Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Konstrukce a řízení kolového robotického prostředku

Bakalářská práce

Jakub Švarc

Vedoucí práce: Mgr. Jakub Geyer

Konzultant: PhDr. Milan Novák, Ph.D.

České Budějovice 2024

Bibliografické údaje

Švarc J., 2024: Konstrukce a řízení kolového robotického prostředku. [Construction and control of a wheeled robotic vehicle. Bc. Thesis, in Czech.] – 95 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Práce se zabývá návrhem a konstrukcí kolového robotického prostředku a jeho řízením. Úvodem podává přehled problematiky kolových robotických prostředků, jejich komponent a metod řízení. Dále popisuje návrh a realizaci jednotlivých fyzických součástí konstrukce, senzoriky a řídicí elektroniky. Věnuje se též popisu návrhu a implementace řídicího a obslužného softwaru a použití algoritmů pro autonomní řízení.

Abstract

The thesis deals with the design and construction of a wheeled robotic vehicle and its control. It gives an overview of wheeled robotic vehicles, their components and control methods. It then describes the design and implementation of the individual physical components of the system, sensors and control electronics. It also describes the design and implementation of control and operator software and the use of algorithms for autonomous control.

Prohlašuji, že jsem autorem této kvalifikační práce a že jsem ji vypracoval pouze s použitím pramenů a literatury uvedených v seznamu použitých zdrojů.

V Českých Budějovicích dne 12.4.2024

Jakub Švarc

Poděkování

Rád bych poděkoval panu Mgr. Jakubu Geyerovi a panu PhDr. Milanu Novákovi, Ph.D. za odborné vedení, cenné rady a pomoc při tvorbě práce. Dále děkuji za možnost využití zázemí a vybavení laboratoře 3D tisku a aditivní výroby, jež vyhotovení práce dosti usnadnilo. Rovněž bych chtěl poděkovat mé rodině, přítelkyni a přátelům za podporu a trpělivost.

Obsah

Seznam použitých zkratek **1**

1 Úvod **5**

1.1 Cíle práce 5

1.2 Metodika práce 6

1.3 Funkční požadavky 7

1.4 Nefunkční požadavky 7

1.4.1 Nefunkční požadavky konstrukce a senzoriky 7

1.4.2 Nefunkční požadavky softwaru a řídicího subsystému 8

2 Teoretická část **9**

2.1 Kolové robotické prostředky 9

2.1.1 Dělení kolových robotů 9

2.1.2 Řídicí jednotky a software 11

2.1.3 Komunikace 19

2.1.4 Senzorika mobilních robotů 22

2.2 Řízení KRP 24

2.2.1 Získávání informací o prostředí a KRP 25

2.2.2 Lokalizace prostředku 25

2.2.3 Tvorba mapy prostředí 26

2.2.4 Plánování cesty 27

2.2.5 Řízení pohybu 29

2.3 Analýza dostupných řešení 30

2.3.1 Průmyslové řešení 30

2.3.2 Akademické řešení 31

2.3.3 Hobby řešení 32

2.3.4 Shrnutí analýzy 33

3 Praktická část **36**

3.1 Návrh a realizace konstrukce 36

3.1.1 Výběr materiálů 36

3.1.2 3D model 37

3.1.3 Pohonná soustava 39

3.1.4 Elektronika 42

3.1.5 Postup konstrukce 42

3.2 Senzorický subsystém 45
3.2.1 Čelní LiDAR 45
3.2.2 Soustavy sonarů 47
3.2.3 Kontaktní senzory 49
3.2.4 Hallovy sondy 50
3.2.5 Kamera pro sledování linky 50
3.2.6 GPS modul a kompas 50
3.3 Řídicí subsystém 51
3.3.1 Původní řešení 51
3.3.2 Výsledné řešení 58
3.3.3 Řídicí aplikace 59
3.4 Klientská aplikace 64
3.4.1 Obrazovka řízení 65
3.4.2 Obrazovka monitoringu 67
3.4.3 Obrazovka nastavení 69
3.4.4 Obrazovka informací o aplikaci 69
3.5 Řízení prostředku 70
3.5.1 Sledování linky a čtení značek 70
3.5.2 Autonomní jízda podle souřadnic 71
3.5.3 Manuální řízení 75
3.5.4 Detekce kolizí a prevence pádu 76
3.6 Testování 76

4 Závěr **78**
4.1 Budoucí rozvoj 78

Seznam použité literatury **79**

Seznam obrázků **92**

Seznam tabulek **93**

Seznam fragmentů kódu **94**

Seznam příloh **95**

Seznam použitých zkratek

A/D (Analog-to-Digital) např. A/D převodník. 12, 14, 20, 51

APL (Arduino Programming Language) standardní API pro programování na platformě Arduino. 13

BCM (Broadcom) standardní číslování pinů na počítačích Raspberry Pi. 59

BKO bistabilní klopný obvod. 53

CAD (Computer Aided Design) technologie návrhu 2D a 3D objektů pomocí softwaru. 78

CAN (Controller Area Network) je sériové asynchronní rozhraní. 15, 21

CSI (Camera Serial Interface) specifikace definující univerzální sériové rozhraní pro připojení kamer. 17, 50

D/A (Digital-to-Analog) např. D/A převodník. 12, 14, 20

DC (Direct Current) stejnosměrný proud. 31, 32

DIO (Digital Input/Output) digitální vstupní či výstupní port. 18, 51

DIP (Dual In-line Package) typ pouzdra pro integrované obvody s obdélníkovým tvarem a vývody podél obou delších stran. 14

DPS deska plošných spojů. 17, 19, 20, 48, 49, 57, 76, 78, 91

DSI (Display Serial Interface) specifikace definující univerzální sériové rozhraní pro připojení displejů. 17

EEPROM (Electrically Erasable Programmable Read-only Memory) elektricky přepisovatelná non-volatilní paměť. 12, 13, 20, 51

ESP-IDF (ESP IoT Development Framework) open-source framework pro vývoj aplikací na deskách ESP. 15

FCL (Fuzzy Control Language) jazyk pro definici fuzzy systémů. 73

GND (Ground) referenční napěťová úroveň v obvodu (efektivně 0 V). 42, 46, 59

GNSS (Global Navigation Satellite System) obecné označení pro satelitní navigační systém s globálním pokrytím. 32, 34

GPIO (General-purpose Input/Output) vstupně-výstupní rozhraní, jehož účel definuje až programátor aplikace. 12, 14, 17, 18, 42, 53, 58, 59

GPS (Global Positioning System) americký globální družicový polohový systém. 11, 23

HAL (Hardware Abstraction Layer) softwarové rozhraní, jež umožňuje unifikovaný a zjednodušený přístup k hardwaru. 61

HEVC (High Efficiency Video Coding) standard pro kompresi videa. 17

HMI (Human-Machine Interface) rozhraní, skrze které přistupuje uživatel ke stroji či obecně systému. 18

HSV (Hue, Saturation, Value) alternativní barevný model. 71

I^2C (Inter-Integrated Circuit) viz *TWI*. 20

I^2S (Inter-Integrated Circuit Sound) sériové rozhraní pro připojení audio zařízení. 14, 16

IoT (Internet of Things) koncept vzájemného propojení zařízení, jež běžně pracují samostatně a jejich připojení do standardních komunikačních sítí. 18

KRP kolový robotický prostředek. 4, 9–12, 19–25, 29–32, 34, 35, 93

MAC (Medium Access Control) standard (též vrstva ISO/OSI modelu) jež definuje vlastnosti rozhraní pro vzájemnou síťovou komunikaci. 14

MIB (mals-intermodule-broker). 50, 59, 60, 62–65, 67, 71, 76

MIPI (Mobile Industry Processor Interface Alliance) organizace definující standardy pro mobilní systémy. 17

MISO (Master In, Slave OUT) rozhraní pro přenos dat od podřízeného zařízení k řídicímu. 21

MOSI (Master Out, Slave In) rozhraní pro přenos dat od řídicího zařízení k podřízenému. 21

MQTT (MQ Telemetry Transport) síťový komunikační protokol s nízkou režií, užívaný především v IoT. 60

MRP mobilní robotický prostředek/platforma. 9

OPA (Optical Phased Array) dvourozměrné pole optických antén, umístěných velmi blízko k sobě. 23

PETG (Polyethylene Terephthalate + Glycol) termoplast, velmi často užívaný ve 3D tisku. 36–39, 41, 47, 48, 50

PID proporcionální, integrační, derivační (např. kontrolér). 29

PIR (Passive Infrared) technologie snímání infračerveného světla generovaného objekty ve snímané oblasti. 22

PLA (Polylactic Acid) termoplast vyráběný z přírodních zdrojů, velmi často užívaný ve 3D tisku. 32

PoE (Power over Ethernet) metoda napájení zařízení skrze ethernetové kabely. 18

PWM (Pulse-Width Modulation) diskrétní nekvantovaná modulace, obvykle užívaná k řízení množství přenášené energie. 12, 14, 30, 31, 46, 52, 53, 58, 59, 74, 75, 92

QSPI (Quad Serial Peripheral Interface) rozhraní pro komunikaci mezi procesorem a flash pamětí s využitím SPI. 14

RC (Radio Control) metoda přenosu řídicích příkazů pomocí rádiových vln. 30

REST (Representational State Transfer) architektonický styl pro aplikace typu klient-server. 22

RISC (Reduced Instruction Set Computer) počítačová architektura s krátkými a jednoduchými instrukcemi shodné délky. 12, 14

RTC (Real Time Clock/Counter) obvod zajišťující měření skutečného času. 12, 14

RTK (Real-Time Kinematic) metoda pro korekci chyb GNSS. 32

Rx (Receiver) přijímač. 46

SBC (Single-Board Computer) malý počítač se všemi komponentami na jedné DPS (jednodeskový počítač). 35

SCL (Serial Clock) linka pro hodinový signál. 20, 21, 46, 59

SCU (Subsystems Control Unit). 46, 47, 51, 52, 56, 61, 62

SDA (Serial Data) datová linka. 20, 46, 59

SDRAM (Synchronous Dynamic Random-Access Memory) dynamická RAM se synchronním rozhraním. 17

SLAM (Simultaneous Localisation and Mapping) metoda současné lokalizace a tvorby mapy v nestrukturovaném prostředí. 27

SoC (System on a Chip) integrovaný obvod obsahující důležité části počítače; obvykle CPU, GPU a rozhraní pro operační paměť, sekundární paměť a vstupní a výstupní zařízení. 17, 58

SPI (Serial Peripheral Interface) sériové asynchronní plně duplexní rozhraní. 12, 14, 16, 21, 51

SRAM (Static Random-Access Memory) polovodičová paměť tvořená bistabilními klopnými obvody. 12–14, 51

SRSC (Simple Reliable Serial Communication) komunikační protokol navržený pro komunikaci mezi řídicími jednotkami prostředku. 54, 56, 62

SS (Slave Select) rozhraní pro výběr podřízeného zařízení. 21

THT (Through-hole technology) způsob osazování komponent pomocí pájení drátových vývodů vložených do otvorů v DPS. 46, 48

TWI (Two-Wire Interface) sériové synchronní half-duplexní rozhraní, kompatibilní s I^2C. 12, 14–16, 20, 45–48, 51, 53, 59

Tx (Transmitter) vysílač. 46

UART (Universal Asynchronous Receiver-Transmitter) protokol pro definici asynchronní sériové komunikace. 14–16, 20, 45, 51, 53, 54, 57

USART (Universal Synchronous/Asynchronous Receiver-Transmitter) UART s možností synchronní činnosti. 12

UWB (Ultra-Wideband) bezdrátová technologie nízkoenergetické komunikace na krátkou vzdálenost v širokém pásmu. 32

1. Úvod

Mobilní robotika se v posledních desetiletích stává čím dál důležitějším oborem. Oblast použití mobilních a zvláště kolových robotů je rok od roku širší a zahrnuje celou řadu oborů, od výroby, dopravy, zdravotnictví, záchranných složek, až po zábavní průmysl, vzdělávání a výzkum.

Důležitou ukázkou uplatnění kolových robotů byla pandemie nemoci Covid-19. Při ní vyvstala snaha o minimalizaci mezilidského kontaktu, který musel být nějakým způsobem nahrazen. Obzvláště ve velmi rizikových oblastech jeho roli převzaly právě kolové roboty, jako např. robot pro dezinfekci povrchů popsaný v [1], teleprezenční robot, díky kterému nejsou lékaři přímo vystaveni nebezpečí nákazy [2] či robot monitorující teplotu lidí a dodržování bezpečnostních opatření [3].

Díky těmto faktorům se problematice kolových robotů věnuje stále více pozornosti, a to jak z řad průmyslových společností, tak výzkumníků a čím dál více i hobby nadšenců. To příznivě ovlivňuje rychlost vývoje nových technologií i množství dostupných řešení na trhu. I přes to všechno však stále přetrvává jeden problém, a tím je cena. Průmyslová řešení, obzvláště ta s vyšší nosností, jsou velmi nákladná - jejich cena se obvykle pohybuje v řádu vyšších stovek tisíc korun. Proto vyvstal návrh sestrojit vlastní řešení, jež bude navrženo tak, aby tuto cenu pokud možno co nejvíce snížilo. Nižší cena však není jedinou výhodou vlastního řešení. Komerční produkty málokdy přesně odpovídají konkrétním požadavkům, a i přes to, že se výrobci mnohdy snaží o univerzální řešení, vlastní implementace vyhovuje požadavkům zpravidla lépe.

Tato práce má tedy za cíl přinést univerzální, rozšiřitelné a dostupné řešení s možností autonomního řízení, které bude možno v budoucnu využít k různým aktivitám nejen v rámci činnosti katedry.

1.1 Cíle práce

Hlavním cílem práce je navrhnout a sestrojit kolový robotický prostředek a implementovat jeho řízení. Prostředek musí být univerzálně použitelný, a to jak ve vnitřním, tak vnějším prostředí. Řízení musí umožňovat automatizaci cest pomocí zadávání GPS souřadnic a čtení horizontálního značení na povrchu pod prostředkem. Dále, prostředek musí být schopen během autonomní jízdy detekovat překážky v cestě a předcházet kolizím s nimi. Rovněž musí detekovat a řešit možné pády, např. ze schodů. Konstrukce musí být navržena tak, aby prostředek zvládl uvézt náklad s vyšší hmotností (např. robotické rameno) a aby byla v rámci možností jednoduše replikovatelná. Při výbě-

ru materiálů by měla být zohledněna jejich dostupnost, ideálně v zásobách laboratoře 3D tisku a aditivní výroby, a možnost jejich zpracování pomocí nástrojů, jež jsou v laboratoři k dispozici.

Z hlavního cíle vyplývají následující dílčí úkoly:

1. Rešerše kolových robotických prostředků, jejich komponent a dostupných řešení.
2. Výběr vhodných materiálů pro konstrukci.
3. Výběr komponent pro řízení prostředku a snímání jeho okolí.
4. Návrh konstrukce v CAD softwaru.
5. Návrh a výroba desek plošných spojů.
6. Návrh a implementace firmwaru pro senzory.
7. Návrh a implementace řídicí aplikace spolu s algoritmy pro řízení prostředku.
8. Návrh a implementace klientské aplikace.
9. Fyzická konstrukce prostředku.
10. Zprovoznění softwaru na řídicí jednotce.

