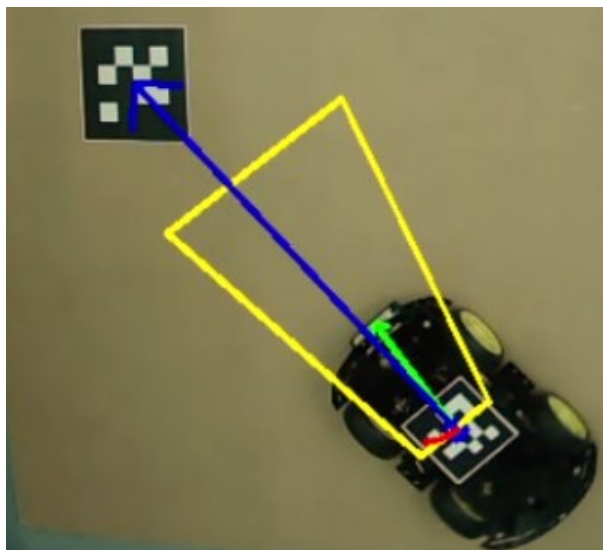


Obrázek 49: Zobrazení ujeté trasy

6.1 ArUco markery

Pro praktickou realizaci navigace byla zvolena metoda využívající ArUco markery, které umožňují přesnou detekci a rozlišení jednotlivých objektů v prostoru. Každý klíčový objekt testovací scény byl označen specifickým markerem s vlastním identifikátorem (dále jen „ID“). Při vyhodnocování obrazu v reálném čase je sledována aktuální poloha vozidla a současně identifikován navigační bod, který je v daném okamžiku určen jako cíl (Obrázek 50). Tím je zajištěno, že vozidlo nejprve směřuje k checkpointu, a po jeho dosažení automaticky pokračuje k cílovému bodu.

Všechny překážky v prostředí byly opatřeny jednotným ID, aby mohly být systémem odlišeny od navigačních bodů. Detekce překážek umožňuje průběžnou úpravu plánované trajektorie a efektivní vyhýbání se kolizím během jízdy. Použití ArUco markerů významně přispělo ke zjednodušení detekčního procesu a ke zvýšení spolehlivosti navigace v dynamickém prostředí.



Obrázek 50: Snímání ArUco markeru

6.2 Překážky

Překážky umístěné v testovacím prostoru byly tvořeny modely malých domečků. Jejich účelem bylo ověření schopnosti vozidla detekovat neplánované objekty na trase a bezpečně se jim vyhýbat. Markery byly na dřevěných domečcích připevněny šikmo, což v některých případech způsobovalo problémy při jejich správném rozpoznávání, zejména při větší vzdálenosti nebo nepříznivém úhlu pohledu kamery. Z důvodu nižší kvality kamery byly překážky upraveny tak, aby byly snímány co nejvodorovněji. Markery byly proto nově umístěny na dřevěné krychle, čímž byla výrazně zlepšena jejich detekce a stabilita rozpoznávání v obraze (Obrázek 51).



Obrázek 51: Horní pohled na překážky

6.3 Algoritmus

Hlavní algoritmus řídící pohyb vozidla byl navržen tak, aby umožňoval autonomní navigaci mezi jednotlivými navigačními body s automatickou detekcí překážek a plánováním objízdne trasy.

Po spuštění systému je nejprve vyhledán první navigační bod (checkpoint), ke kterému je vozidlo orientováno na základě vypočítaného směrového vektoru mezi jeho aktuální polohou a polohou checkpointu. Vozidlo je natočeno do požadovaného směru, a následně je uvedeno do pohybu směrem vpřed. Po dosažení checkpointu je navigace automaticky přesměrována k cílovému bodu. V průběhu jízdy je obraz nepřetržitě vyhodnocován a je kontrolováno, zda se v předem definované kolizní zóně před vozidlem nenachází překážka. Kolizní zóna je vymezena jako lichoběžníková oblast před vozidlem, v jejímž rámci je prováděna detekce překážek pomocí ArUco markerů. V případě, že je překážka detekována, je vypočítán objízdny bod pomocí normálového vektoru na trasu mezi aktuální pozicí vozidla a cílovým bodem. Vozidlo je natočeno směrem k tomuto bodu, je navedeno k objetí překážky, a poté je navraceno zpět na původní přímou trasu vedoucí k cíli. Celý proces je řízen v reálném čase na základě aktuálních údajů získávaných z webkamery a probíhá plně autonomně, bez nutnosti zásahu operátora.

6.4 Plánované testovací scénáře

Pro ověření funkčnosti algoritmu autonomní jízdy byly navrženy konkrétní testovací scénáře, které odrážejí reálné situace, s postupně rostoucí úrovní složitosti. Každý scénář kombinuje různý počet překážek, průjezd kontrolním bodem (checkpointem) a dosažení cílového bodu, přičemž je ověřována schopnost vozidla adekvátně reagovat na překážky v různých pozicích a včas upravovat trajektorii pohybu.

Scénáře byly sestaveny tak, aby pokrývaly základní situace, ve kterých je nezbytné detekovat překážku, aktivovat objízdny manévr a pokračovat dále k vytyčenému bodu. Hlavním cílem bylo ověřit funkčnost reakce systému na konkrétní překážku umístěnou v kolizní zóně a schopnost vozidla tuto překážku objet na základě výpočtu odkloněného směru jízdy.

Přehled navržených scénářů

- Scénář 1 – jedna překážka mezi checkpointem a cílem: checkpoint a cíl jsou od sebe vzdáleny přibližně jeden metr, překážka je umístěna uprostřed mezi nimi.

- Scénář 2 – dvě překážky mezi startem a cílem: první překážka se nachází mezi startem a checkpointem, druhá mezi checkpointem a cílem.
- Scénář 3 – dvě překážky vedle sebe mezi checkpointem a cílem: obě překážky jsou umístěny vedle sebe v úseku mezi checkpointem a cílem.
- Scénář 4 – tři překážky se startem vozidla přímo proti překážce: vozidlo startuje přímo proti překážce, další dvě překážky se nachází na trase mezi checkpointem a cílem.

7 TESTOVÁNÍ A VYHODNOCENÍ FUNKČNOSTI

V této kapitole jsou uvedeny testovací scénáře navržené za účelem ověření funkčnosti a spolehlivosti autonomního systému řízení vozidla. Cílem testování bylo prověřit jednotlivé části navrženého řešení, a to jak z hlediska správného rozpoznání objektů v prostoru, tak i z pohledu reakce vozidla na překážky, dosažení cílového bodu a celkové schopnosti dokončit trasu bez zásahu uživatele.

Součástí testování bylo rovněž měření času jízdy v předem definovaném úseku mezi checkpointem a cílem (od bodu A k bodu B). Toto měření sloužilo k orientačnímu vyhodnocení výkonnosti systému a stability algoritmu v prostředí s překážkami. V následujících podkapitolách jsou popsány jednotlivé testovací scénáře, průběh jejich realizace, zaznamenané výsledky a zhodnocení funkčnosti systému na základě pozorování i naměřených dat.