Výsledkem práce bude prototyp prostředku, včetně senzoriky, řídicího softwaru a klientské aplikace. Dále pak schémata elektroniky, návrhy desek plošných spojů, zdrojová data 3D modelů a v neposlední řadě pak zdrojové kódy řídicí aplikace, klientské aplikace, firmwaru senzorů a též uživatelská dokumentace.

1.2 Metodika práce

Obecně lze říci, že hlavní metodikou využitou k vývoji softwaru i hardwaru byl *rapid prototyping*. Tedy nejprve byla vybrána malá dílčí část práce (např. komponenta či funkcionalita), ta byla navržena a vytvořena podle dostupných požadavků a následně samostatně otestována. Pokud byly výsledky testů uspokojivé, vývoj se přesunul k další části práce. Pokud tomu bylo naopak, byly provedeny úpravy k odstranění nedostatků a testování se opakovalo. Tento postup se opakoval, dokud daná část nesplňovala všechny požadavky. Nutno zmínit, že na začátku vývoje byla snaha o využití metodiky *scrum*, ta se však záhy ukázala jako příliš těžkopádná pro jednočlenný tým a byla nahrazena právě rapid prototypingem.

1.3 Funkční požadavky

Funkčními požadavky je myšleno požadované chování a funkcionalita řešení. Jedná se především o vlastnosti, které řešení nabízí uživateli k využití.

- Klientská aplikace musí umožnit autonomní řízení prostředku po trase zadané GPS souřadnicemi.
- Klientská aplikace musí umožnit autonomní řízení prostředku podle horizontálního značení.
- Klientská aplikace by měla umožnit manuální ovládání prostředku.
- Klientská aplikace by měla poskytnout uživateli základní přehled o aktuálním stavu prostředku.
- Řídicí aplikace musí umožnit spojení a vzájemnou výměnu dat a příkazů s klientskou aplikací.
- Řídicí aplikace by měla být připravena na komunikaci a kooperaci s převáženým zařízením.

1.4 Nefunkční požadavky

Nefunkčními požadavky jsou myšleny vlastnosti řešení, které přímo nesouvisí s jeho funkčností. Jedná se např. o bezpečnost, spolehlivost, výkon, možnosti rozšíření atd.

1.4.1 Nefunkční požadavky konstrukce a senzoriky

- Konstrukce musí zvládnout převoz nákladu, jako např. robotické ruky.
- Materiály konstrukce by měly být běžně dostupné, primárně ze zásob laboratoře.
- Dodatečné prvky konstrukce by měly být primárně vyráběny pomocí 3D tisku.
- Konstrukce musí umožnit snadné odebrání či přidání dodatečných senzorů.
- Senzorika musí zajistit snímání okolí ve směru pohybu prostředku, tedy před a za prostředkem.

1.4.2 Nefunkční požadavky softwaru a řídicího subsystému

- Řídicí software by měl být modulární, a tedy umožnit relativně jednoduché přidání ucelené funkcionality a její zapojení do celého systému.
- Řídicí aplikace by měla zamezit současnému přístupu více uživatelů k ovládání prostředku.
- Klientská aplikace by měla umožňovat jednoduché a uživatelsky přívětivé ovládání prostředku.
- Pokud to bude možné, jako řídicí jednotka by měl být využit průmyslový počítač POC-200.

2. Teoretická část

Cílem této kapitoly je podat čtenáři základní přehled o problematice kolových robotických prostředků (KRP) jako celku a rovněž o jednotlivých součástech, ze kterých se skládají. Jsou zde popsány principy fungování a typy KRP, druhy řídicích jednotek, způsoby komunikace mezi různými subsystémy KRP a typy senzorů. Druhá část kapitoly se věnuje procesu řízení prostředku a jeho jednotlivým krokům. Ve třetí části jsou analyzovány KRP dostupné na trhu, především ty s vyšší nosností.

2.1 Kolové robotické prostředky

Kolové robotické prostředky jsou typem mobilních robotů[1], který se vyznačuje pohybem po zemi, a to za pomoci kol (na rozdíl od pásových či kráčejících MRP).

2.1.1 Dělení kolových robotů

KRP lze klasifikovat podle celé řady různých kritérií, dále budou popsána ta nejdůležitější.

2.1.1.1 Dělení podle holonomnosti

Dle práce [5] reprezentuje holonomnost vztah mezi celkovým počtem stupňů volnosti KRP a jeho řiditelnou podmnožinou.

Za *holonomní* je KRP považován tehdy, je-li počet jeho řiditelných stupňů volnosti shodný s jeho celkovým počtem stupňů volnosti. To znamená, že pro pohyb v požadovaném směru není nutné vykonávat žádné dodatečné pohyby (např. natočení nápravy v požadovaném směru). [6] Holonomní KRP proto bývají často nazývány *všesměrové*. Příkladem může být KRP *Ethon* popsaný v práci [7], který k dosažení holonomnosti užívá tzv. "Kiwi pohon", skládající se ze tří samostatně řízených všesměrových kol.

Naproti tomu za *neholonomní* se považuje KRP s menším počtem řiditelných stupňů volnosti než je jejich celkový počet [5], což znamená, že pokud není požadovaný pohyb v aktuálním směru natočení KRP, je nutné provést nejprve korekci - natočení do požadovaného směru. [6] Jako příklad lze uvést KRP popsaný v práci [8], který se pohybuje pomocí šesti pevných kol.

[1] Mobilní robotické prostředky (MRP) lze definovat jako automatická zařízení, která jsou schopna pohybovat se v daném prostředí. [4]

2.1.1.2 Dělení podle typu kol

KRP lze z pohledu typu kol dělit do dvou hlavních kategorií, které korespondují s dělením podle holonomnosti:

První je *KRP se směrovými (klasickými) koly*, kam spadá většina kolových robotů. Jedná se zpravidla o neholonomní KRP. [9] Kola zde využívaná mohou být:

- *pevná* - klasická kola s pevnou vazbou na osu [9]
- *orientovatelná centrická* - kola s pohyblivým rámem (otočný kolem svislé osy), kde osa otáčení kola protíná osu otáčení rámu [9]
- *orientovatelná mimostředná* - kola s pohyblivým rámem (otočný kolem svislé osy), kde je osa otáčení kola položena mimo osu otáčení rámu [9]

Obrázek 2.1: Příklad pevného, orientovatelného centrického a orientovatelného mimostředného kola *(zdroj: převzato z [10] a [11]; upraveno)*

Druhou kategorií je *KRP s všesměrovými koly*. Tato řešení obvykle disponují lepší a jednodušší manévrovatelností v malých prostorech, to však na úkor pořizovací ceny. [9] Využívají se zde tyto typy kol:

- *švédská (mecanum)* - kola s šikmo položenými soudečky po obvodu, jejichž soustavy umožňují správným řízením pohyb KRP do všech směrů [12]
- *poly-wheel* - kola s principem podobným švédským kolům, rozdílem je kolmé uložení soudečků [13]
- *sférická* - kola kulového tvaru, jejichž natáčení probíhá pomocí dvou a více motorů

Obrázek 2.2: Příklad švédského kola, poly-wheel a sférického kola *(zdroj: převzato z [14], [15] a [16]; upraveno)*

2.1.1.3 Dělení podle operačního prostředí

Operačním prostředím je myšlen prostor, ve kterém má robot vykonávat svou činnost. V případě KRP to tedy je prostor, ve kterém se bude pohybovat. Rozlišujeme celkem tři typy operačních prostředí.

Prvním typem je tzv. *strukturované prostředí*. Jedná se o prostředí, které je KRP zcela známé a neobsahuje dynamické překážky. Plánování pohybu KRP je zde výrazně zjednodušeno, avšak na úkor jeho využitelnosti. [9]

Dalším typem je *polo-strukturované* či *částečně strukturované prostředí*. Zde KRP nemá úplný přehled o daném prostředí a je tedy sice schopen globálního plánování (např. cesta mezi dvěma body danými GPS souřadnicemi), avšak musí být vybaven mechanismy pro řešení nečekaných lokálních problémů (např. kontakt s dynamickou překážkou). [9]

Posledním typem je *nestrukturované prostředí*, kde má KRP velmi omezené (případně vůbec žádné) informace o prostředí a proto se musí spoléhat téměř výhradně na lokálně získané informace ze senzorů a rozhodovat se podle nich. [9]

Nutno dodat, že KRP často spadají, z hlediska operačního prostředí, do více kategorií. Obvykle lze tedy říci, že robot schopný pracovat v méně strukturovaném prostředí je rovněž schopen pracovat v prostředí více strukturovaném (v něm však nemusí dosahovat takové efektivity jako KRP pro něj uzpůsobené).

2.1.2 Řídicí jednotky a software

Řídicí jednotka je nejdůležitější součástí KRP. Z názvu je patrné, že jejím účelem je řízení činnosti prostředku, k čemuž je nutná komunikace se všemi subsystémy, a tedy lze řídicí jednotku považovat za centrální spojovací prvek celého KRP. Její činnost lze zjednodušeně popsat takto: ze zdrojových subsystémů (např. senzorika) jsou čtena

data, na jejich základě je řídicím programem určena následující činnost a jsou informovány výstupní subsystémy (např. pohon). Tyto kroky se neustále opakují během celé doby provozu KRP.

Důležitým prvkem jednotky je výše zmíněný řídicí program. Jeho vlastnosti jsou definovány požadavky na funkcionalitu prostředku. Je zřejmé, že komplexnější KRP vyžadují složitější program, který analogicky vyžaduje více hardwarových prostředků, což je nutné zvážit při výběru jednotky. Možností je mnoho, od nejjednodušších osmibitových mikrokontrolérů po pokročilá hradlová pole. Následuje popis těch nejběžněji užívaných.

2.1.2.1 Mikrokontroléry

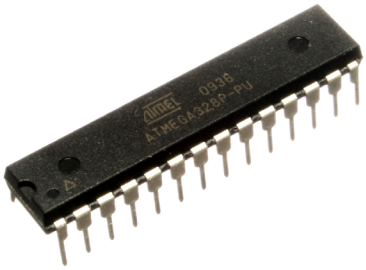
Mikrokontroléry, nebo též jednočipové počítače, jsou integrované obvody obsahující všechny komponenty nutné pro samostatnou činnost, tedy procesní jednotku, operační paměť, paměť programu a podpůrné obvody (např. A/D a D/A převodníky či obvody pro externí sběrnice). Vyznačují se obvykle nižším výkonem, velmi malou spotřebou a kompaktními rozměry. Z toho vyplývá, že jsou vhodné především pro řízení jednodušších KRP, nebo jako podpůrné jednotky, např. pro řízení jednotlivých subsystémů. [17]

Na trhu je k dispozici celá řada mikrokontrolérů pro různé aplikace. Jako příklad byly vybrány dva velmi populární s velmi odlišnými parametry.

ATmega328 je osmibitový mikrokontrolér z rodiny megaAVR od společnosti Microchip Technology. Obsahuje jedno RISC jádro s 32 obecně použitelnými pracovními registry, 2 KiB SRAM, 32 KiB programové flash paměti a 1 KiB EEPROM pro perzistentní data. Z hlediska rozhraní obsahuje 23 GPIO linek, dva osmibitové a jeden šestnáctibitový časovač/čítač, 6 PWM kanálů, osmikanálový desetibitový A/D převodník, RTC s vlastním oscilátorem, programovatelný Watchdog časovač s vlastním oscilátorem, analogový komparátor a sériová rozhraní USART, SPI a TWI. Výčet parametrů není úplný, pokrývá pouze ty nejdůležitější. [18]

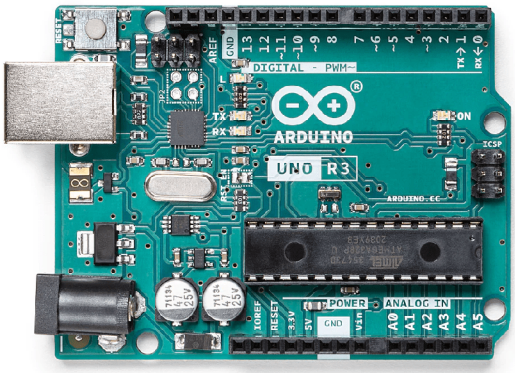
I přesto, že se zdaleka nejedná o nejvýkonnější zařízení v dané kategorii, je ATmega328 velmi populární a hojně užívaný mikrokontrolér, především v oblasti hobby a výukových řešení. Je tomu tak patrně především díky platformě *Arduino*[2], jež jej užívá u vlajkových modelů (Arduino UNO, Arduino Nano) jako hlavní komponentu. Díky tomu je ATmega328 obvykle první mikrokontrolér, se kterým se začátečníci setkávají a v budoucnu jej často užívají ve svých řešeních. Z využití v deskách Arduino plyne

[2]Projekt, v rámci kterého jsou vyvíjeny open-source mikrokontroléry a doprovodný software určené pro začátečníky.



Obrázek 2.3: ATmega328 *(zdroj: převzato z [19])*

řada výhod. Pravděpodobně tou největší z nich je softwarová podpora, v čele s Arduino Core a APL, díky které lze mikrokontroléry programovat velmi jednoduchým způsobem, vhodným i pro úplné začátečníky. Pro APL navíc existuje nepřeberné množství knihoven, které programování ještě více zjednodušují, obzvláště při práci s externími komponentami, jako jsou senzory, aktuátory či displeje.



Obrázek 2.4: Arduino UNO Rev3 *(zdroj: převzato z [20])*

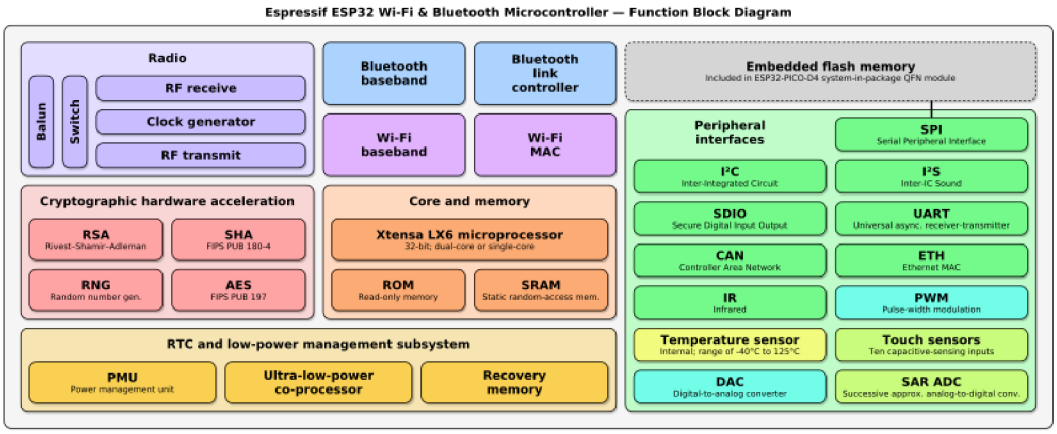
Vedle výše zmíněných výhod spojených s platformou Arduino lze zmínit i řadu vlastních samotnému mikrokontroléru. Především se jedná o poměrně nízkou cenu a spotřebu energie, dále širokou řadu dostupných pouzder a v souvislosti s dříve zmíněným častým použitím začátečníky i poměrně jednoduchou strukturu a s ní spojené snadné pochopení vnitřního fungování celého mikrokontroléru.