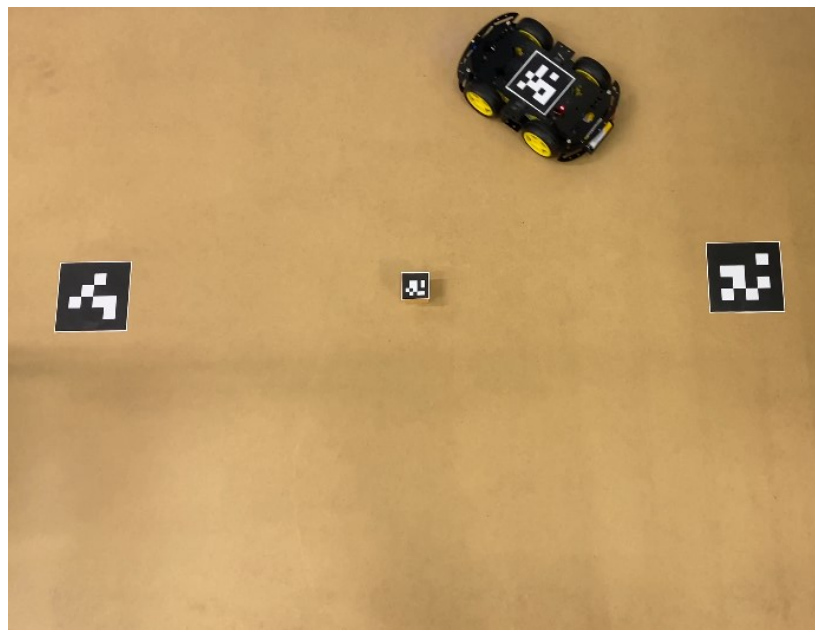
7.1 Scénáře testování

Na základě návrhu uvedeného v kapitole 6.4 bylo přistoupeno k praktickému testování všech čtyř scénářů. Každý scénář byl testován opakovaně a jeho průběh byl vizuálně zaznamenán (Příloha P III: Záznam scénářů). Následující popis a vyobrazení shrnují průběh jednotlivých situací a chování vozidla v daných podmínkách.

Scénář 1 – Jedna překážka mezi checkpointem a cílem

V tomto scénáři byly checkpoint a cíl umístěny přibližně jeden metr od sebe. Překážka byla umístěna přímo na spojnici těchto bodů (Obrázek 52). Po projetí checkpointem byla překážka detekována v kolizní zóně, čímž došlo k aktivaci objízdného režimu a výpočtu nového směru jízdy pomocí odkloněného vektoru. Po úspěšném objetí překážky bylo vozidlo navráceno zpět do původního směru a následně bezpečně dorazilo do cílového bodu.

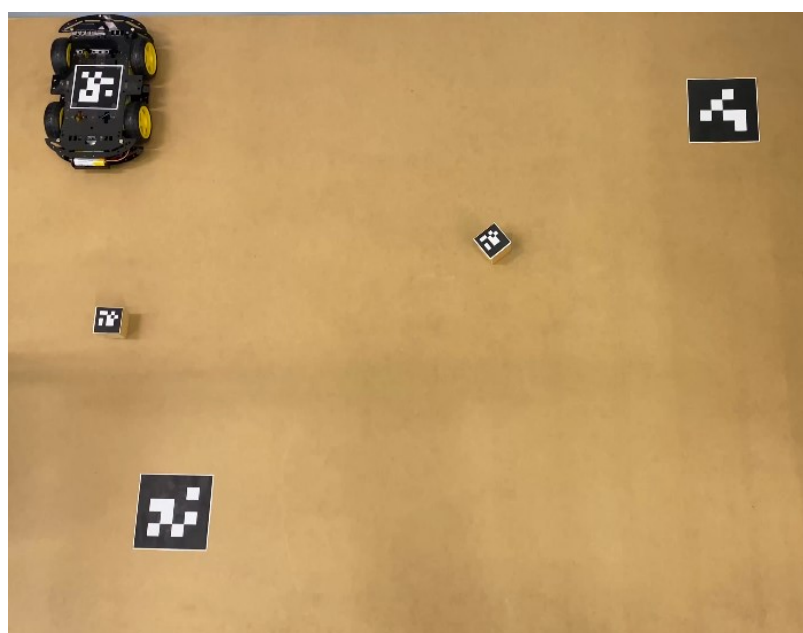
Tento scénář byl využit pro kvantitativní testování, při kterém bylo provedeno deset jízd. Měření času bylo zahájeno v okamžiku průjezdu checkpointem a ukončeno při dosažení cíle. Průměrný čas dosažený ve všech pokusech je uveden v následující podkapitole.



Obrázek 52: Scénář 1 – Překážka mezi checkpointem a cílem

Scénář 2 – Dvě překážky mezi startem a cílem

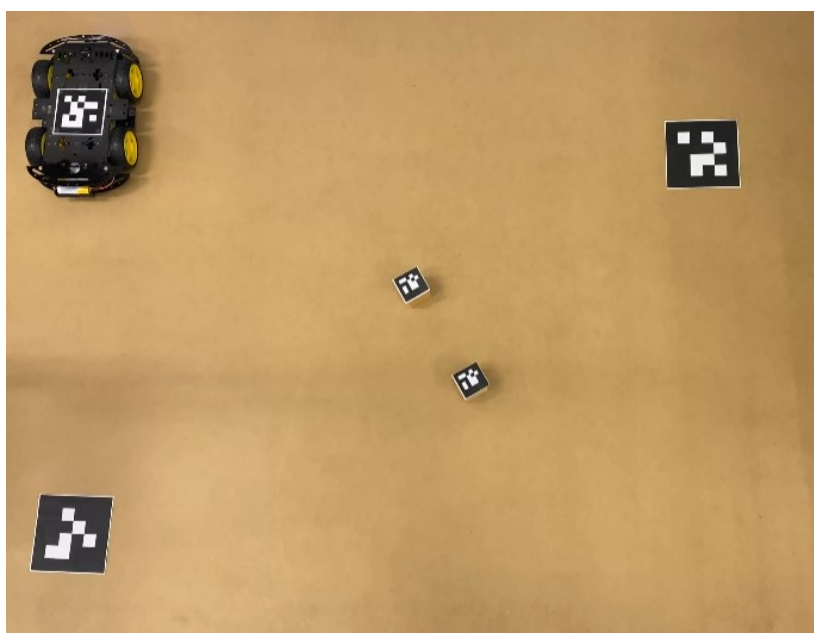
V tomto testu se nacházely dvě překážky – první byla umístěna mezi startem a checkpointem, druhá mezi checkpointem a cílovým bodem (Obrázek 53). První překážka byla úspěšně detekována a následně byl proveden objízdný manévr. Poté bylo vozidlo navedeno k průjezdu checkpointem a obdobná reakce následovala i v případě druhé překážky. Obě detekce byly systémem správně vyhodnoceny a celá trasa byla vozidlem dokončena bez nutnosti zásahu.



Obrázek 53: Scénář 2 – Překážky před a po checkpointu

Scénář 3 – Dvě překážky vedle sebe mezi checkpointem a cílem

V tomto scénáři byly dvě překážky umístěny těsně vedle sebe v prostoru mezi checkpointem a cílovým bodem (Obrázek 54). Po průjezdu checkpointem byla jedna z překážek detekována v kolizní zóně a byl aktivován objízdný manévr. Vzhledem k symetrickému umístění překážek vedle sebe nebylo rozhodující, ze které strany bude objíždění zahájeno. Test byl úspěšně realizován – vyhýbací manévr byl proveden správně a vozidlo bezpečně dorazilo do cílového bodu.

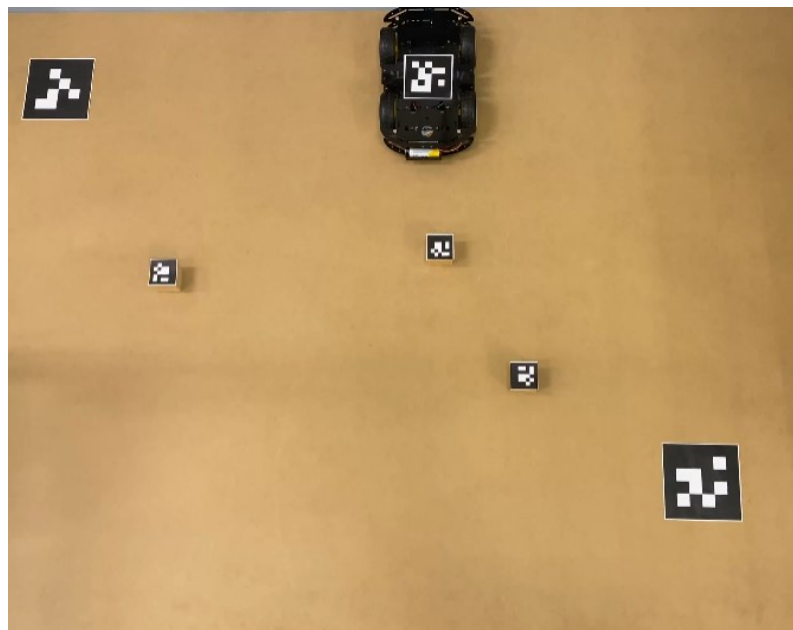


Obrázek 54: Scénář 3 – Dvojice překážek v cestě

Scénář 4 – Tři překážky se startem vozidla přímo proti překážce

Nejnáročnější scénář zahrnoval start vozidla umístěného přímo proti překážce, která se nacházela ve výchozím směru jízdy (Obrázek 55). Bezprostředně po spuštění systému byla překážka detekována v kolizní zóně a byl aktivován objízdný režim. Následně byl proveden vyhýbací manévr a vozidlo bylo navedeno směrem k checkpointu, kterým úspěšně projelo.

Ve druhé části trasy, mezi checkpointem a cílovým bodem, se nacházely dvě další překážky. Algoritmem byla detekována nejbližší z nich, následně byl aktivován další objízdný režim, který umožnil bezpečné pokračování v jízdě. Přestože se trajektorie mírně odchýlila od ideální přímky, test byl úspěšně dokončen – překážky byly bezpečně objety a cílový bod byl dosažen.



Obrázek 55: Scénář 4 – Kritická výchozí situace s více překážkami

7.2 Výsledky testování

Pro kvantitativní vyhodnocení funkčnosti algoritmu byl zvolen scénář s jednou překážkou umístěnou mezi checkpointem a cílovým bodem (viz Scénář 1). Tento scénář byl realizován desetkrát, přičemž byl měřen čas jízdy mezi průjezdem checkpointem a dosažením cílového bodu. Měření bylo prováděno automaticky na základě detekce průjezdu předem definovanými oblastmi v obraze. Naměřené časy jednotlivých pokusů jsou uvedeny v následující tabulce (Tabulka 1):

Tabulka 1: Časový záznam úseku

Číslo měření	Ujetá vzdálenost [m]	Čas měření [s]	Rychlost [m/s]
1	1	7,01	0,14
2	1	7,88	0,12
3	1	6,58	0,15
4	1	7,62	0,13
5	1	8,48	0,11
6	1	7,55	0,13
7	1	7,54	0,13
8	1	7,49	0,13
9	1	7,98	0,12
10	1	7,76	0,13
Průměr		7,59	0,13

Z výsledků je patrné, že dosažené časy jsou konzistentní a rozdíly mezi jednotlivými jízdami minimální. Algoritmus pro aktivaci a provedení objížděného manévru vykazoval stabilní chování, přičemž v žádném z pokusů nedošlo k přerušení jízdy ani k chybné reakci na překážku.

7.3 Identifikace problémů a jejich řešení

Během testování a vývoje autonomního systému se postupně objevilo několik problémů, které ovlivňovaly spolehlivost detekce a plynulost jízdy. Část těchto problémů souvisela s vnějšími podmínkami (např. osvětlením a odlesky), jiné byly způsobeny implementací algoritmu nebo omezeními hardwaru.

Identifikované problémy:

1. Odlesky podlahy představovaly zásadní problém zejména v případě lesklého, reflektujícího povrchu. Silné světelné odrazy vytvářely světlé oblasti v obraze, které narušovaly detekci překážek i ArUco markerů a vedly k chybným reakcím systému.
2. Zhoršené osvětlení způsobovalo snížení kontrastu v obraze, což vedlo k chybnému rozpoznání hran markerů. V důsledku toho docházelo k trhavému chování vozidla (tzv. poskakování), jelikož algoritmus střídavě ztrácel a znovu nacházel referenční body.
3. Nízká kvalita použité kamery se projevovala omezeným rozlišením, slabou ostrostí a nízkou citlivostí na světlo. Tyto faktory výrazně zhoršovaly detekci objektů zejména při větší vzdálenosti nebo rychlejším pohybu vozidla.
4. Výpadky detekce při šikmém natočení vozidla byly způsobeny deformací tvaru markeru v obraze, pokud bylo vozidlo natočeno pod ostrým úhlem vůči kameře. Tím docházelo k selhání při určování jeho polohy a směru.
5. Omezená reakce na vícenásobné překážky souvisela s tím, že algoritmus zpracovával vždy pouze jednu překážku aktivující kolizní zónu. Pokud se po objetí první překážky v novém směru objevil další objekt, nemusel být systém připraven adekvátně reagovat.
6. Obtížná detekce markerů na šikmých překážkách – původní překážky byly tvořeny malými dřevěnými domečky, jejichž ArUco markery byly připevněny na šikmé střešní ploše. To vedlo k výraznému zkreslení tvaru markeru v obraze a spolu s jeho

malými rozměry často způsobovalo selhání detekce. Marker nebyl včas rozpoznán nebo nebyl rozpoznán vůbec, což negativně ovlivňovalo funkčnost kolizní zóny.

Možnosti vylepšení:

1. Použití matného, neodrazivého povrchu (např. gumové nebo kartonové podložky) by výrazně omezilo výskyt světelných odlesků, které narušují detekční proces.
2. Zajištění rovnoměrného, stálého umělého osvětlení prostřednictvím LED panelů bez výrazných stínů by vedlo ke stabilnímu kontrastu v obraze. Dále by bylo možné zvýšit spolehlivost detekce markerů v horších světelných podmínkách úpravou algoritmu pro detekci hran.
3. Výměna kamery za kvalitnější model s vyšším rozlišením, kvalitnější optikou a vyšší světelností by vedla k ostřejšímu obrazu, lepší detekci detailů a vyšší spolehlivosti sledování při rychlejším pohybu.
4. Použití většího ArUco markeru na vozidle nebo nasazení více markerů na různých stranách karoserie by zlepšilo čitelnost i při šikmém natočení vůči kameře.
5. Rozšíření algoritmu o schopnost zpracovávat více překážek současně nebo aktualizace kolizní zóny v reálném čase by zvýšily odolnost systému vůči složitějším scénářům s vícenásobnými překážkami.
6. Přesun ArUco markerů ze šikmých střech na kolmé kostky orientované čelně ke kameře výrazně zlepšil úspěšnost detekce překážek. Tato úprava přinesla stabilnější rozpoznávání a vyšší spolehlivost kolizního systému.

ZÁVĚR

Tato bakalářská práce byla zaměřena na návrh, sestavení a softwarovou realizaci autonomního modelu vozidla, které je schopné samostatného pohybu v definovaném prostoru, detekce překážek a jejich bezpečného objíždění. Hlavním cílem bylo ověření, zda může být s využitím běžně dostupného hardwaru a open-source nástrojů navržen funkční systém autonomního řízení, jenž dokáže reagovat na okolní prostředí a dosáhnout stanoveného cíle bez zásahu uživatele.

V rámci řešení byl model vozidla sestaven z robotického kitu obsahujícího mikrokontroler Arduino, přičemž řízení bylo realizováno prostřednictvím aplikace vytvořené v jazyce Python. Celý systém využíval horní kamerový pohled a detekci ArUco markerů pro určení polohy vozidla, překážek, kontrolního bodu a cíle. Pro přenos řídicích příkazů byla implementována obousměrná komunikace prostřednictvím protokolu WebSocket. Klíčovým prvkem návrhu byla implementace kolizní zóny, která umožnila včasnou reakci na detekované překážky a následný výpočet objížděné trasy.