Rodina mikrokontrolérů AVR obsahuje kromě ATmega328 i další často užívaná zařízení. Ve skupině *megaAVR* jsou to mikrokontroléry s nižším množstvím SRAM, EEPROM a programové paměti *ATmega48/88/168* (x8, x znamená počet KiB programové paměti). Dále je zajímavá skupina *tinyAVR*, určená pro nenáročné aplikace. Jedná se o mikrokontroléry s menší pamětí, omezeným množstvím periferií a menšími pouz-

dry. Za zmínku stojí např. mikrokontrolér *ATtiny85*, jež je vyráběn v osmivývodovém pouzdře DIP8, o rozměrech 10 × 8 mm. [21][22]

ESP32 je 32bitový mikrokontrolér od společnosti Espressif Systems. Obsahuje jedno či dvou jádrový mikroprocesor Xtensa® LX6 s architekturou RISC [23], 520 KiB SRAM (320 KiB pro data a 200 KiB pro instrukce) [24], až 4 MiB programové flash paměti připojené přes QSPI (některé čipy nemají integrovanou flash paměť, případně mají jen 2 MiB), interní 8 MHz oscilátor, 2 skupiny časovačů, každá s vlastním Watchdog obvodem a 1 RTC časovač. Rozhraní ESP32 obsahuje mnoho prvků, pro zjednodušení budou vypsány pouze ty nejdůležitější: 34 GPIO linek, dvanáctibitový A/D převodník s 18 kanály, 2 osmibitové D/A převodníky, 1 motorový a 16 standardních PWM kanálů, 4 SPI, 2 TWI, 2 I^2S a 3 UART kanály, Wi-Fi operující v pásmu 2,4 GHz se 4 virtuálními rozhraními, Bluetooth v4.2 splňující *Low Energy* specifikace a Ethernet MAC rozhraní splňující standard IEEE 802.3-2008. [25]

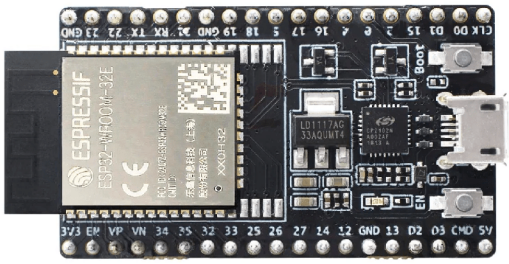
Za zmínku stojí také řešení spotřeby elektrické energie mikrokontroléru. Jelikož byl navržen s ohledem na možné použití v zařízeních, jež vyžadují minimální spotřebu, obsahuje pět různých napájecích režimů: *Active*, *Modem-sleep*, *Light-sleep*, *Deep-sleep* a *Hibernation*. Hojně užívaný je především režim *Deep-sleep*, ve kterém je v činnosti pouze *Ultra-Low-Power* koprocesor a mikrokontrolér dosahuje spotřeby pouze 10 µA. [26] Kromě úsporných režimů umožňuje ESP32 snížit spotřebu také pomocí individuálního řízení jednotlivých komponent, konfigurace taktovací frekvence a nastavení pracovního režimu Wi-Fi. [25]



Obrázek 2.5: Blokové schéma mikrokontroléru ESP32 *(zdroj: převzato z [27])*

ESP32 je v posledních letech velmi populárním mikrokontrolérem, k čemuž přispívá celá řada různých faktorů. Nejdůležitějším je patrně bezdrátová konektivita, která u jiných mikrokontrolérů ze stejné oblasti užití chybí. Dále pak velmi nízká cena (mnoh-

dy nižší než u ATmega mikrokontrolérů), široká škála dostupných vývojových desek [28] a výborná softwarová podpora. Společnost Espressif Systems nabízí open-source vývojový framework ESP-IDF [29], který umožňuje přímý přístup ke všem komponentám mikrokontroléru, a proto je oblíbený především u pokročilých vývojářů a v průmyslových řešeních. Pro méně pokročilé a hobby vývojáře jsou k dispozici běžné vývojové platformy Arduino a MicroPython, jež umožňují použití širokého spektra knihoven, jak je uvedeno v kapitole 2.1.2.1. [30]



Obrázek 2.6: Vývojová deska ESP32-DevKitC-32E *(zdroj: převzato z [31])*

Společnost Espressif Systems vyvíjí vedle ESP32 i další mikrokontroléry. Většinou se jedná o deriváty ESP32 se změnami v interních komponentách, cílící na konkrétní aplikace. Vedle nich stojí za zmínku také mikrokontrolér ESP8266, ze kterého skupina ESP32 vychází a díky kterému získaly mikrokontroléry ESP svou popularitu. V tabulce 2.1 je uveden stručný přehledem vlastností nejběžnějších ESP mikrokontrolérů.

2.1.2.2 Jednodeskové a průmyslové počítače

Mikrokontroléry jsou ideálními řídicími jednotkami pro nenáročné aplikace, kde je kladen důraz především na nízkou cenu a spotřebu elektrické energie. Vyžaduje-li však aplikace sofistikovanější nástroje, jako například univerzální operační systém či běhové prostředí pro některý z vyšších programovacích jazyků, jsou vhodným řešením jednodeskové a průmyslové počítače.

Jedná se o kompletní univerzální počítače, jejichž komponenty jsou úsporně uloženy ve skříni odpovídající plánovanému užití (průmyslové počítače) či přímo napájeny na jedné DPS (jednodeskové počítače). Průmyslové počítače se vyznačují vysokou mírou spolehlivosti, schopností dlouhodobé činnosti v náročném prostředí a dlouhou životností, s čímž jde ruku v ruce vyšší cena. Kromě toho také obvykle disponují širokou škálou rozhraní, a to těch běžných v oblasti mikrokontrolérů (např. UART, TWI, ...), osobních počítačů (např. USB, Ethernet, ...) i průmyslových systémů (např. CAN). Naproti tomu jednodeskové počítače lze považovat za jakousi kombinaci mikrokontrolérů a osobních počítačů. Vyznačují se, stejně jako mikrokontroléry, nízkou spotřebou

	ESP8266	**ESP32**	**ESP32-S2**	**ESP32-S3**	**ESP32-C3**	**ESP32-C6**
Použití	Nenáročné aplikace	Obecné aplikace	Aplikace vyžadující bohaté rozhraní	AI a HMI aplikace	Nízkonákladové aplikace	Aplikace vyžadující nízkou spotřebou
Rychlost přenosu Wi-Fi [Mb/s]	75	150	150	150	150	150
Bluetooth	Ne	BR/EDR + BLE v4.2	Ne	BLE v5.0	BLE v5.0	BLE v5.3
Thread/ Zigbee	Ne	Ne	Ne	Ne	Ne	Ano
Počet GPIO pinů	11	33	43	45	22	30
Velikost SRAM [KiB]	160	520	320	512	400	512
Velikost ROM [KiB]	0	448	128	384	384	320
Procesor	Tensilica L106 32-bit	2x Tensilica Xtensa 32-bit LX6	Tensilica Xtensa 32-bit LX7	Tensilica Xtensa 32-bit LX7	RISC-V 32-bit	RISC-V 32-bit
Taktovací frekvence [MHz]	160	240	240	240	160	160
Počet UART	1	3	2	3	2	3
Počet SPI	2	4	4	4	3	3
Počet TWI	1	2	2	2	1	2
Počet I^2S	1	2	1	2	1	1
Napájecí napětí [V]	2.7 - 3,6	2,3 - 3,6	3,0 - 3,6	3,0 - 3,6	3,0 - 3,6	3,0 - 3,6

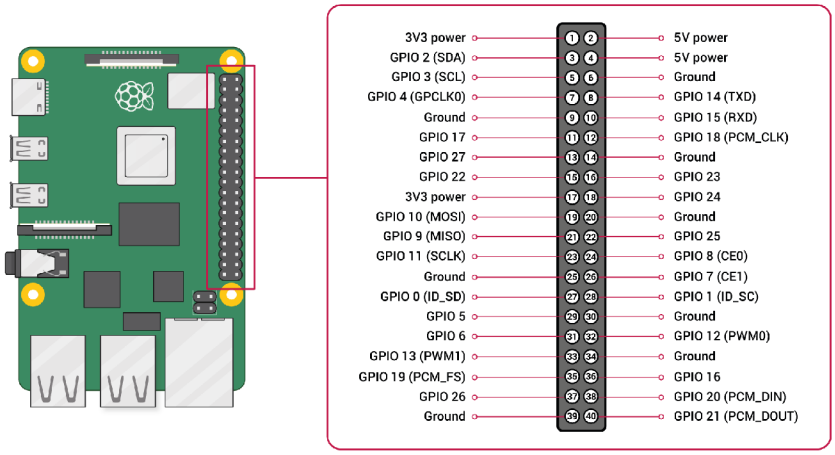
Tabulka 2.1: Porovnání vlastností mikrokontrolérů ESP [32][33][34][35]

a cenou (i když v porovnání s běžnými mikrokontroléry je cena řádově vyšší), malými rozměry a přítomností běžných mikrokontrolérových rozhraní. K tomu navíc disponují vysokým výpočetním výkonem, grafickými procesory a rozhraními běžně užívanými u osobních počítačů. Důležitým aspektem, společným pro obě skupiny, je možnost provozovat na nich vedle jednoduchých vestavěných operačních systémů (např. FreeRTOS) i univerzální operační systémy (např. GNU/Linux), jež razantně zvyšují možnosti výsledné aplikace. [36][37]

Následuje popis reprezentativních příkladů z obou skupin.

Raspberry Pi je skupina jednodeskových počítačů od britské společnosti Raspberry Pi Ltd., původně vyvinutých pro podporu výuky informatiky na školách. Skupina se dělí na dvě větve: hlavní, jež obsahuje výkonné modely a *Zero* větev s důrazem na nízkou cenu.

Zařízení patřící do hlavní větve jsou plnohodnotné univerzální počítače na DPS o rozměru 85 × 56 mm. [38] Nejnovějším modelem této větve je *Raspberry Pi 5* řízený SoC BCM2712 (obsahuje čtyřjádrový procesor ARM Cortex-A76, dvanáctijádrový VideoCore VII grafický procesor, LPDDR4X rozhraní, PCI Express x1 a x4, HDMI řadič s podporou dvou 4kp60 displejů, HEVC dekodér a procesor obrazového signálu) [39] s operační pamětí 4/8 GB LPDDR4X. Z hlediska rozhraní obsahuje Raspberry Pi 5 dva USB 2.0 a dva USB 3.0 porty, RJ-45 1 Gb/s ethernetový port, dvoupásmové 802.11ac Wi-Fi, Bluetooth 5.0 LE, dva micro HDMI 4kp60 porty, dva MIPI CSI/DSI porty se čtyřmi linkami (pro kameru nebo displej), microSD card slot a standardní 40 pinový GPIO port známý z předešlých modelů (obsahuje rozhraní běžná ve vestavěných systémech, viz obrázek 2.7). [40]



Obrázek 2.7: 40 pinové GPIO rozhraní *(zdroj: převzato z [41])*

Zařízení z větve *Zero* obsahují méně výkonné komponenty, díky čemuž jsou násobně levnější, dosahují výrazně nižší spotřeby a též rozměrů (DPS o velikosti 65 × 30 mm). Aktuálně nejnovějším modelem této řady je *Raspberry Pi Zero 2 W* se čtyřjádrovým procesorem ARM Cortex-A53 a 512 MB SDRAM. Rozhraní modelů Zero je oproti hlavním modelům rovněž jednodušší; *Zero 2* obsahuje jeden micro USB 2.0 port, 802.11 b/g/n Wi-Fi, Bluetooth 4.2 LE, jeden mini HDMI port, CSI-2 kamerový port, microSD card slot a výše zmíněný 40 pinový GPIO port. [42]

Na zařízeních z obou větví lze provozovat běžné univerzální operační systémy,

mají-li podporu pro procesory ARM. To znamená většinu distribucí GNU/Linux, FreeBSD, Windows 10/11 a další. Nejčastěji používaným systémem je *Raspberry Pi OS*, což je linuxová distribuce vyvíjená přímo Raspberry Pi Foundation. Je založená na ARM verzi Debianu a obsahuje modifikované prostředí LXDE, předinstalovanou sadu aplikací s nízkými nároky na prostředky a také sadu knihoven pro práci s rozhraním zařízení, především pro podporu GPIO portu. [43]

Jak bylo zmíněno výše, Raspberry Pi bylo původně vyvinuto pro podporu výuky informatiky. Současná oblast využití je však mnohem širší. Běžně se využívá jako domácí server, jádro různých IoT systémů či řídicí jednotka u všech druhů mobilních robotů.

POC-200 je průmyslový počítač od společnosti Neousys Technology Inc. Je určen pro širokou škálu průmyslových aplikací jako např. řízení robotů, HMI, bezpečnostní systémy či inspekční systémy. [44]

Obrázek 2.8: POC-200 *(zdroj: převzato z [45])*

Řízen je čtyřjádrovým procesorem Intel® Atom™ E3845 s integrovaným grafickým procesorem Intel® HD Graphics a až 8 GB DDR3L operační paměti. Z hlediska rozhraní obsahuje tři USB 3.0 a jeden USB 2.0 port, dva softwarově programovatelné RS-232/422/485 porty, dva standardní RS-232 porty, dva 1 Gb/s ethernotové porty s podporou PoE, jeden DVI-I port, výstupní audio port, mini PCI Express slot, interní SATA port a čtyřkanálový DIO port. [46]

Velkou výhodou oproti Raspberry Pi a dalším jednodeskovým počítačům je mechanická konstrukce zařízení. Ta umožňuje provoz v teplotách od -25 °C do 70 °C, při

vzdušné vlhkosti od 10 do 90 % a v prostředí, kde hrozí otřesy a nárazy. Navíc díky velkému pasivnímu chladiči není nutno zařízení chladit aktivně, což snižuje spotřebu a hlučnost při provozu. [44][46]

Vzhledem k tomu, že architektura přítomného procesoru je x86-64, která se užívá v drtivé většině univerzálních počítačů, lze na POC-200 provozovat všechny běžné operační systémy. Přímo výrobcem testované jsou Windows 7, 8 a 8.1, ve 32 i 64 bitové verzi a Ubuntu 14.04. [46]

2.1.3 Komunikace

Problematiku komunikace lze ve spojitosti s KRP rozdělit do dvou oblastí. První z nich je oblast vnitřní komunikace, která zahrnuje předávání příkazů a přenos dat mezi jednotlivými komponentami KRP. Druhou je oblast vnější komunikace, do níž spadá interakce mezi KRP a uživatelem a rovněž komunikace s externími systémy, jako např. systémy pro automatické ovládání či další KRP. Každá z oblastí představuje specifickou množinu problémů, které budou popsány dále.

2.1.3.1 Vnitřní komunikace

Jak již bylo řečeno, vnitřní komunikací se myslí vzájemné propojení jednotlivých komponent KRP a přenos dat či příkazů mezi nimi. Z toho vyplývá několik základních vlastností vnitřní komunikace:

- *Krátká vzdálenost* - jedná se o komunikaci na krátkou vzdálenost, řádově maximálně desítky centimetrů. Lze proto využít drátové spoje či propojení přímo na DPS.
- *Malé zprávy* - subsystémy KRP obvykle generují spíše menší zprávy, zato však ve velkém množství. Tomu by měl odpovídat užitý komunikační protokol.
- *Nízká komplexita* - některé komunikující komponenty KRP mohou mít velmi nízký procesní výkon, proto je žádoucí, aby byla komunikace mezi nimi co nejjednodušší a zbytečně je nezatěžovala.

Při návrhu systému vnitřní komunikace je však nutno zohlednit nejen tyto aspekty, ale stejnou měrou i specifika konkrétního KRP. Lze při tom s výhodou využít některé z mnoha dostupných komunikačních rozhraní a souvisejících protokolů. Následuje krátký popis některých z nich.

UART (Universal Asynchronous Receiver-Transmitter) je standard definující sériové asynchronní rozhraní pro vzájemnou komunikaci primárně dvou rovnocenných zařízení, však s možností úpravy na komunikaci typu *single master, multi slave*. Jedná se o komunikaci plně duplexní. Standard samotný definuje pouze způsob a strukturu přenosu a dva základní vodiče pro přenos použité. Další, doplňkové, vodiče a vlastnosti přenášeného signálu jsou definovány komplementárními standardy, nejčastěji RS-232 a RS-485. [47][48]

Komunikace probíhá v paketech o velikosti 7-13 b. Její začátek je indikován vystavením log. 0 (tzv. *start bit*) na sběrnici. Poté následuje přenos zbylých bitů paketu, které příjemce ukládá v posuvném registru. Po přijetí tzv. *stop bitu/bitů* se provede kontrola parity a je-li paket v pořádku, je přesunut z registru do bufferu. Následně je systém informován o tom, že v bufferu jsou k dispozici data a přenos je považován za ukončený. [48]

Hlavní výhodou UART je především jednoduchost, a to jak z pohledu hardwarové realizace, tak doprovodného softwarového vybavení. Díky tomu jsou UART rozhraní přítomna prakticky na každém zařízení užívaném v oblasti KRP, od nejjednodušších mikrokontrolérů, po průmyslové počítače.

Hlavní nevýhodou je pak patrně omezení komunikace na dvě zařízení (v základním provedení), což ovšem hraje roli pouze v některých aplikacích. Dále lze zmínit ještě poměrně nízkou maximální rychlost přenosu, v porovnání s dalšími druhy rozhraní.

TWI (Two-Wire Interface), známé též jako I^2C (jedná se téměř o totožná a vzájemně kompatibilní rozhraní, ale název I^2C byl do roku 2014 ochranou známkou firmy Philips) je sériové synchronní half-duplexní rozhraní typu *multi master, multi slave*, běžně však používané jako *single master*. Původně bylo navrženo pro komunikaci v rámci jedné DPS (např. propojení procesoru s EEPROM, A/D a D/A převodníky, atd.), používá se však úspěšně i k propojení oddělených zařízení. [49][50]

Rozhraní se fyzicky skládá ze dvou vodičů, jimiž jsou propojena všechna zařízení: *SDA* pro datový přenos a *SCL* přenos hodinového signálu. Vnitřní zapojení rozhraní je obvykle typu *otevřený kolektor*, což znamená, že je nutné obě linky připojit k napájecímu napětí pomocí tzv. *pull-up rezistorů*. Běžně užívané hodnoty napájecího napětí jsou +5 V a +3,3 V, lze však použít i jiné. [50]

Vzhledem k tomu, že jsou všechna zařízení propojena jednou sběrnicí, je nutné odlišit příjemce zpráv. K tomu slouží adresa o 7 či 10 bitech jež je z výroby přiřazena každému zařízení. Předpokládá-li se použití více kusů téhož zařízení, bývá část adresy nastavitelná, např. pomocí tzv. jumperů. [51]

Hlavními výhodami jsou na množství připojených zařízení nezávislý počet vodičů,

jednoduchost použití, široká softwarová podpora a poměrně vysoká rychlost přenosu.

SPI (Serial Peripheral Interface) je sériové asynchronní plně duplexní rozhraní typu *single master, multi slave*. Užívá se především pro komunikaci mezi kontrolérem a externí pamětí, displejem, převodníky, či dalším kontrolérem. [52][53]

Fyzicky se SPI skládá minimálně ze čtyř vodičů:

- *MOSI (master out, slave in)* - přenos dat od řídicího zařízení k podřízenému
- *MISO (master in, slave out)* - přenos dat od podřízeného zařízení k řídicímu
- *SCL* - hodinový signál
- *SS1...n (slave select)* - slouží k výběru podřízeného zařízení, se kterým bude řídicí komunikovat

Podřízená zařízení mají pouze jeden SS vodič, kdežto řídicí zařízení potřebují jeden pro každé připojené zařízení. Důvodem je, že adresace, tedy výběr podřízeného zařízení, se kterým řídicí komunikuje, zde probíhá přivedením logické 0 na SS výstup podřízeného zařízení. To je hlavní nevýhodou SPI, zvláště pokud je předpokládán vyšší počet připojených zařízení. [52][53]

CAN (Controller Area Network) je sériové asynchronní rozhraní typu *multi master, multi slave*. Používá se primárně pro vnitřní komunikaci v automobilech, průmyslové automatizaci a také v systémech reálného času. [54]

Adresace zde probíhá odlišně od předešlých rozhraní. Adresa (identifikátor) se zde nepřiděluje zařízení, nýbrž přímo zprávě, kterou obdrží každé zařízení připojené ke sběrnici a samo na základě identifikátoru rozhodne, zdali zprávu přijme či nikoliv. Kromě toho identifikátor určuje prioritu zpráv. [54]

Velkou výhodou CAN rozhraní jsou kontrolní mechanismy pro detekci chyb a kolizí a obecně robustnost přenosu. S tím se však pojí hlavní nevýhoda, kterou je poměrně složité použití a vysoká cena hardwaru.

2.1.3.2 Vnější komunikace

Vnější komunikace má oproti té vnitřní poněkud širší záběr, zahrnuje totiž celou řadu scénářů. Z těch běžnějších: komunikaci mezi KRP a uživatelem, respektive rozhraním, skrz které uživatel s KRP interaguje, dále pak vzájemnou komunikaci mezi kooperujícími KRP nebo při majákové lokalizaci komunikaci KRP s majáky. O každém z těchto scénářů toho lze napsat mnoho, vzhledem k tématu práce však bude stručně popsán jen první z nich. [55]

Komunikace s uživatelským rozhraním probíhá takřka výhradně bezdrátově (výjimkou mohou být jednoduchá tlačítková rozhraní integrovaná přímo do konstrukce KRP). Na fyzické vrstvě se obvykle využívá standard *IEEE 802.11* neboli Wi-Fi. [56] Na aplikační vrstvě je běžný protokol *HTTP* a na něm postavená *REST* rozhraní, případně s HTTP kompatibilní protokol *WebSocket*, který umožňuje komunikaci směrem od serveru ke klientovi bez nutnosti zaslání požadavku od klienta.

2.1.4 Senzorika mobilních robotů

Následující kapitola čerpá z práce [57].

Lze říci, že senzorický subsystém je naprosto klíčovou součástí KRP, jelikož slouží jako primární (a většinou i jediný) zdroj dat o okolí a interním stavu prostředku. Jeho výstupy jsou základním předpokladem pro fungování řídicího subsystému a tedy i celého KRP. Elementárním prvkem tohoto subsystému je *senzor*, což je zařízení, jehož úkolem je převádět fyzikální vlastnosti okolního prostředí na elektrické signály, které lze měřit a následně interpretovat jejich hodnoty, jakožto hodnoty snímané veličiny. Senzory lze podobně jako KRP klasifikovat podle různých kritérií, následují ta nejběžnější z nich.

Prvním kritériem je **oblast měření**, čili, zdali se jedná o senzor interní (proprioceptivní), či externí (exteroceptivní). Úkolem vnitřních senzorů je snímat pouze vlastnosti a stavy KRP samotného, např. rychlost otáčení kol. Naproti tomu externí senzory snímají veličiny, které závisí jak na aktuálním stavu KRP, tak na vlastnostech jeho okolí, např. přítomnost a pozice překážky v momentálním směru cesty prostředku.

Dále lze senzory klasifikovat podle **zdroje budícího signálu**. V rámci této metody rozlišujeme senzory pasivní a aktivní. Pasivní senzor mění svůj výstup na základě budícího signálu vyslaného externím zdrojem (tj. různým od senzoru samotného), lze tedy říci, že senzor je pasivní, vyžaduje-li napájení pouze pro převod a zesílení vstupního signálu. Příkladem může být termistor nebo PIR čidlo. Aktivní senzor naproti tomu sám generuje budící signál, vysílá jej a následně měří reakci okolí na něj. Je tedy zřejmé, že kromě napájení pro převod a zesílení signálu vyžaduje aktivní senzor rovněž napájení emitoru budícího signálu. Příkladem aktivního senzoru je LiDAR, který emituje laserový paprsek a následně snímá jeho odraz od překážky.

Dalším klasifikačním kritériem je **druh měření**, čili to, jakou informaci lze měřením získat. Nelze obecně říci, že se jedná o klasifikaci na základě měřené veličiny (i když tomu tak často je), jelikož v některých případech lze z jedné veličiny odvodit více informací, jako např. z rychlosti otáčení kola vyplývá jak rychlost prostředku, tak jeho aktuální poloha v prostoru.

Druhů měření je celá řada, následuje krátký výčet těch nejběžnějších, spolu s příklady senzorů:

- *Fyzický kontakt* - taktilní spínač, nárazník
- *Vzdálenost* - LiDAR, time-of-fligt kamera, sonar
- *Globální orientace a pozice* - Galileo/GPS/GLONASS modul, magnetometr
- *Relativní orientace a pozice* - gyroskop, Hallova sonda, rotační enkodér (pro měření otáček kola)
- *Rychlost* - akcelerometr, rotační enkodér
- *Barva prostředí* - reflexní optočlen, RGB kamera, senzor detekce čáry

Následuje popis vybraných, hojně užívaných typů senzorů.

2.1.4.1 LiDAR

LiDAR (Light Detection And Ranging) je aktivní senzor využívající laser (obecně zdroj koherentního světla) k měření vzdálenosti od objektů v prostoru. Běžně LiDARy vysílají jednobodový paprsek, a tedy mohou měřit vzdálenost pouze jednoho bodu v konkrétní okamžik. Doba měření je však velmi krátká, díky čemuž lze oblast měření rozšířit rychlým vychylováním paprsku. K tomu se užívají vychylovací zrcadla nebo pohyblivé konstrukce, ke kterým je LiDAR připevněn. Takto lze i s jedním emitorem docílit 2D či 3D měřeného vzorku.

Jiný způsob rozšíření snímané oblasti využívají tzv. *solid-state LiDARy*. Ty namísto komponent pro mechanické vychýlení paprsku obsahují OPA (Optical Phased Array), což je dvourozměrné pole optických antén, umístěných velmi blízko k sobě. Ty jsou individuálně fázově regulovány, čímž se mění směr vyzařovaného paprsku a je tak docíleno vícerozměrného snímku.

Z pohledu principu, kterým je zjišťována vzdálenost, existují dva hlavní typy LiDARů. **LiDARy založené na principu měření fázového posuvu** (nazývané taky *continuous-wave LiDARy*) vysílají souvislý světelný paprsek a následně měří rozdíl jeho fáze a fáze přijatého paprsku, odraženého od objektu.

Druhým typem jsou **time-of-flight LiDARy** (nebo též *pulse-based LiDARy*), jež vysílají místo souvislého paprsku jednotlivé pulzy a následně měří dobu letu světla ke snímači, po odražení od objektu v prostoru.

LiDARy jsou momentálně jedním z nejpoužívanějších typů senzorů pro detekci překážek a navigaci v oblasti KRP. Jejich hlavní výhodou je vysoká přesnost a dosah

měření. Naproti tomu hlavní negativní aspekty jsou vysoká cena a silný vliv malých částic v oblasti měření (jako např. prach a mlha). [57]

2.1.4.2 Sonar

Sonar (Sound Navigation and Ranging) je aktivní senzor velmi podobný LiDARu. Rozdílem je, že místo laserového paprsku využívá zvukové vlny, obvykle o frekvenci vyšší než 20 kHz, čili ultrazvuk. Princip činnosti je s LiDARem shodný, tedy emitor vysílá zvukové vlny, ty se odrážejí od překážek, dopadají na snímač a následně je měřen čas mezi vysláním a dopadem, který je nakonec převeden na vzdálenost.

Sonary jsou spolu s LiDARy nejpoužívanějšími senzory pro detekci překážek u KRP. Preferovány jsou především díky nízké ceně a větší oblasti měření, bez nutnosti vychylování (což je ovšem výhodou pouze v některých aplikacích). Nevýhodami jsou naopak nízká přesnost a kratší dosah měření. [57]

2.1.4.3 Nárazník

Nárazníky jsou nejjednodušším typem senzoru pro detekci překážek. Princip jejich činnosti je shodný se spínačem (což je obvykle jediná akční komponenta nárazníku), jehož sepnutí je interpretováno jako kontakt s překážkou. Používají se obvykle jako záchranné senzory pro případ, kdy dojde k selhání či chybné detekci (*false negative*) pokročilejších senzorů, případně jako hlavní navigační senzory u jednoduchých KRP, jako např. robotické vysavače. [57]

2.2 Řízení KRP

Kapitola *Řízení KRP* čerpá z práce [58].

Návrh řízení je velmi důležitou součástí celkového návrhu prostředku, jelikož jeho výstup umožňuje vykonávání základního úkolu prostředku, tedy pohybu v daném prostředí. Skládá se z následujících úkolů:

- *Získávání informací o prostředí a KRP samotném*
- *Lokalizace prostředku*
- *Tvorba mapy prostředí*
- *Plánování cesty*
- *Řízení pohybu*

Vzájemný vztah těchto úkolů lze popsat takto: KRP za pomoci senzorického subsystému snímá vlastní provozní informace a informace o prostředí a na jejich základě určí svou pozici v prostoru. Ta je spolu s vytvořenou mapou a zadáním úkolu použita k výpočtu cesty do cíle. Cesta je poté převedena na příkazy, které zpracovává pohybový kontrolér a na jejich základě řídí aktuátory (motory) prostředku.

2.2.1 Získávání informací o prostředí a KRP

Získávání informací je základním úkolem v procesu řízení prostředku, jelikož získané informace jsou nezbytné pro vykonávání úkolů následujících. Provádí ho senzorický subsystém prostřednictvím svých elementárních jednotek - senzorů.

Hlavními výzvami tohoto úkolu jsou *přesnost* a *spolehlivost* měření. Problémy z nich vyplývající lze rozdělit do dvou kategorií: *šum senzoru* a *aliasing*.