Testováním v různých scénářích bylo ověřeno, že navržený systém je funkční, přestože vykazuje určitá omezení vyplývající z kvality obrazu, rozlišení použité kamery a přesnosti detekce markerů. Hlavní cíl práce – vytvoření autonomního vozidla schopného vizuální orientace a samostatného pohybu v dynamickém prostředí – byl úspěšně splněn. Zároveň bylo prokázáno, že i bez specializovaného vybavení lze navrhnout systém spojující hardwarové a softwarové komponenty do funkčního celku.

Tato práce přispívá k prohloubení znalostí v oblasti autonomního řízení a poskytuje praktické zkušenosti se zpracováním obrazu, řízením v reálném čase a návrhem softwarových i hardwarových komponent. V rámci možného budoucího rozšíření by mohlo být systémové řešení doplněno o pokročilejší plánování trasy, zpřesněnou detekci překážek či zvýšenou odolnost vůči rušivým vlivům prostředí.

Celkově lze výsledky této práce hodnotit jako úspěšné z pohledu naplnění stanovených cílů, získaných poznatků a praktických zkušeností, které mohou být dále využity při dalším studiu autonomních systémů a robotiky.

SEZNAM POUŽITÉ LITERATURY

- [1] Autonomní vozidla. *Akademie městské mobility* [online]. c2025 [cit. 2025-01-07]. Dostupné z: <https://www.akademiamobility.cz/autonomni-vozidla>
- [2] Novela o autonomních vozidlech: krok k širšímu využití autonomních technologií na našich silnicích. *Ministerstvo dopravy ČR* [online]. 2024, 30.10.2024 [cit. 2025-01-07]. Dostupné z: <https://md.gov.cz/Media/Media-a-tiskove-zpravy/Novela-o-autonomnich-vozidlech-krok-k-sirsimu-vy>
- [3] ČERNÝ, David. *Automatizované řízení vozidel a autonomní doprava: technické a humanitní perspektivy*. Praha: Academia, 2022. ISBN 978-80-200-3358-1.
- [4] Úrovně autonomního řízení. *Asistenční systémy* [online]. 2022, 07. 07. 2022 [cit. 2025-01-07]. Dostupné z: <https://www.asistencnisystemy.cz/urovne-autonomniho-rizeni>
- [5] BUREŠ, David. Úrovně autonomních aut. Jaký je mezi nimi rozdíl? A která fáze je opravdu auto bez řidiče? *Auto.cz* [online]. 2018, 27. března 2018 [cit. 2025-01-08]. Dostupné z: <https://www.auto.cz/urovne-autonomnich-aut-jaky-je-mezi-nimi-rozdil-a-ktera-faze-je-opravdu-auto-bez-ridice-120259>
- [6] Pět stupňů k autonomnímu řízení. *Škoda Storyboard* [online]. 2018 [cit. 2025-01-08]. Dostupné z: <https://www.skoda-storyboard.com/cs/skoda-svet-cs/inovace-a-technologie/pet-stupnu-k-autonomnimu-rizeni>
- [7] BARTUSKA, Ladislav a Remigiusz LABUDZKI. Research of basic issues of autonomous mobility. *ScienceDirect*. 2020, (44), 356-360.
- [8] ATANER. Autonomní vozidla: Historie, vývoj a budoucnost. *Médium.cz* [online]. 2024, 31. 12. 2024 [cit. 2025-02-11]. Dostupné z: <https://medium.seznam.cz/clanek/ataner-autonomni-vozidla-historie-vyvoj-a-budoucnost-109181>
- [9] Počátky autonomních vozidel. *Autonomne.cz* [online]. 2020, 21. 10. 2020 [cit. 2025-02-11]. Dostupné z: <https://www.autonomne.cz/clanky/pocatky-autonomnich-vozidel>
- [10] RAVANBAKSH, Mahdyar, Mohamad BAYDOUN, Damian CAMPO, Pablo MARIN, David MARTIN, Lucio MARCENARO a Carlo REGAZZONI. Learning Self-Awareness for Autonomous Vehicles: Exploring Multisensory Incremental Models. *Arxiv.org* [online]. 18 Apr 2020 [cit. 2025-02-13]. Dostupné z: <https://arxiv.org/abs/2004.10049>

- [11] Autonomous Vehicle Perception: Enabling Self-Driving Cars. *Dorleco* [online]. 2024, August 27, 2024 [cit. 2025-02-13]. Dostupné z: <https://dorleco.com/autonomous-vehicle-perception/>
- [12] Practical Uses of Autonomous Vehicles Across Industries. *Innoviz technologies* [online]. 2021, September 2, 2021 [cit. 2025-02-13]. Dostupné z: <https://innoviz.tech/blog/autonomous-vehicle-applications>
- [13] Senzory a moderní technologie jako naši osobní řidiči. *IoTPort* [online]. 2024, 18. 09. 2024 [cit. 2025-02-16]. Dostupné z: <https://www.iotport.cz/iot-novinky/logistika-a-doprava/senzory-a-moderni-technologie-jako-nasi-osobni-ridici>
- [14] Auta startupu Navya: Šance Evropy na vlastní autonomní dopravu, nebo zatím jen velký sen? In: *Elektrina.cz* [online]. 2018, 16. dubna 2018 [cit. 2025-05-02]. Dostupné z: <https://www.elektrina.cz/navya-startup-autonomni-vozidla-autonom-shuttle-autonom-cab>
- [15] Česká logistika 2025: Připravená na budoucnost s digitalizací, automatizací a udržitelností. *Svět Průmyslu* [online]. 2025, 30. 1. 2025 [cit. 2025-02-16]. Dostupné z: <https://svetprumyslu.cz/ceska-logistika-2025-pripravena-na-budoucnost-s-digitalizaci-automatizaci-a-udrizitelnosti/>
- [16] TROBLOVÁ, Petra. Pět klíčových trendů pro rok 2025 v logistice a skladování. *Spolek Kompetentních Logistiků a Dodavatelů* [online]. 2025, 10. 1. 2025 [cit. 2025-02-16]. Dostupné z: <https://www.sklad.cz/pet-klicovych-trendu-pro-rok-2025-v-logistice-a-skladovani/>
- [17] KRUPÍČKA, Jiří. Autonomní manipulační vozík Agilox. In: *TRANSPORT-LOGISTIKA.CZ* [online]. 2023, 17.08.2023 [cit. 2025-05-02]. Dostupné z: <https://transport-logistika.cz/logistika/autonomni-manipulacni-vozik-agilox/>
- [18] LETKO, Vojtěch. Nechte se svézt autonomními vozíky do světa Průmyslu 4.0. *Aimtec* [online]. 2020, 21. 2. 2020 [cit. 2025-02-16]. Dostupné z: <https://insights.aimtecglob.com/nechte-se-svezet-autonomnimi-voziky-do-sveta-prumyslu-4-0/>
- [19] Jak se orientovat v budoucnosti: Transformační dopad autonomních vozidel napříč odvětvími. *Atlas Copco* [online]. c2025 [cit. 2025-02-16]. Dostupné z: <https://www.atlascopco.com/cs-cz/itba/expert-hub/articles/the-transformative-impact-of-autonomous-vehicles-across-industries>

- [20] John Deere vyvíjí autonomní a plně elektrický traktor. Kabinu k němu bude možné připojit. In: *Agroportal24h.cz* [online]. 2022, 16. 3. 2022 [cit. 2025-05-02]. Dostupné z: <https://www.agroportal24h.cz/clanky/john-deere-vyviji-autonomni-a-plne-elektricky-traktor-kabinu-k-nemu-bude-mozne-pripojit>
- [21] LUKÁŠ, Vít. Vyvíjené české autonomní vozidlo TAROS pod záštitou ministerstva má obrovskou budoucnost a již ho testovali vojáci AČR. *ArmádníZpravodaj.cz* [online]. 2024, 26. července 2024 [cit. 2025-02-16]. Dostupné z: <https://armadnizpravodaj.cz/pozemni-technika/ceske-autonomni-vozidlo-taros>
- [22] GROHMANN, Jan. Český vojenský robot TAROS 6×6 V2. In: *Armádní noviny.cz* [online]. 2014, 13. 11. 2014 [cit. 2025-05-02]. Dostupné z: <https://www.armadninoviny.cz/cesky-vojensky-robot-taros-6c3976-v2.html>
- [23] Autonomní mobilita: technologie dneška a trend budoucnosti. *Autonomne.cz* [online]. 2021, 10. 8. 2021 [cit. 2025-02-16]. Dostupné z: <https://www.autonomne.cz/aktivita-detail/autonomni-mobilita-technologie-dneska-a-trend-budoucnosti>
- [24] Temná strana autonomních vozů: Opravdu jsou bezpečnější? *Mix24.cz* [online]. 2025, 11. 2. 2025 [cit. 2025-02-16]. Dostupné z: <https://www.mix24.cz/temna-strana-autonomnich-vozu-opravdu-jsou-bezpecnejsi/>
- [25] ČERNÝ, David, Ondřej VACULÍN a Petr ZÁMEČNÍK, ed. *Automatizované řízení vozidel a autonomní doprava : technické a humanitní perspektivy*. Praha : Academia, 2022. ISBN 978-80-200-3358-1.
- [26] MARTINEK, Radislav. *Senzory v průmyslové praxi*. Praha : BEN - technická literatura, 2004. ISBN 8073001144 9788073001148.
- [27] VOJÁČEK, Antonín. Jen obraz nestačí aneb i snímání zvuku pro autonomní stroje. *Automatizace.hw.cz* [online]. 2021, 28. Zář 2021 [cit. 2025-03-30]. Dostupné z: <https://automatizace.hw.cz/jen-obraz-nestaci-aneb-i-snimani-zvuku-pro-autonomni-stroje.html>
- [28] KUCHAR, Martin. Megatest webkamer pro USB - část první. *Svět hardware* [online]. 2005, 21.12.2005 [cit. 2025-03-30]. Dostupné z: <https://www.svethardware.cz/recenze-megatest-webkamer-pro-usb-cast-prvni/13178>
- [29] Co je to webkamera. *Speechify* [online]. c2025 [cit. 2025-03-30]. Dostupné z: <https://speechify.com/cs/blog/webcam/>

- [30] Eternico Webcam ET201 Full HD, černá. In: *Alza.cz* [online]. c1994-2025 [cit. 2025-04-03]. Dostupné z: <https://www.alza.cz/eternico-webcam-et201-full-hd-cerna-d6900130.htm>
- [31] IP kamery. *SecuriaPro.cz* [online]. c2025 [cit. 2025-03-30]. Dostupné z: <https://www.securiapro.cz/digitalni-ip-kamery/>
- [32] 8.0 MP IP Kamera DS-2CD2186G2-I(2.8mm). In: *KameryDomu.cz* [online]. c2025 [cit. 2025-05-02]. Dostupné z: <https://www.kamerydomu.cz/8-0-mp-ip-kamera-ds-2cd2186g2-i-2-8mm-/>
- [33] Analogové kamery. *Spyobchod.cz* [online]. c2010-2025 [cit. 2025-03-30]. Dostupné z: <https://www.spyobchod.cz/bezpecnostni-kamery/analogove-kamery/s16060.htm>
- [34] Securia Pro AHD bezpečnostní kamera 2MP 2.8mm bullet A640XF-200W-W. In: *SecuriaPro.cz* [online]. c2025 [cit. 2025-05-02]. Dostupné z: <https://www.securiapro.cz/securia-pro-ahd-kamera-2mp-2-8mm-bullet-a640xf-200w-w/>
- [35] HUGHES, Austin a Bill DRURY. *Electric motors and drives: fundamentals, types and applications*. Fifth Edition. Oxford: Newnes, 2019. ISBN 978-0-08-102615-1.
- [36] DC motory - stejnosměrný proud. *Botland* [online]. c2025 [cit. 2025-03-28]. Dostupné z: <https://botland.cz/724-dc-motory-stejnosmerny-proud>
- [37] KALOUSOVÁ, Marcela, B., Andrea, ed. Co je indukční motor? *DUDLU* [online]. 2022, 3.11.2022 [cit. 2025-03-28]. Dostupné z: <https://www.dudlu.cz/rady-pro-kutily/co-je-indukcni-motor/>
- [38] What is 3 Phase Induction Motor? Diagram, Working & Types. In: *ElectricalWorkbook* [online]. 28 Jun 2021 [cit. 2025-05-02]. Dostupné z: <https://electricalworkbook.com/three-phase-induction-motor/>
- [39] MRÓZ, Mateusz. Bldc motor – Co to je a jak funguje? *Botland* [online]. 16 března, 2023 [cit. 2025-03-28]. Dostupné z: <https://botland.cz/blog/blde-motor-co-to-je-a-jak-funguje/>
- [40] Brushless DC Motor vs. Hub Motor. In: *Brushless.com* [online]. 2024, October 4, 2024 [cit. 2025-05-02]. Dostupné z: <https://www.brushless.com/brushless-dc-motor-vs-hub-motor>

- [41] Jak funguje servomotor a zpětnovazební řízení. *RAVEO* [online]. c2025 [cit. 2025-03-28]. Dostupné z: <https://www.raveo.cz/aktualita/jak-funguje-servomotor-a-zpetnovazebni-rizeni/>
- [42] Servo Motor – Types, Construction, Working, and Applications. In: *Electrical Technology* [online]. c2024 [cit. 2025-05-02]. Dostupné z: <https://www.electricaltechnology.org/2019/05/servo-motor-types-construction-working.html>
- [43] BLUM, Jeremy. *Exploring Arduino: tools and techniques for engineering wizardry* [online]. Indianapolis: Wiley, c2013 [cit. 2025-04-04]. ISBN 978-1-118-54936-0. Dostupné z: https://mirzaproject.ir/_files/En/Wiley.Exploring.Arduino.Jul.2013.pdf
- [44] VODA, Zbyšek. Seznámení s Arduinem. *Bastlirna* [online]. 2014, 11.10.2014 [cit. 2025-03-23]. Dostupné z: <https://bastlirna.hwkitchen.cz/seznameni-s-arduinem/>
- [45] Arduino Pro Mini 328 - 5V / 16MHz - SparkFun DEV-11113. In: *Botland* [online]. c2025 [cit. 2025-05-02]. Dostupné z: <https://botland.cz/arduino-hlavni-moduly/1595-arduino-pro-mini-328-5v-16mhz-sparkfun-dev-11113-5903351240673.html>
- [46] Modul WeMos D1 mini ESP8266 Wifi. *AtosElektro* [online]. c2025 [cit. 2025-03-23]. Dostupné z: https://shop.atoselektro.cz/modul-wemos-d1-mini-esp8266-wifi_d575278.html
- [47] , Martin. Začínáme s Wemos D1 mini s WiFi modulem ESP8266. In: *Lackakit.cz* [online]. 2020, 23. dubna 2020 [cit. 2025-05-02]. Dostupné z: <https://blog.lackakit.cz/zaciname-s-wemos-d1-mini-s-wifi-modulem-esp8266/>
- [48] Micro. *Arduino Documentation* [online]. c2025 [cit. 2025-03-23]. Dostupné z: <https://docs.arduino.cc/hardware/micro/>
- [49] Arduino MICRO s konektory originál. In: *HWKITCHEN* [online]. c2025 [cit. 2025-05-02]. Dostupné z: <https://www.hwkitchen.cz/arduino-micro-a000053-original/>
- [50] Arduino Uno Rev3. *Arduino Official Store* [online]. c2025 [cit. 2025-03-28]. Dostupné z: <https://store.arduino.cc/en-cz/products/arduino-uno-rev3>
- [51] Programovatelná deska Arduino UNO A000073. In: *Conrad Electronic* [online]. b. r. [cit. 2025-05-02]. Dostupné z: <https://www.conrad.cz/cs/p/programovatelna-deska-arduino-uno-a000073-191789.html>