Šum senzoru způsobují obvykle proměnlivé vlivy prostředí, které senzor není schopen zpracovat. Může se jednat např. o nevhodný odrazný povrch při sonarovém snímání - je-li povrch hladký a šikmý, je většina vyslaného impulzu rozptýlena a do senzoru nedorazí odraz o dostatečné síle. Dále např. u kamerových snímačů se může jednat o špatné zaostření či nevyhovující světelné podmínky, což rovněž vyústí v nepoužitelné vzorky.

Aliasing je možné popsat jako nedostatečnost měření k určení aktuálního stavu prostředku. Jinými slovy, obvykle je k určení stavu prostředku nutno provést více než jedno nezávislé měření nebo provádět měření více senzory současně - tedy vztah mezi stavy a měřeními je $1 : n$. [59]

Tyto problémy je nutno kompenzovat, a to vždy s ohledem na to, k čemu mají být data právě užita. Tedy u každého z kroků řízení existují kompenzační metody pro data, která daný krok využívá.

2.2.2 Lokalizace prostředku

Jak již bylo řečeno, k lokalizaci prostředku se využívají naměřená data ze senzorického subsystému. Různé metody lokalizace vyžadují různý typ snímaných dat. Lokalizační metody lze rozdělit do dvou skupin: *relativní* a *absolutní*.

Relativní lokalizace (též nazývaná *odometrie*) se provádí měřením ujeté vzdálenosti od daného počátečního bodu. Využívají se při tom obvykle data z inkrementálních enkodérů či Hallových sond umístěných na kolech KRP. Problémem je, že toto měření

není přesné (dochází ke smykům a prokluzům kol) a vzhledem k tomu, že se jedná o inkrementální metodu, celková chyba měření roste s růstem ujeté vzdálenosti. Zdroje chyb mohou být dvojího druhu:

- *Systémové* - projevují se především ve vnitřním prostředí a jimi způsobená chyba se obvykle přičítá během celé jízdy. Jedná se např. o špatné usazení kol, rozdílný průměr kol, rozdíl mezi nominálním a skutečným rozvorem kol, nízká vzorkovací frekvence a rozlišení enkodéru.
- *Nesystémové* - působí především ve venkovním prostředí. Jedná se např. o vnější síly působící na prostředek a nerovný či kluzký podklad.

Absolutní lokalizace umožňuje určit přesnou polohu prostředku v prostoru. Využívají se při ní obvykle data z globálních navigačních systémů, komunikace s aktivními majáky či čtení značek umístěných v prostředí. Možné metody určení polohy jsou:

- *Triangulace*
- *Trilaterace*
- *Mapové porovnání*

2.2.3 Tvorba mapy prostředí

Tvorba mapy prostředí je velmi podstatný úkol pro roboty operující v *částečně strukturovaném* a zvláště pak v *nestrukturovaném prostředí*. Spočívá v inkrementálním zjišťování vlastností prostředí (především pozic překážek) z dat získaných ze senzorického subsystému. Prostředek si udržuje mapu s takto získanými vlastnostmi v paměti a postupně ji zpřesňuje. Jak bylo řečeno výše, data ze senzorického subsystému je nutno kompenzovat. Běžně užívané metody kompenzace při tvorbě mapy prostředí jsou:

- *Bayesovský odhad*
- *Kalmanův odhad*
- *Částicový filtr*
- *Neuronové sítě*
- *Fuzzy metody*

Tvorba mapy prostředí je úzce spjatá s lokalizací prostředku a obvykle bývají studovány dohromady. Pokud robot operuje v kompletně nestrukturovaném (a tedy neznámém) prostředí, nazývá se tento problém *SLAM* (Simultaneous Localisation and Mapping).

2.2.4 Plánování cesty

Proces plánování cesty spočívá v kombinování *mapy prostředí*, *pozice robotu* a *zadání vykonávané úlohy* a jeho výstupem je cesta z aktuální pozice do cíle daného zadáním. Centrálním pojmem při plánování je *konfigurace robotu*. Jedná se o uspořádanou n-tici jež obsahuje parametry určující pozici prostředku. Dalším důležitým pojmem je *konfigurační prostor*, což je množina všech konfigurací, které mohou v daném prostoru nastat. Jeho podmnožinou je takzvaný *volný konfigurační prostor*, který obsahuje všechny konfigurace, jež nejsou v kolizi s žádnou z překážek v prostředí. V tomto kontextu lze plánování cesty popsat jako snahu o nalezení takové souvislé posloupnosti konfigurací, která povede od aktuální do cílové konfigurace a bude obsahovat pouze konfigurace z volného prostoru. Plánování může být dvojího typu: *globální* a *lokální*.

Globální plánování vytváří cestu z úplného počátku cesty do cíle, a to před začátkem pohybu robotu. Takto vytvořená cesta není dále nijak upravována. Lze jej proto použít pouze v plně strukturovaném prostředí, které neobsahuje žádné dynamické překážky.

Lokální plánování se naproti tomu vykonává periodicky, dokud prostředek nedorazí do cíle. V každé iteraci se využívá aktuální poloha a momentální stav mapy prostředí. Díky tomu může prostředek reagovat na dynamické či dříve neznámé překážky a přizpůsobovat cestu tak, aby se jim vyhnul.

Běžně užívané algoritmy pro plánování cesty jsou následující:

- *A* algoritmus* je heuristický algoritmus typu *best-first search*. Jeho princip je velmi podobný Dijkstrovu algoritmu, ale na rozdíl od něj postupuje po nejslibnějších stavech, čímž se výpočet značně zrychluje. Nevýhodou je nutnost kompletního přepočtu při každé změně mapy prostředí. [60]

- *D* Lite algoritmus* řeší problémy A* algoritmu. Cestu počítá od cíle do aktuální pozice a využívá při tom dříve vypočtenou cestu. Díky tomu je mnohem efektivnější při opakovaném přepočítávání. [60]

- *Probabilistic roadmap algoritmus* je na rozdíl od předchozích stochastický algoritmus. Jeho princip se skládá ze dvou kroků. V prvním jsou náhodně vybrány

konfigurace z konfiguračního prostoru, sousední jsou vzájemně propojeny a tak je vytvořen graf. V druhém kroku je pomocí heuristických metod spočítána optimální cesta tímto grafem k cíli. [60]

Důležitou součástí plánování cesty je **vyhýbání se překážkám**. Existuje celá řada algoritmů, které reakci na kontakt s překážkou řeší, toto jsou nejběžnější z nich:

- *Bug algoritmus* je jedním z nejstarších algoritmů pro vyhýbání se překážkám. Robot při něm sleduje přímou cestu do cíle, dokud nenarazí na překážku a následně ji začne objíždět podél okraje. Další činnost se liší podle konkrétní varianty algoritmu:
  - *Bug-1* - v tomto případě objede prostředek celou překážku dokola a poté spočítá vzdálenost od každého projetého bodu do cíle. Bod s nejkratší vzdáleností je vybrán jako počátek nově propočítané cesty do cíle. [61]
  - *Bug-2* - v každém bodě se zjišťuje, zdali aktuální pozice neleží na původní cestě. Jakmile je taková pozice nalezena, prostředek pokračuje po původní cestě dál. [61]
  - *DIST-Bug* - v každém bodě počítá vzdálenost od cíle pro aktuální a následující pozici. Je-li vzdálenost následující pozice vyšší než aktuální, je proces vyhýbání se překážce považován za ukončený a dochází k přepočítání cesty z aktuální pozice do cíle. [62]

  První dvě varianty jsou velmi jednoduché a spolehlivé, naproti tomu u třetí je největší výhodou efektivita. Obecně platí, že Bug algoritmus je vhodný především do jednodušších prostředí s nižší hustotou překážek. [60]

- *Artificial Potential field algoritmus* pracuje s konceptem tzv. *potenciálových polí*. Robot a překážky jsou považovány za kladně nabité (navzájem se odpuzují) a cíl za záporně nabitý (s robotem se přitahují). Síla, se kterou na sebe jednotlivá pole působí, je dána jejich vzájemnou vzdáleností. Nevýhodou tohoto algoritmu je náchylnost k uváznutí v lokálním minimu v případě, že přitažlivá síla cíle je menší než součet odpudivých sil překážek před prostředkem. Kritické jsou především symetrické překážky a překážky tvaru U. [61]

- *Metoda Vector Field Histogram* se skládá ze tří kroků. Nejprve je vytvořen 2D histogram okolí robotu, který je průběžně aktualizován na základě dat ze senzorů. Dále je 2D histogram převeden na 1D a poté na polární histogram. Nakonec je zvolen sektor s nejnižší (polární) hustotou překážek a nová cesta je vypočtena skrz něj.

2.2.5 Řízení pohybu

Proces řízení pohybu je posledním krokem v cyklu řízení KRP. Jeho úkolem je převést vypočtenou cestu na posloupnost příkazů pro aktuátory prostředku tak, aby prostředek cestu optimálně následoval. Využívají se k tomu různé metody, obvykle nazývané *kontroléry*. Ty mohou být dvojího typu: *open-loop* a *closed-loop*. [60]

Open-loop kontroléry nevyužívají při řízení *zpětnou vazbu*, což znamená, že systém sám není schopen reflektovat úspěšnost následování vstupní cesty. Jako kompenzace zde slouží lidský operátor, který sám úspěšnost kontroluje a na základě toho zadává řídicí instrukce. Nevýhodou této metody je velká náchylnost na chyby modelování a nutná přítomnost operátora během řízení. [60][63]

Closed-loop kontroléry jsou naproti tomu založené na principu zpětné vazby. To znamená, že odchylka od naplánované cesty je představena jako vstup a kontrolér na ni tak může bezprostředně reagovat. Cílem je dosáhnout tzv. *stabilního stavu*, což je stav, při kterém je odchylka dlouhodobě nulová. [60]

Běžně užívané closed-loop kontroléry jsou následující:

- *PID kontrolér* je nejtypičtějším příkladem kontroléru se zpětnou vazbou. Jeho princip spočívá ve výpočtu rozdílu mezi parametry požadované a skutečné konfigurace (získané právě skrz zpětnou vazbu) nazývaného *odchylka*, který je následně předán výpočetním složkám, jež z něj vypočítají následující konfiguraci. [60] Výpočetní složky jsou:
  - *Proporcionální (P) složka* přivádí na výstup přímo část aktuální odchylky. Její velikost je dána konstantou K_p. Výpočet složky je tedy následující $P = K_p e(t)$, kde $e(t)$ je aktuální odchylka. [60]
  - *Integrační (I) složka* akumuluje (integruje) všechny předešlé hodnoty odchylky. Zvyšování hodnoty konstanty K_i vede k minimalizaci odchylky ve stabilním stavu, ale zároveň k zesílení oscilací. Složka je počítána takto: $I = K_i \int_0^t e(t)\, dx$. [60]
  - *Derivační (D) složka* funguje jako odhad budoucího vývoje odchylky na základě jejího aktuálního trendu. Používá se k tlumení kmitů odchylky, ale příliš vysoká hodnota konstanty K_d vede k velké odchylce ve stabilním stavu. Výpočet složky je $D = K_d \frac{de(t)}{dt}$. [60]

  Vedle plnohodnotného PID kontroléru se běžně užívají rovněž zjednodušené varianty, ve kterých bývá jedna ze složek vynechána (např. PI kontrolér).

- *Fuzzy kontrolér* využívá principy fuzzy logiky ke zjednodušení nelineárních problémů (jako právě řízení pohybu) a kompenzaci nepřesností senzorických vstupů. Rovněž jej lze použít v kombinaci s jinými typy kontrolérů ke zvýšení inteligence systému. [60] Rozhraní kontroléru je obvykle následující:
  - *Vstupy* kontroléru jsou odchylky ve vzdálenosti a natočení od požadované cesty. [60]
  - *Výstupy* jsou běžně PWM signály pro jednotlivé motory prostředku. [60]

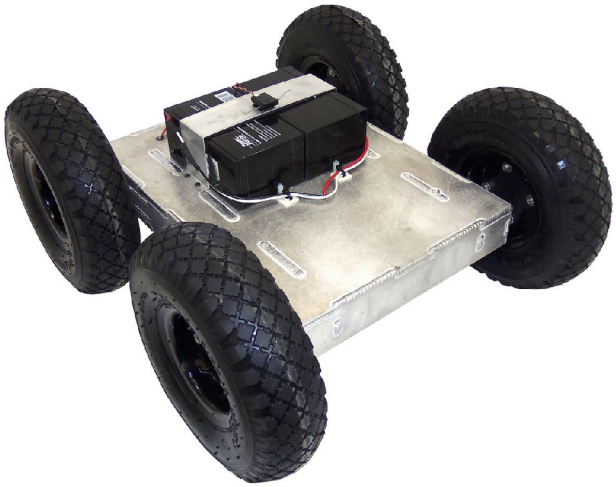
2.3 Analýza dostupných řešení

Na trhu je k dispozici nepřeberné množství KRP pro různé aplikace, následující analýza se proto vzhledem k orientaci práce zaměří na univerzální prostředky s vyšší nosností. Před shrnutím výsledků analýzy budou stručně představeny reprezentativní příklady KRP z oblasti průmyslových, akademických a hobby řešení.

2.3.1 Průmyslové řešení

IG42-DB4 je čtyřkolový diferenciálně řízený robot od společnosti *SuperDroid Robots*. Je navržen jako platforma bez řídicího a senzorického subsystému, a je tedy velmi vhodný jako základ pro návrh vlastního plnohodnotného robotu. Poháněn je čtyřmi samostatně řízenými motory, které jsou s nápravami kol spojeny řetězovými převody. Nápravy jsou uloženy v kuličkových ložiskách v šasi prostředku, které je vyrobeno ze svařených hliníkových plátů. Vzhledem k provedení konstrukce je prostředek vhodný do všech typů terénů. [64]

Výrobce nabízí široké možnosti konfigurace komponent jako napájecí akumulátory, kola či motory (různé rychlosti otáčení). Rovněž lze zvolit možnost přidání RC kontroléru, díky kterému je možné prostředek manuálně řídit. [64]



Obrázek 2.9: Robot IG42-DB4 *(zdroj: převzato z [64])*

2.3.2 Akademické řešení

V článku [65] je popsán čtyřkolový KRP vyvinutý na *Kumoh National Institute of Technology*. Jedná se o manuálně řízený prostředek určený pro výzkumné účely. Je řízen Ackermanovým způsobem řízení s tím, že natočitelná jsou všechna čtyři kola (každé má vlastní servomotor). Poháněn je jedním DC motorem, z nějž je výkon přenášen diferenciálním převodem k jednotlivým kolům, díky čemuž se mohou při zatáčení otáčet různými rychlostmi.

Řízení prostředku provádí mikrokontrolér *Arduino Nano 33 IoT*. Ten ovládá motory pomocí PWM signálu a skrze ZigBee rozhraní komunikuje s ovladačem pro manuální řízení. Ovladač obsahuje dvouosý joystick, kterým se řídí rychlost a úhel zatáčení a čtyři programovatelná tlačítka.

Článek rovněž představuje jednoduchý adaptivní algoritmus pro manuální řízení. Jeho princip spočívá v dynamickém zapojování zadních kol do zatáčení prostředku v závislosti na velikosti úhlu zatáčení. Je-li požadovaný úhel zatočení vyšší než stanovené maximum, jsou do zatáčení zapojena jak přední tak zadní kola, je-li menší, zatáčí se pouze předními. Díky tomu dosahuje velmi dobré přesnosti a časové efektivity (není nutné couvání při vyšších úhlech).

Obrázek 2.10: Akademický robot z Kumoh National Institute of Technology *(zdroj: převzato z [65])*

2.3.3 Hobby řešení

Na webu *Robotika.cz* je v článku [66] popsán čtyřkolový diferenciálně řízený robot *Freyja*, který vytvořil Jiří Iša v rámci Cogito Teamu. Jedná se o KRP určený pro účast na soutěži *SubT Tunnel Circuit* pořádané agenturou DARPA. Je navržen pro autonomní řízení a proto obsahuje celou řadu senzorů. Na každé straně je osazen jednou stereo kamerou, na vrcholu jedním čtyřřádkovým všesměrovým LiDARem, dále dvěma Ardusimple simpleRTK3B GNSS moduly, RTK stanicí, modulem pro indoor pozicování UWB BU01, kombinovaným kompasem, gyroskopem a akcelerometrem CMPS14, hallovými sondami a dalšími podpůrnými senzory. Řízení provádí minipočítač Intel NUC.

Šasi prostředku je zhotoveno z překližky. K němu jsou pomocí úchytů vytištěných z PLA s příměsí uhlíku na 3D tiskárně připevněny osy motorů. Samotné DC motory FLD13S jsou integrovány do kol prostředku.

Obrázek 2.11: Robot Freyja *(zdroj: převzato z [66])*

2.3.4 Shrnutí analýzy

Analyzovaná řešení a jejich nejdůležitější aspekty jsou stručně představeny v tabulce 2.2. Z informací v ní obsažených vyplývá několik důležitých poznatků:

- Preferovaným typem podvozků je čtyřkolový podvozek s diferenciálním řízením. Je tomu tak patrně z důvodu dobré stability a rozložení váhy při využití čtyř kol, což je u prostředků s vysokým zatížením klíčové. Diferenciální řízení je preferováno zřejmě kvůli možnosti otáčení na místě a jednoduššímu návrhu robustních úchytů motorů a kol, jež jsou zde rovněž nutností. Vedle toho stojí za zmínku dva případy užití všesměrových (švédských) kol, u kterých je vzhledem ke složité konstrukci vyšší nosnost poměrně nákladná a jsou tedy běžnější u prostředků s nízkým zatížením.

- Mezi řídícími jednotkami jsou nejběžnější průmyslové a jednodeskové počítače. To je dáno především velmi příznivým poměrem mezi jejich cenou a výkonem. Vliv má jistě také zvyšování nároků na výkon nutný pro zpracování senzorických dat z čím dál pokročilejších senzorů. U hobby řešení pak převažují mikrokontroléry, patrně kvůli jejich nízké ceně a jednoduššímu programování. Za povšimnutí stojí také prostředky bez řídicí jednotky. Tento přístup je poměrně běžný u průmyslových řešení navržených jako *platforma*, kde výrobce počítá s tím, že řídicí subsystém navrhne sám uživatel.

- Senzorický subsystém často chybí, zvláště pak u průmyslových platforem. U hobby řešení dominují jednodušší senzorické soustavy, obvykle základní sonary,

Hallovy sondy a taktilní senzory. U akademických a kompletních průmyslových řešení dominují složité soustavy, obvykle obsahující stereokamery, LiDARy a GNSS moduly.

- Dominantní způsob řízení je manuální. Je tomu tak patrně kvůli jednoduché implementaci. Autonomní řízení není není příliš běžné, zvláště u průmyslových řešení. Obvykle se totiž počítá s tím, že řízení navrhne sám uživatel na míru požadované aplikaci.

Obecně lze říci, že mezi prostředky s vyšší nosností jsou dominantní průmyslová řešení a zvláště pak konfigurovatelné platformy. Mezi akademickými řešeními jsou nejčastější prostředky určené k testování algoritmů pro autonomní řízení, u kterých není nosnost prioritou. U hobby řešení dominují jednoduché a malé KRP, ale zvláště v posledních letech se začínají objevovat i prostředky s vyšší nosností. Tento trend je patrně způsoben stále dostupnějšími komponentami a také rostoucím počtem amatérských soutěží, kde se prostředky tohoto typu dobře uplatňují, jako např. [67] a [68].

	Oblast	Typ podvozku	Typ řídicí jednotky	Způsob řízení	Senzorický subsystém	Nosnost
IG42-DB4 [64]	Průmyslová	Čtyřkolový, diferenciálně řízený	-	Manuální (volitelné)	-	9 - 45 kg (podle konfigurace)
A012 [69]	Průmyslová	Čtyřkolový, s Ackermannovým řízením	-	-	Základní	50 kg
Hunter SE [70]	Průmyslová	Čtyřkolový, s Ackermannovým řízením	SBC/Průmyslový počítač	Manuální	-	50 kg
Scout 2.0 [71]	Průmyslová	Čtyřkolový, diferenciálně řízený	SBC/Průmyslový počítač	Manuální	-	50 kg
SUMMIT-XL STEEL [72]	Průmyslová	Čtyřkolový (všesměrový), diferenciálně řízený	SBC/Průmyslový počítač	-	Pokročilý	250 kg
4W MRP [65]	Akademická	Čtyřkolový, s Ackermannovým řízením	Mikrokontrolér	Manuální	-	Nespecifikovaná
CrunchBot [73]	Akademická	Tříkolový, diferenciálně řízený	Univerzální počítač	Autonomní	Pokročilý	Nespecifikovaná
Skladový KRP [74]	Akademická	Šestikolový (všesměrový), diferenciálně řízený	SBC/Průmyslový počítač	-	Pokročilý	150 kg
Freyja [66]	Hobby	Čtyřkolový, diferenciálně řízený	SBC/Průmyslový počítač	Autonomní	Pokročilý	Nespecifikovaná (uveze dospělého člověka)
6W MRP [75]	Hobby	Šestikolový, diferenciálně řízený	Mikrokontrolér	Manuální	Základní	$>$ 12 kg
Terénní 4W MRP [76]	Hobby	Čtyřkolový, diferenciálně řízený	Mikrokontrolér	Autonomní i manuální	Základní	Nespecifikovaná

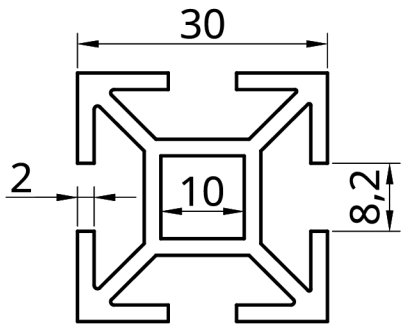
Tabulka 2.2: Porovnání vybraných KRP

3. Praktická část

3.1 Návrh a realizace konstrukce

3.1.1 Výběr materiálů

Jedním z požadavků práce bylo co největší využití materiálů dostupných v zásobách *laboratoře 3D tisku a aditivní výroby*, proto bylo rozhodnuto, že kostra prostředku bude vyrobena z duralových profilů systému Kombi 30 ×30 (odpovídajících normám *EN AW-6060*, *EN 573-3* a *EN 755-1,2,9*). Jedná se o extrudované profily o průřezu 30 ×30 mm s drážkou na každé straně, která umožňuje jednoduché spojování více kusů profilů dohromady. Vedle dostupnosti vedly k výběru i následující vlastnosti: poměrně nízká hmotnost (930 g na 1 m délky [77]), vysoká odolnost a snadná obrobitelnost pomocí standardních dílenských nástrojů.



Obrázek 3.1: Průřez Kombi profilu *(zdroj: vlastní)*

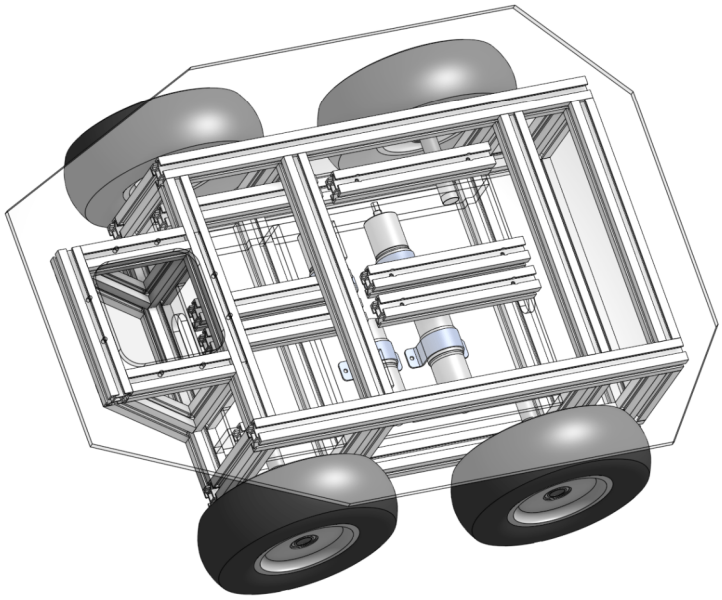
Hlavními požadavky na materiál pro výplně prostorů kostry, tedy boční stěny a dno prostředku, byly nízká hmotnost, odolnost proti povětrnostním vlivům a alespoň průměrná odolnost proti nárazu. Zvažována byla dřevotříska a polymethylmetakrylát (tzv. plexisklo). Zvolena byla nakonec možnost druhá, a to především kvůli dlouhodobě stabilní odolnosti proti povětrnostním vlivům. Konkrétně byly vybrány polymethylmetakrylátové pláty o tloušťce 5 mm, ze zásoby laboratoře. U vrchního plátu šasi byly požadavky podobné, ale přibyla zde nutnost odolat dlouhodobému tlaku, jelikož se počítá s umístěním dalších zařízení na jeho vrchní ploše. Proto byl zvolen polykarbonát se zvýšenou UV ochranou, který má mnohem větší odolnost a také pružnost než polymethylmetakrylát.

U drobnějších a pomocných dílů se od začátku počítalo primárně s výrobou pomocí 3D tisku. Vzhledem k tomu, že prostředek bude při používání vystaven UV záření, byl zvolen materiál PETG, který při jeho působení nedegraduje. Vedle toho dosahuje

poměrně dobré mechanické i teplotní odolnosti, díky čemuž je obecně vhodný pro mechanické díly. Konkrétně byly použity filamenty od české firmy Prusa Research a.s., jmenovitě *Prusament PETG Recycled* a *Prusament PETG Jet Black*. První zmíněný byl použit primárně, jelikož se jedná o 100% recyklovaný materiál, který má menší dopad na životní prostředí.

3.1.2 3D model

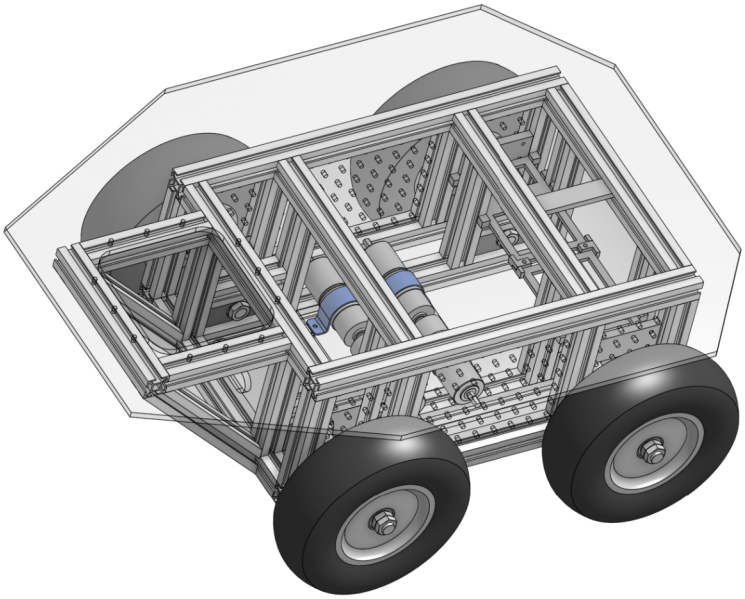
Po výběru materiálů následoval úvodní návrh konstrukce, provedený v programu Onshape. První verze návrhu počítala s řetězovými převody mezi motory a koly, vnitřním umístěním akumulátoru, prostorem pro akumulátory připojených zařízení a vnitřní platformou pro přídavné senzory (viz obrázek 3.2).



Obrázek 3.2: Úvodní návrh modelu *(zdroj: vlastní)*

Po revizi byla kvůli zbytečnému zvyšování váhy prostředku odebrána vnitřní platforma, čímž se ušetřilo přibližně 2,82 kg. Jako náhrada byly přidány otvory v bočních a čelních plátech prostředku, o rozměrech vyhovujících šroubům s metrickým závitem M5. Přídavné senzory lze buď připevnit přímo k otvorům (jako je tomu u čelního LiDARu), nebo k závitovým tyčím jež lze do otvorů umístit.

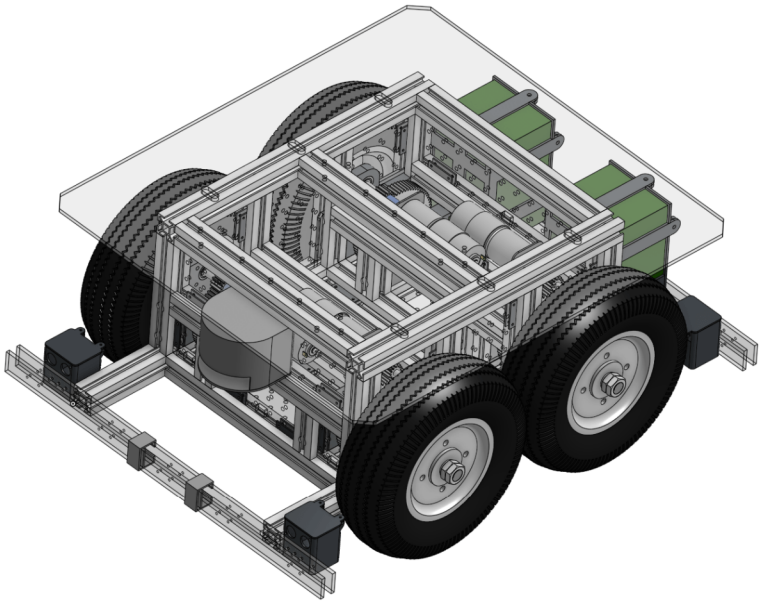
Druhá revize vyústila v přehodnocení proveditelnosti řetězových převodů a jejich následné nahrazení ozubenými převody. Rovněž došlo k odebrání prostoru pro uložení akumulátorů přídavných zařízení, a to opět z důvodu snížení hmotnosti a též zjednodušení konstrukce. Dále též došlo k přemístění akumulátoru z vnitřku prostředku do vnějších klece, díky čemuž se značně usnadnilo jeho odebrání za účelem opětovného



Obrázek 3.3: Model po odebrání vnitřní platformy *(zdroj: vlastní)*

nabití. Poslední změnou byl přechod na strukturovanou organizaci komponent modelu. V předešlých iteracích nebyly komponenty nikterak organizovány, což s jejich přibývajícím počtem způsobilo, že se model stal příliš nepřehledným a i drobná úprava jedné komponenty často znamenala nutnost mnohočetných oprav okolních komponent. Proto bylo rozhodnuto přetvořit model od začátku tak, aby se podobným problémům dále předešlo.