- [52] Arduino MEGA 2560 R3 kompatibilní vývojová deska (5V 16Mhz). *Pájeničko* [online]. c2024 [cit. 2025-03-28]. Dostupné z: <https://pajenicko.cz/arduino-mega-2560-r3-kompatibilni-vyvojova-deska-5v-16mhz>
- [53] LaskaKit Mega2560 rev3. In: *Laskakit.cz* [online]. c2025 [cit. 2025-05-02]. Dostupné z: <https://www.laskakit.cz/arduino-mega2560/>
- [54] JEŽKOVÁ, Kamila. Seznamte se s Arduino vývojovým prostředím. *HWkitchen.cz* [online]. 2014, 15.8.2014 [cit. 2025-04-04]. Dostupné z: <https://bastlirna.hwkitchen.cz/seznamte-s-arduino-vyvojovym-prostredim/#Sketch>
- [55] Jak správně naimportovat Arduino knihovnu? *HWKITCHEN* [online]. c2025 [cit. 2025-04-04]. Dostupné z: <https://www.hwkitchen.cz/navody-hwkitchen/jak-spravne-naimportovat-arduino-knihovnu-arduino-navody/>
- [56] MALÝ, Martin. Arduino: jak pro něj začít programovat. *Root.cz* [online]. 2010, 8. 7. 2010 [cit. 2025-04-04]. Dostupné z: <https://www.root.cz/clanky/arduino-jak-pro-nej-zacit-programovat/>
- [57] ESPRESSIF SYSTEMS. *ESP8266EX datasheet* [online]. Espressif Systems, 2020 [cit. 2025-04-04]. Dostupné z: https://www.laskakit.cz/user/related_files/esp8266ex_datasheet_en.pdf
- [58] Arduino na ĚspĚčkových steroidech. *HWKITCHEN* [online]. c2025 [cit. 2025-04-04]. Dostupné z: <https://www.hwkitchen.cz/wemos-d1-uno-wifi-esp8266-ota-ch340g/>
- [59] M., Luboš. ESP WiFi Webserver. *Navody.drateg.cz* [online]. b. r. [cit. 2025-04-04]. Dostupné z: <https://navody.drateg.cz/navody-k-produktum/esp-wifi-webserver.html>
- [60] HUDECZEK, Adam. Lekce 4 - Webový server na modulu ESP-32. *Itnetwork.cz* [online]. c2025 [cit. 2025-04-04]. Dostupné z: <https://www.itnetwork.cz/hardware-pc/arduino/esp32/webovy-server-na-modulu-esp-32>
- [61] Websockets. *Websockets* [online]. c2013-2025 [cit. 2025-04-04]. Dostupné z: <https://websockets.readthedocs.io/en/stable/>
- [62] Modul ovladače L298N pro stejnosměrné DC a krokové motory - Arduino. In: *Pajtech.cz* [online]. b. r. [cit. 2025-05-02]. Dostupné z: <https://pajtech.cz/modul-ovladae-l298n-pro-stejnosmrn-dc-a-krokov-motory-arduino-p-8695.html>
- [63] ON/OFF Switch KCD1 2 Pins Dim:17x11mm. In: *MAKERS Co* [online]. c2025 [cit. 2025-05-02]. Dostupné z: <https://makerselectronics.com/product/on-off-switch-kcd1-2-pins-dim17x11mm>

- [64] COMF BH-351A Držák baterie 5xAA drátové vývody. In: *Laskakit.cz* [online]. c2025 [cit. 2025-05-02]. Dostupné z: <https://www.laskakit.cz/comf-bh-351a-drzak-baterie-5xaa-dratove-vyvody/>

SEZNAM OBRÁZKŮ

Obrázek 1: Autonomní autobus pro 15 osob [14].....	16
Obrázek 2: Autonomní manipulační vozík [17]	17
Obrázek 3: Autonomní a plně elektrický traktor [20].....	18
Obrázek 4: TAROS 6x6 [22]	19
Obrázek 5: Webkamera [30].....	23
Obrázek 6: IP digitální kamera [32].....	23
Obrázek 7: Analogová kamera [34].....	24
Obrázek 8: Komutátorový motor [35]	25
Obrázek 9: Třífázový indukční motor [38]	26
Obrázek 10: BLDC motor [40]	27
Obrázek 11: Servomotor [42]	27
Obrázek 12: Arduino Mini [45]	29
Obrázek 13: Arduino Wemos D1 Mini [47]	30
Obrázek 14: Arduino Micro [49]	30
Obrázek 15: Arduino Uno [51]	31
Obrázek 16: Arduino Mega2560 [53]	32
Obrázek 17: Arduino IDE	32
Obrázek 18: Skecth v Arduino IDE	33
Obrázek 19: Výběr knihoven	34
Obrázek 20: Izometrický pohled 3D modelu vozidla	37
Obrázek 21: Spodní pohled 3D modelu vozidla	38
Obrázek 22: Horní pohled 3D modelu vozidla bez vrchní konstrukce	38
Obrázek 23: Boční pohled 3D modelu vozidla	38
Obrázek 24: 3D model základní konstrukční desky	39
Obrázek 25: 3D model kola	40
Obrázek 26: Boční a čelní pohled 3D modelu DC motoru	40
Obrázek 27: H-můstek [62].....	41
Obrázek 28: Vypínač [63].....	41
Obrázek 29: Držák na baterie [64]	42
Obrázek 30: Arduino Wemos D1 Mini [47]	42
Obrázek 31: Horní pohled – rozložení komponent	43
Obrázek 32: Schéma zapojení komponent.....	45
Obrázek 33: Inicializace knihoven a nastavení pinů.....	46
Obrázek 34: Inicializace pinů, připojení k WiFi síti a spuštění serverů	47

Obrázek 35: Zpracování příkazů přijatých přes WebSocker komunikaci	48
Obrázek 36: Funkce pro pohyb vozidla vpřed.....	48
Obrázek 37: Funkce pro pohyb vozidla vzad	49
Obrázek 38: Funkce pro pohyb vozidla vlevo	49
Obrázek 39: Funkce pro pohyb vozidla vpravo.....	49
Obrázek 40: Funkce pro zastavení vozidla	49
Obrázek 41: Generování HTML stránky pro ovládání vozidla	50
Obrázek 42: Inicializace knihoven, komunikace a základních parametrů.....	52
Obrázek 43: Pomocné funkce pro výpočet pozice, úhlů a kolizní zóny	53
Obrázek 44: Detekce markerů a určení pozice vozidla	54
Obrázek 45: Detekce překážek a výpočet objízdného bodu	55
Obrázek 46: Výpočet směru a zobrazení informací v obraze	56
Obrázek 47: Řízení jízdy, měření času a ukončení programu	57
Obrázek 48: Horní pohled – vizualizace projektu	59
Obrázek 49: Zobrazení ujeté trasy	60
Obrázek 50: Snímání ArUco markeru	61
Obrázek 51: Horní pohled na překážky	61
Obrázek 52: Scénář 1 – Překážka mezi checkpointem a cílem	65
Obrázek 53: Scénář 2 – Překážky před a po checkpointu.....	65
Obrázek 54: Scénář 3 – Dvojice překážek v cestě.....	66
Obrázek 55: Scénář 4 – Kritická výchozí situace s více překážkami	67