Po poslední revizi došlo ke zkrácení šasi za účelem snížení vzdálenosti mezi nápravami, čímž se zjednodušilo zatáčení (pohon je diferenciální). Rovněž byla přidána ložiska pro uložení motorových hřídelí ve vnější stěně prostředku, což zvýšilo celkovou robustnost soustavy. Finální návrh se tedy skládá z duralové kostry doplněné o děrované bočnice a dno z polymethylmetakrylátu, jež jsou usazeny v drážkách profilů pomocí rozpěrek z PETG. Jednotlivé převodové soustavy jsou usazeny v protilehlých blocích rovněž z PETG, ke kterým jsou pomocí ocelových plátů přichyceny motory. Vrchní plocha je kryta polykarbonátovou deskou a stejně tak i kontaktní senzory na na čelech prostředku. Ke konstrukci jsou připevněny senzory v krytech z PETG a v zadní části klec na akumulátor z téhož materiálu.



Obrázek 3.4: Finální návrh modelu *(zdroj: vlastní)*

3.1.3 Pohonná soustava

Pohon prostředku byl vzhledem k četnosti využití u analyzovaných řešení zvolen diferenciální se čtyřmi samostatně poháněnými koly. Vybrána byla nafukovací kola o průměru 260 mm s kovovým diskem a nosností 150 kg. Problémem bylo, že tato kola byla k dispozici pouze s nábojem osazeným kuličkovými ložisky, a tedy jejich výrobcem zamýšlené použití bylo pro volné otáčení kolem hřídele, nikoliv spolu s ní, jak vyžadovalo zamýšlené uchycení. Proto byly vymodelovány a z PETG vytištěny krytky pro nasazení na oba konce náboje (viz obrázek 3.5), ke kterým je možno hřídel přichytit a ložiska tak ignorovat.

Motory byly vybrány ze zásob laboratoře, a to PG420-24-49-BE. Jedná se o stejnosměrné komutátorové motory s uhlíkovými kartáči, vnitřní planetovou převodovkou s kovovými ozubenými koly, hřídelí uloženou v kuličkovém ložisku a dvěma Hallovými sondami pro měření otáček. Jejich parametry jsou popsány v tabulce 3.1.

Parametr	**Hodnota**
Napájecí napětí	24 V
Jmenovitá rychlost	122 $ot \cdot min^{-1}$
Jmenovitý moment	16 $kg \cdot cm$

Tabulka 3.1: Parametry motorů PG420-24-49-BE

Vzhledem k tomu, že jedním z cílů práce je vyšší nosnost, bylo rozhodnuto, že

Obrázek 3.5: Kryty náboje *(zdroj: vlastní)*

bude pomocí převodovky snížena provozní rychlost prostředku, ve prospěch zvýšení provozního momentu pohonné soustavy. V rámci zachování jednoduchosti konstrukce byl zvolen ozubený převod se dvěma koly. Orientační výpočet převodového poměru i byl inspirován výpočtem v [78] a je popsán dále:

$$i = \frac{M_C}{\eta_P M_M n}, \tag{3.1}$$

kde M_C je celkový potřebný moment pohonu, η_P účinnost převodovky, M_M jmenovitý moment jednoho motoru a n počet motorů v celém pohonu. M_C lze spočítat následovně:

$$M_C = F_C r, \tag{3.2}$$

kde r je poloměr hnaného kola a F_C celková síla, kterou musí pohon překonat, vypočítaná následovně:

$$F_C = (F_f + F_W + F_a) \cdot \theta \tag{3.3}$$

$$F_f = \mu m g \cos \alpha \tag{3.4}$$

$$F_W = m g \sin \alpha \tag{3.5}$$

$$F_a = m a, \tag{3.6}$$

kde F_f reprezentuje třecí odpory, F_W sílu potřebnou pro překonání výškového rozdílu, F_a odpor zrychlení způsobený setrvačností posuvných hmot a θ součinitel rotačních odporů. Dále μ značí trakční součinitel tření, m hmotnost prostředku s nákladem,

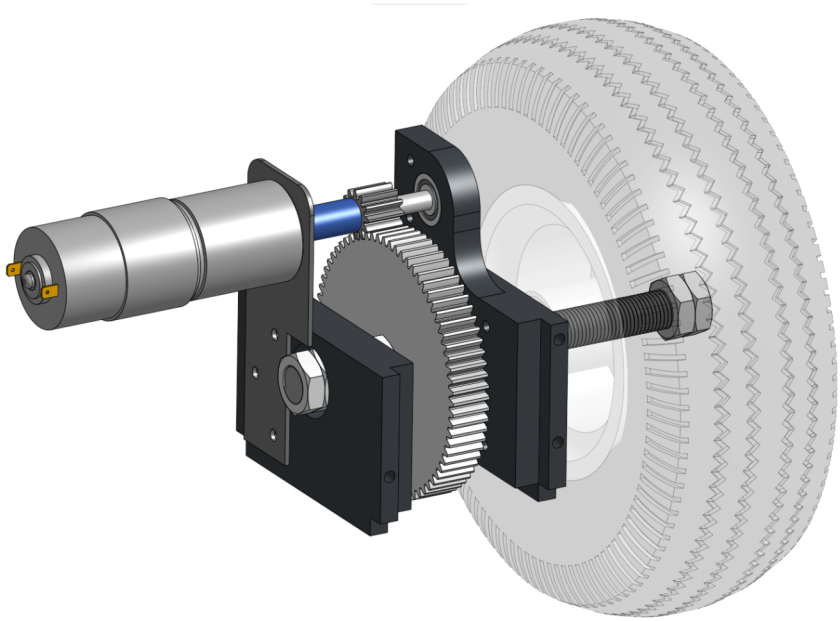
g tíhové zrychlení, α úhel sklonu nakloněné roviny pod prostředkem a a požadované zrychlení prostředku.

Tyto vzorce byly použity k několika výpočtům s různými parametry (viz tabulka 3.2), na základě kterých byla z dostupné nabídky vybrána kola s převodovým poměrem 6:1. Jedná se o ozubená kola s modulem 1,5 od firmy *matis s.r.o* s 12 a 72 zuby.

m $[kg]$	a $[m \cdot s^{-2}]$	r $[m]$	M_M $[kg \cdot cm]$	η_P [%]	α [°]	μ [–]	n [–]	θ [–]	**poměr** [–]
70	0,2	0,13	16	98	**0**	0,3	4	1,3	6,05
55	0,2	0,13	16	98	**5**	0,3	4	1,3	6,03
45	0,2	0,13	16	98	**10**	0,3	4	1,3	5,94

Tabulka 3.2: Výpočty převodového poměru (tučně jsou zvýrazněny měněné parametry); hodnoty μ a θ jsou převzaty z [78] a η_P z [79]

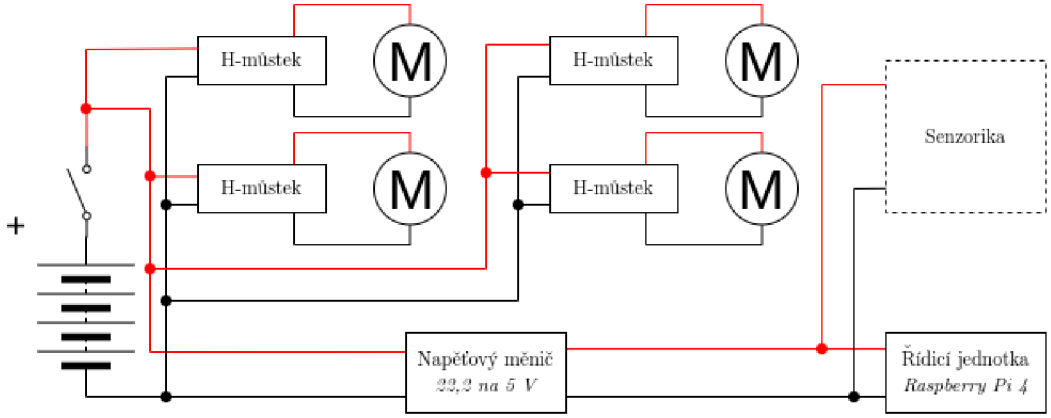
Ozubená kola jsou maticemi a pružnými podložkami připevněna k závitovým tyčím, jež jsou uloženy v kuličkových ložiskách. Ta jsou zasazena do bloků vytištěných z PETG, které jsou umístěny v kostře a připevněny vodorovnou závitovou tyčí, jež prochází skrz celý blok. Motor je k bloku připevněn pomocí ocelové vložky a přidružená hřídel je na osu motoru připevněna spojkou.



Obrázek 3.6: Model pohonu jednoho kola *(zdroj: vlastní)*

3.1.4 Elektronika

Celý prostředek je napájen lithium-polymerovým akumulátorem *Gens Ace Tattu Plus* s nominálním napětím 22,2 V a kapacitou 22 Ah. Z něj vede kladná větev skrz hlavní vypínač do instalačních svorek WAGO, odkud je rozvedena do napájecích svorkovnic H-můstků a také do napěťového měniče XL4016E1, který snižuje napájecí napětí na 5,3 V pro napájení řídicí jednotky a senzorů. Záporná větev je rozvedena obdobně, s tím rozdílem, že je vynechán vypínač. Výstup napěťového měniče je jakožto napájecí přiveden přímo na *5 V* a *GND* piny řídicí jednotky. Zapojení GPIO pinů řídicí jednotky je popsáno v kapitole 3.3.2 a zapojení senzorů v podkapitolách kapitoly 3.2. Jejich vzájemné propojení je realizováno kabely se čtyřmi žilami, jež jsou na koncích osazeny konektory Molex KK-254.

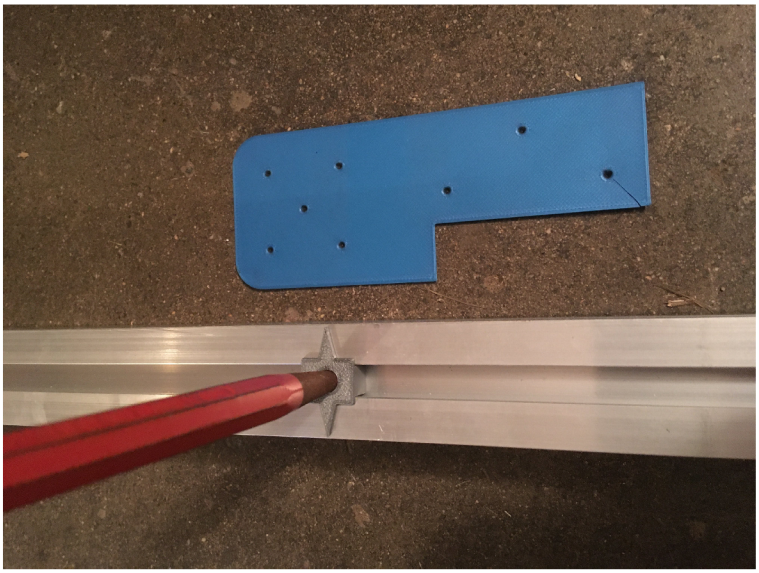


Obrázek 3.7: Diagram napájení komponent *(zdroj: vlastní)*

3.1.5 Postup konstrukce

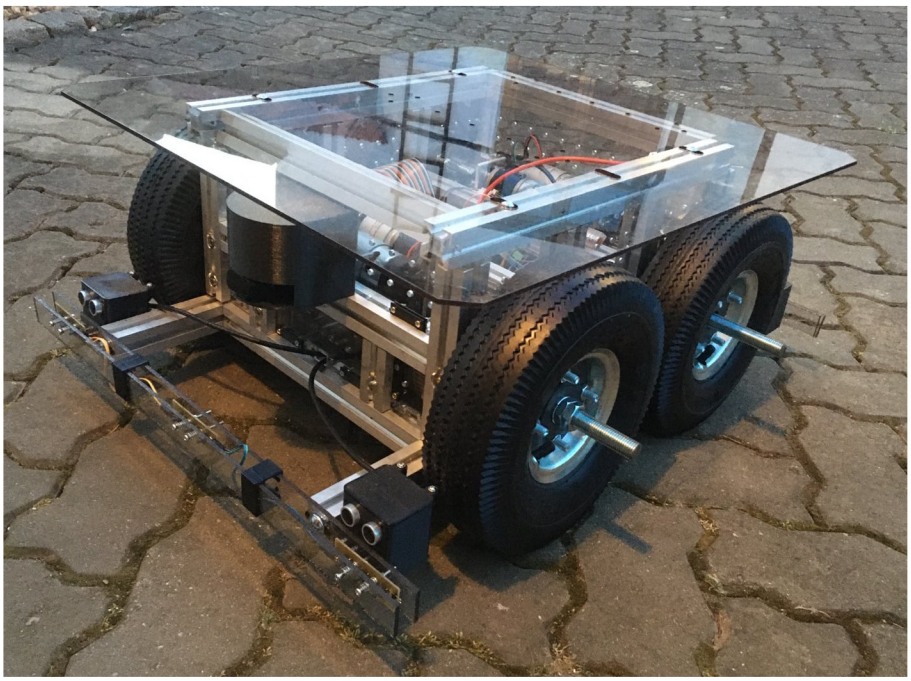
Konstruování prototypu začalo nařezáním profilů na požadované délky a vyvrtáním otvorů pro přichycení šroubů. Pro přesné určení místa vrtání byla vymodelována a vytištěna šablona (viz obrázek 3.8). Následně byly na laserovém plotru vyřezány polymethylmetakrylátové a polykarbonátové díly. Dále byly vytištěny bloky pro uchycení hřídelí a navrtány vložky pro upevnění motorů. Pro ně byla rovněž vytvořena šablona (viz obrázek 3.8). Následovalo složení pohonných soustav a sestavení okolní části kostry. Jednotlivé profily byly vzhledem k vysoké ceně originálních spojovacích dílů (180 Kč bez DPH za jeden kus) spojeny pomocí šroubů a mosazných hmoždinek s metrickým závitem M8 umístěných ve vnitřní dutině profilu (viz obrázek 3.9); tento postup je převzat z práce [80]. Následovalo složení dna prostředku a připevnění náprav k němu. Poté byla zapojena a složena vnitřní elektronika. Následně byly složeny vnější senzory, jejichž přívodní kabely jsou do prostředku přivedeny skrz otvory vyříznuté v

bočnicích. Nakonec následovalo upevnění klece s akumulátorem a horní desky šasi. Ta má v sobě vyříznuté otvory pro uchycení dalšího zařízení a vespod je podpořena posuvným profilem, který lze umístit podle potřeby pod místa úchytu.