SEZNAM TABULEK

Tabulka 1: Časový záznam úseku	67
--------------------------------------	----

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

BLDC	Brushless Direct Current
DC	Direct Current
HTML	Hypertext Markup Language
ID	Identifikátor
IDE	Integrated Development Environment
IP	Internet Protocol
LED	Light Emitting Diode
např.	například
NVR	Network Video Recorder
PWM	Pulse Width Modulation
RFID	Radio Frequency Identification
tzv.	takzvaně
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
WiFi	Wireless Fidelity

SEZNAM PŘÍLOH

Příloha P I: Arduino kód

Příloha P II: Python kód

Příloha P III: Záznam scénářů

PŘÍLOHA P I: ARDUINO KÓD

```
#include <ESP8266WiFi.h>

#include <ESP8266WebServer.h>

#include <WebSocketsServer.h>


const char* ssid = "*****";

const char* password = "*****";


#define ENB D3

#define MOTORB_1 D4

#define MOTORB_2 D5

#define MOTORA_1 D7

#define MOTORA_2 D6

#define ENA D8


WebSocketsServer webSocket = WebSocketsServer(81); // WebSocket server na
portu 81

ESP8266WebServer server(80); // WebServer na portu 80


void setup() {

    pinMode(ENA, OUTPUT);

    pinMode(ENB, OUTPUT);

    pinMode(MOTORA_1, OUTPUT);

    pinMode(MOTORA_2, OUTPUT);

    pinMode(MOTORB_1, OUTPUT);

    pinMode(MOTORB_2, OUTPUT);

    digitalWrite(ENA, LOW);

    digitalWrite(ENB, LOW);


    Serial.begin(115200);
```

```

WiFi.begin(ssid, password);

Serial.print("Connecting to WiFi...");
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("Connected!");
Serial.print("IP Address: ");
Serial.println(WiFi.localIP());

server.on("/", HTTP_GET, [](){
    server.send(200, "text/html", getHTML());
});

WebSocket.begin(); // Start WebSocket server
WebSocket.onEvent(WebSocketEvent); // Při událostech WebSocketu
server.begin();
}

void loop() {
    server.handleClient(); // Zpracování požadavků na HTTP
    WebSocket.loop(); // Zpracování WebSocket událostí
}

void WebSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t
length) {
    switch(type) {
        case WStype_DISCONNECTED:
            Serial.println("Disconnected from WebSocket");
            break;
        case WStype_CONNECTED:
            Serial.println("Connected to WebSocket");

```

```

        break;

    case WStype_TEXT:

        String message = String((char*)payload);

        if (message == "forward") {

            moveForward();

        } else if (message == "backward") {

            moveBackward();

        } else if (message == "left") {

            turnLeft();

        } else if (message == "right") {

            turnRight();

        } else if (message == "stop") {

            stopMotors();

        }

        break;

    }

}

void moveForward() {

    digitalWrite(ENA, HIGH);

    digitalWrite(ENB, HIGH);

    digitalWrite(MOTORA_1, HIGH);

    digitalWrite(MOTORA_2, LOW);

    digitalWrite(MOTORB_1, LOW);

    digitalWrite(MOTORB_2, HIGH);

    Serial.println("Moving forward");

}

void moveBackward() {

    digitalWrite(ENA, HIGH);

    digitalWrite(ENB, HIGH);

```

```
digitalWrite(MOTORA_1, LOW);  
digitalWrite(MOTORA_2, HIGH);  
digitalWrite(MOTORB_1, HIGH);  
digitalWrite(MOTORB_2, LOW);  
Serial.println("Moving backward");  
}  
  
void turnLeft() {  
    digitalWrite(ENA, HIGH);  
    digitalWrite(ENB, HIGH);  
    digitalWrite(MOTORA_1, HIGH);  
    digitalWrite(MOTORA_2, LOW);  
    digitalWrite(MOTORB_1, HIGH);  
    digitalWrite(MOTORB_2, LOW);  
    Serial.println("Turning left");  
}  
  
void turnRight() {  
    digitalWrite(ENA, HIGH);  
    digitalWrite(ENB, HIGH);  
    digitalWrite(MOTORA_1, LOW);  
    digitalWrite(MOTORA_2, HIGH);  
    digitalWrite(MOTORB_1, LOW);  
    digitalWrite(MOTORB_2, HIGH);  
    Serial.println("Turning right");  
}  
  
void stopMotors() {  
    digitalWrite(ENA, LOW);  
    digitalWrite(ENB, LOW);  
    digitalWrite(MOTORA_1, LOW);  
    digitalWrite(MOTORA_2, LOW);
```

```

    digitalWrite(MOTORB_1, LOW);

    digitalWrite(MOTORB_2, LOW);

    Serial.println("Stopped");
}

String getHTML() {
    return "<!DOCTYPE html>"

        "<html><head><title>Ovladani auta</title>"

        "<script>"

        "var ws = new WebSocket('ws://192.168.1.x:81');" // konkrétní IP
adresa Wemos zařízení

        "ws.onopen = function() {"

        "  console.log('WebSocket connection established');"

        "};"

        "ws.onerror = function(error) {"

        "  console.log('WebSocket Error: ' + error);"

        "};"

        "document.addEventListener('keydown', function(event) {"

        "  if(event.key === 'ArrowUp') ws.send('forward');"

        "  if(event.key === 'ArrowDown') ws.send('backward');"

        "  if(event.key === 'ArrowLeft') ws.send('left');"

        "  if(event.key === 'ArrowRight') ws.send('right');"

        "});"

        "document.addEventListener('keyup', function(event) {"

        "  ws.send('stop');"

        "});"

        "</script></head><body>"

        "<h1>Ovladani auta</h1>"

        "<p>⬆ - dopředu, ⬇ - dozadu, ⬵ - doprava, ⬵ -
- doleva</p>"

        "</body></html>";

}

```

PŘÍLOHA P II: PYTHON KÓD

```
import cv2

import numpy as np

import websocket

import time

from cv2 import aruco

class WebSocketCar:

    def __init__(self, url):

        self.ws = websocket.WebSocket()

        self.ws.connect(url)

    def send(self, command):

        print(f"[SEND] {command}")

        self.ws.send(command)

car = WebSocketCar("ws://192.168.137.35:81")

aruco_dict_auto = aruco.getPredefinedDictionary(aruco.DICT_5X5_250)
aruco_dict_nav = aruco.getPredefinedDictionary(aruco.DICT_4X4_50)
aruco_dict_obstacles = aruco.getPredefinedDictionary(aruco.DICT_5X5_100)
parameters = aruco.DetectorParameters()

AUTO_ID = 0

CHECKPOINT_ID = 39

GOAL_ID = 14

OBSTACLE_ID = 11

cap = cv2.VideoCapture(1)
```

```
STOP_DISTANCE = 50

ANGLE_TOLERANCE = 15

step = 1

trajectory = []

last_nav_pos = None

nav_pos = None

obstacle_positions = []

detour_active = False

detour_point = None

ZONE_LENGTH = 150

ZONE_WIDTH_FRONT = 100

ZONE_WIDTH_BACK = 40

zone_polygon = None

start_time = None

travel_time = None

def get_marker_center(corners):

    x = int(np.mean(corners[:, 0]))

    y = int(np.mean(corners[:, 1]))

    return (x, y)

def get_marker_angle(corners):

    top_left, top_right = corners[0], corners[1]

    dx = top_right[0] - top_left[0]

    dy = top_right[1] - top_left[1]
```

```

angle = np.arctan2(dy, dx) * 180 / np.pi

return angle % 360

def angle_difference(a, b):
    diff = (b - a + 360) % 360
    if diff > 180:
        diff -= 360
    return diff

def is_in_collision_zone(auto_pos, auto_angle, obstacle_pos):
    global zone_polygon

    angle_rad = np.deg2rad(auto_angle)
    dir_vec = np.array([np.cos(angle_rad), np.sin(angle_rad)])
    side_vec = np.array([-dir_vec[1], dir_vec[0]])

    a = np.array(auto_pos)
    b1 = a + dir_vec * ZONE_LENGTH + side_vec * (ZONE_WIDTH_FRONT / 2)
    b2 = a + dir_vec * ZONE_LENGTH - side_vec * (ZONE_WIDTH_FRONT / 2)
    a1 = a + side_vec * (ZONE_WIDTH_BACK / 2)
    a2 = a - side_vec * (ZONE_WIDTH_BACK / 2)

    pts = np.array([a1, a2, b2, b1], dtype=np.int32)
    zone_polygon = pts.copy()

    obstacle_float = tuple(map(float, obstacle_pos))
    is_inside = cv2.pointPolygonTest(pts, obstacle_float, False) >= 0
    print(f"[DEBUG] Kolizni test - obstacle {obstacle_pos} in zone: {is_inside}")
    return is_inside

```

```

while True:

    ret, frame = cap.read()

    if not ret:

        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    corners_auto, ids_auto, _ = aruco.detectMarkers(gray, aruco_dict_auto,
parameters=parameters)

    corners_nav, ids_nav, _ = aruco.detectMarkers(gray, aruco_dict_nav,
parameters=parameters)

    corners_obs, ids_obs, _ = aruco.detectMarkers(gray,
aruco_dict_obstacles, parameters=parameters)

    auto_pos, auto_angle = None, None

    target_id = CHECKPOINT_ID if step == 1 else GOAL_ID

    zone_polygon = None

    obstacle_positions = []

    if ids_auto is not None:

        for i, id in enumerate(ids_auto.flatten()):

            if id == AUTO_ID:

                corners = corners_auto[i][0]

                auto_pos = get_marker_center(corners)

                trajectory.append(auto_pos)

                auto_angle = (get_marker_angle(corners) + 90) % 360

                cv2.circle(frame, auto_pos, 6, (255, 0, 0), -1)

    arrow_len = 60

```

```

        angle_rad = np.deg2rad(auto_angle)
        end_x = int(auto_pos[0] + arrow_len * np.cos(angle_rad))
        end_y = int(auto_pos[1] + arrow_len * np.sin(angle_rad))
        cv2.arrowedLine(frame, auto_pos, (end_x, end_y), (0, 255,
0), 2)

```

```

if ids_nav is not None:

```

```

    for i, id in enumerate(ids_nav.flatten()):
        corners = corners_nav[i][0]
        if id == target_id:
            nav_pos = get_marker_center(corners)
            if not detour_active:
                last_nav_pos = nav_pos

```

```

if ids_obs is not None:

```

```

    for i, id in enumerate(ids_obs.flatten()):
        if id == OBSTACLE_ID:
            corners = corners_obs[i][0]
            pos = get_marker_center(corners)
            obstacle_positions.append(pos)
            cv2.circle(frame, pos, 6, (0, 0, 255), -1)
            cv2.putText(frame, "PREKAZKA", (pos[0]+10, pos[1]),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

```

```

    if not detour_active and auto_pos and auto_angle and
obstacle_positions:

```

```

        for pos in obstacle_positions:
            if is_in_collision_zone(auto_pos, auto_angle, pos):
                offset = 150
                angle_rad = np.deg2rad(auto_angle)
                dir_vec = np.array([np.cos(angle_rad), np.sin(angle_rad)])

```

```

        side_vec = np.array([-dir_vec[1], dir_vec[0]])

        rel_vec = np.array(pos) - np.array(auto_pos)

        side_sign = np.sign(np.cross(dir_vec, rel_vec))

        if side_sign == 0:
            side_sign = 1

        detour_point = tuple((np.array(pos) + side_vec * offset *
side_sign).astype(int))

        last_nav_pos = detour_point

        detour_active = True

        print(f"[INFO] PREKAZKA V ZONE - OBJIZDIM PREKOZKU NA BOD:
{detour_point}")

        break

    if zone_polygon is not None:

        cv2.polylines(frame, [zone_polygon], True, (0, 255, 255), 2)

    if detour_active and auto_pos and detour_point:

        if np.linalg.norm(np.array(auto_pos) - np.array(detour_point)) <
STOP_DISTANCE:

            print("[INFO] Objizdka hotova - pokračuju na cil")

            last_nav_pos = nav_pos

            detour_active = False

    if last_nav_pos is not None and auto_pos and auto_angle is not None:

        dx = last_nav_pos[0] - auto_pos[0]

        dy = last_nav_pos[1] - auto_pos[1]

        target_angle = (np.arctan2(dy, dx) * 180 / np.pi) % 360

        angle_diff = angle_difference(auto_angle, target_angle)

        distance = np.linalg.norm(np.array(last_nav_pos) -
np.array(auto_pos))

```

```

cv2.arrowedLine(frame, auto_pos, last_nav_pos, (255, 0, 0), 2)

if len(trajjectory) > 1:
    cv2.polylines(frame, [np.array(trajjectory, dtype=np.int32)],
False, (0, 0, 255), 2)

    cv2.putText(frame, f"Step: {step}", (10, 15),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 255, 100), 2)

    cv2.putText(frame, f"Car angle: {auto_angle:.2f}", (10, 40),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (200, 200, 0), 2)

    cv2.putText(frame, f"Target angle: {target_angle:.2f}", (10, 70),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (200, 200, 0), 2)

    cv2.putText(frame, f"Angle diff: {angle_diff:.2f}", (10, 100),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (200, 200, 0), 2)

    cv2.putText(frame, f"Distance: {distance:.2f}", (10, 130),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (200, 200, 0), 2)

if auto_pos:
    cv2.putText(frame, f"AUTO: {auto_pos}", (10, 160),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

if nav_pos:
    nav_label = "CHECKPOINT" if step == 1 else "CIL"
    cv2.putText(frame, f"{nav_label}: {nav_pos}", (10, 190),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

if obstacle_positions:
    for i, pos in enumerate(obstacle_positions):
        cv2.putText(frame, f"PREKAZKA: {pos}", (10, 220 + i*25),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

#Zobrazení času
if travel_time is not None:
    cv2.putText(frame, f"Cas: {travel_time:.2f}s", (10, 280),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 255, 255), 2)

elif start_time is not None:

```

```

        elapsed_time = time.time() - start_time

        cv2.putText(frame, f"Cas: {elapsed_time:.2f}s", (10, 280),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 255, 255), 2)

        if abs(angle_diff) > ANGLE_TOLERANCE:

            car.send("right" if angle_diff > 0 else "left")

        else:

            if distance < STOP_DISTANCE:

                car.send("stop")

                if step == 1:

                    print("[INFO] Dosahl checkpointu - pokračuji na cil")

                    step = 2

                    start_time = time.time()

                elif step == 2 and not detour_active:

                    if travel_time is None and start_time is not None:

                        travel_time = time.time() - start_time

                        print(f"[INFO] CIL DOSAŽEN! Cas od checkpointu:
{travel_time:.2f} s")

                        print("[INFO] Auto zastaveno, pripraveno na dalsi
akci.")

                        start_time = None

                    else:

                        car.send("forward")

            else:

                car.send("stop")

                cv2.putText(frame, "Hledam auto nebo cil...", (10, 250),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

        cv2.imshow("Auto View", frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):

            break

    cap.release()

    cv2.destroyAllWindows()

```

PŘÍLOHA P III: ZÁZNAM SCÉNÁŘŮ