Obrázek 3.8: Šablony pro přípravu vrtání *(zdroj: vlastní)*

Obrázek 3.9: Uchycení hmoždinek v profilu *(zdroj: vlastní)*



Obrázek 3.10: Výsledná konstrukce *(zdroj: vlastní)*

3.2 Senzorický subsystém

Senzorika je velmi důležitou částí návrhu robotického prostředku, jelikož výběr vhodných senzorů má značný vliv na možnosti řízení prostředku. Vzhledem k požadavkům práce byly vybrány levné senzory s jednodušší obsluhou. Některé z nich byly dále doplněny o přídavný mikrokontrolér, který řídí jejich činnost a umožňuje připojení senzoru ke společné TWI sběrnici. Výčet zvolených senzorů je následující (detailněji jsou popsány v následujících podkapitolách):

- *1×LiDAR* pro snímání prostoru před prostředkem
- *4×soustava sonarů*, kde jeden snímá prostor před a druhý pod soustavou; umístěny jsou v rozích prostředku
- *2×kontaktní senzor (nárazník)* vpředu a vzadu
- *4×Hallova sonda* k počítání otáček jednotlivých motorů

3.2.1 Čelní LiDAR

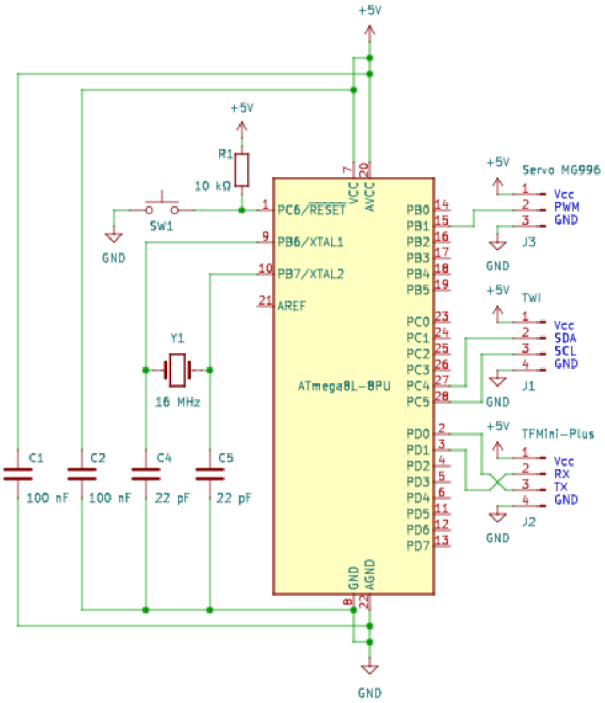
Jako primární zdroj informací o okolí prostředku byl zvolen *LiDAR TFMini-Plus*. Jedná se o single-point LiDAR od společnosti Benewake, schopný měření vzdálenosti v rozsahu 0,1 až 12 metrů s přesností 1 %. Varianta LiDARu, dostupná při návrhu prostředku, využívá ke komunikaci rozhraní UART a tedy bylo nutné přidat mezičlánek pro překlad mezi UART a TWI. K tomu byl zvolen mikrokontrolér *ATmega8* od společnosti Microchip Technology. Jedná se o variantu mikrokontroléru ATmega328 (popsaného v kapitole 2.1.2.1) s menším množstvím paměti. Senzor je umístěn na čele prostředku a snímá prostor před ním.

Obrázek 3.11: LiDAR TFMini-Plus *(zdroj: převzato z [81])*

Vzhledem k tomu, že je LiDAR jednobodový, a má tedy velmi malý zorný úhel (pouze 3,6° [82]), bylo nutné jej doplnit o aktuátor, který zorný úhel rozšíří. K tomuto účelu byl zvolen *servo motor MG996* s maximálním úhlem natočení 180°. Řízen je rovněž výše zmíněným mikrokontrolérem, jež omezuje úhel natočení na 60°.

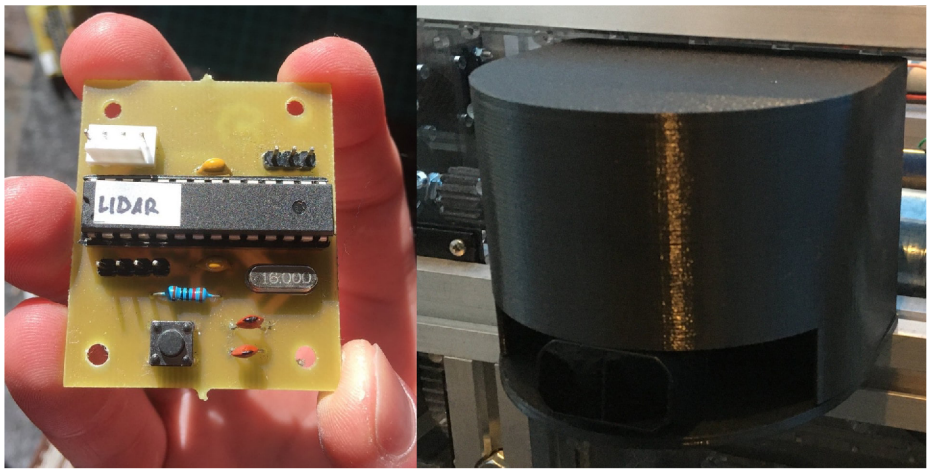
3.2.1.1 Zapojení a konstrukce

Jádrem soustavy je výše popsaný mikrokontrolér, jehož zapojení je shodné se zapojením mikrokontroléru SCU, popsaným v podkapitole 3.3.1.1. Z mikrokontroléru jsou dále vyvedeny tři konektory, a to pro připojení LiDARu, serva a připojení na TWI sběrnici spolu s napájením obvodu. V prvním z nich jsou vyvedeny piny Rx, Tx, +5 V a GND, což odpovídá vývodům LiDARu. Druhý zmíněný konektor obsahuje opět napájecí piny (+5 V a GND) a také PWM pin pro ovládání natočení serva. Poslední konektor též obsahuje napájecí piny, ty zde však neslouží k napájení dalšího zařízení, nýbrž k napájení soustavy samotné. Kromě těchto pinů obsahuje také datové SDA a SCL pro připojení na TWI sběrnici prostředku, která je spolu s napájením vyvedena z SCU.



Obrázek 3.12: Schéma zapojení LiDARu *(zdroj: vlastní)*

Podle schématu 3.12 byla v programu KiCad navržena a následně pomocí frézky vyrobena deska plošných spojů. Vzhledem ke snaze o co nejsnazší replikaci byla navržena jako jednovrstvá a komponenty pro osazení zvoleny v provedení THT.



Obrázek 3.13: Deska plošných spojů a pouzdro LiDARu *(zdroj: vlastní)*

Všechny komponenty jsou uloženy ve dvoudílném pouzdře z PETG, vytištěném na 3D tiskárně. To bylo vymodelováno v programu Onshape. Pouzdro má na čelní straně okénko, ve kterém se LiDAR pohybuje. K servo motoru je připojen pomocí vytištěného ramena, na kterém se otáčí.

3.2.1.2 Program

Mikrokontrolér je programován v jazyce C++ na platformě Arduino. Využívá knihovny `Wire` (komunikace přes TWI), `Servo` (řízení servo motoru), `TFMPlus` (čtení dat z LiDARu) a `Protorduino` (viz podkapitolu 3.3.1). Obsahuje dvě *Protorduino* úlohy:

- *pohybová úloha* slouží k periodickému natáčení serva v úhlu 0-60°
- *měřící úloha* čte naměřené hodnoty z LiDARu a odesílá je do SCU

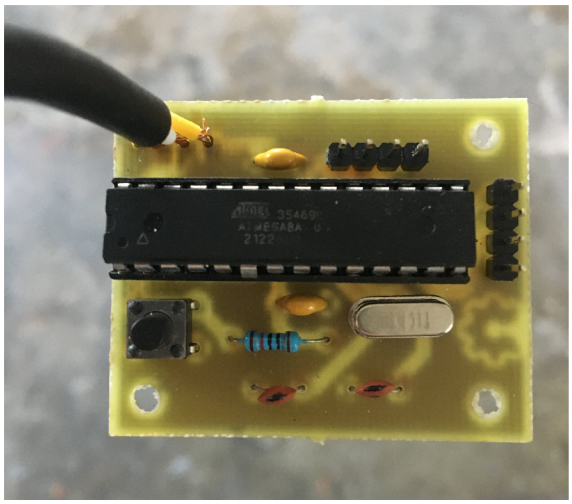
3.2.2 Soustavy sonarů

V každém rohu prostředku je umístěna jedna sonarová soustava. Ta se skládá ze dvou sonarů HY-SR05, které jsou schopny měřit vzdálenost v rozsahu od 2 do 450 cm, s přesností na 3 mm. Jeden ze sonarů míří vpřed, tedy snímá prostor před (potažmo za) prostředkem a druhý dolů, čímž detekuje díry před koly prostředku. Sonary mají nestandardní komunikační rozhraní, skládající se z pinu pro vyslání ultrazvukového pulzu (*Trigger*) a pinu, na kterém je indikováno přijetí odrazu pulzu (*Echo*). Proto byly, podobně jako LiDAR, doplněny o mikrokontrolér ATmega8, který řídí měření a na požádání odesílá naměřená data přes TWI sběrnici.

3.2.2.1 Zapojení a konstrukce

Mikrokontrolér a přídavné komponenty jsou zapojeny stejně jako v případě LiDARu. Z mikrokontroléru jsou vyvedeny tři konektory, z nichž dva jsou čtyřpinové DuPont M20 lišty pro připojení sonarů pomocí DuPont propojovacích kabelů. Třetí byl původně Molex KK-254 konektor, pro připojení k napájení a na TWI sběrnici. Ten byl však při testování nahrazen přímým napájením vodičů do DPS, a to z důvodu problémů s kontaktem při ostrém ohybu kabelu v pouzdře senzoru. Druhá strana kabelu zůstala osazena konektorem, nevznikl tedy problém s odpojením senzoru.

Podobně jako u LiDARu byla podle schématu navržena v programu KiCad jednovrstvá deska plošných spojů, která byla následně pomocí frézky LPKF E44 čtyřikrát vyfrézována. Komponenty pro osazení byly opět zvoleny v provedení THT.



Obrázek 3.14: Deska plošných spojů sonarové soustavy *(zdroj: vlastní)*

Soustavy jsou uloženy ve dvoudílném pouzdře z PETG, vymodelovaném v programu Onshape a vytištěném na 3D tiskárně.

3.2.2.2 Program

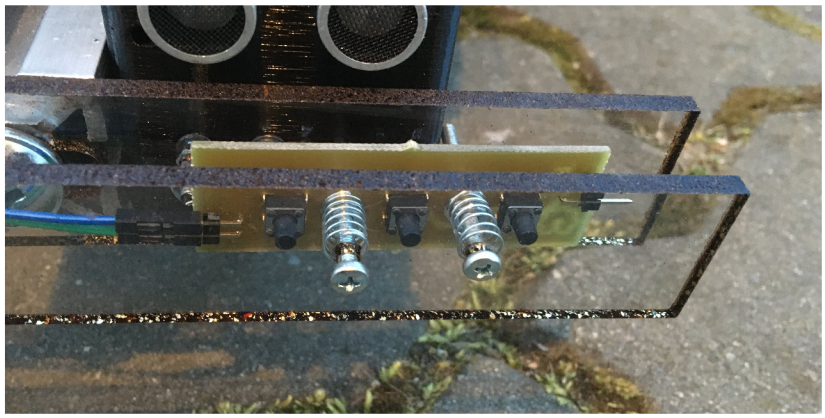
Mikrokontrolér je programován stejným způsobem jako u LiDARu. Program využívá knihovny `Wire` (komunikace přes TWI) a `NewPing` (čtení dat ze sonarů). Jeho činnost spočívá v periodickém čtení a ukládání dat ze sonarů, která jsou na požádání odeslána přes TWI. Jedna čtecí perioda trvá 60 ms s tím, že mezi čtením z prvního a druhého sonaru je prodleva 30 ms (tedy průběh periody je následující: čtení z prvního sonaru, pauza 30 ms, čtení z druhého sonaru, pauza 30 ms). Tím je zajištěno, že se sonary navzájem neovlivňují.

Obrázek 3.15: Pouzdro sonarové soustavy *(zdroj: vlastní)*

3.2.3 Kontaktní senzory

Jako záložní řešení pro případ selhání LiDARu a sonarů byly přidány dva kontaktní senzory. Jsou umístěny na přední a zadní straně prostředku před všemi jeho komponentami. Skládají se z modulů tvořených deskou plošných spojů osazenou třemi tlačítky, která při stisknutí spojí napájecí linku s linkou, připojenou na přerušovací vstup řídicí jednotky, čímž předá informaci o kontaktu a řídicí program okamžitě zastaví všechny motory. Desky plošných spojů jsou navzájem propojeny (tři v každém senzoru) a umístěny mezi dva pláty z polykarbonátu. Ty jsou dostatečně pružné pro přenos nárazu k tlačítkům. Pro zajištění návratu do původní polohy jsou doplněny o pružiny.

Podobně jako u předešlých senzorů, deska plošných spojů byla navržena v programu KiCad jako jednovrstvá a poté šestkrát vyfrézována na frézce LPKF E44.



Obrázek 3.16: Detail DPS kontaktního senzoru *(zdroj: vlastní)*

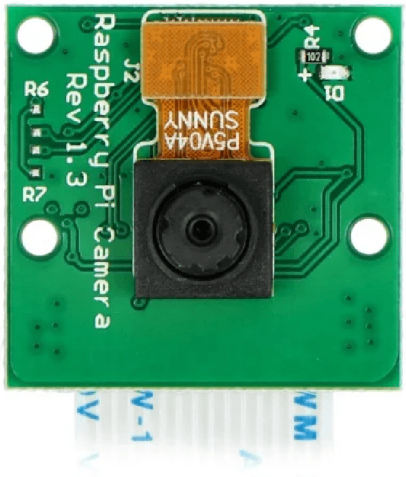
3.2.4 Hallovy sondy

Každý z použitých motorů obsahuje dvě Hallovy sondy, vzájemně pootočené o 90°, pro měření otáček. Vzhledem k zamýšlenému použití v procesu odometrie je použita pouze jedna z nich, jelikož druhá nepřináší v tomto směru žádné informace navíc. Princip činnosti sondy spočívá ve vyslání impulzu při provedení částečné otočky osy, v tomto případě $1/5$ otáčky, z čehož lze dopočítat ujetou vzdálenost za jednotku času.

Zapojení výstupních linek sond je provedeno přímo na vstupní piny řídicí jednotky, které jsou nastaveny na detekci přerušení, a tedy vždy při přijetí impulzu může řídicí program reagovat odesláním příslušné MIB zprávy.

3.2.5 Kamera pro sledování linky

Na dno šasi prostředku byla přidána kamera OdSeven OV5647 s rozlišením 5 Mpx. Ta je k řídicí jednotce připojena pomocí dedikovaného CSI rozhraní. K šasi je připevněna pomocí krytu vytištěného z PETG na 3D tiskárně. Softwarová podpora je zajištěna pomocí knihovny *libcamera*, skrz kterou s ní komunikuje knihovna Pi4J.



Obrázek 3.17: Kamera pro sledování linky *(zdroj: [83])*

3.2.6 GPS modul a kompas

Původně měl prostředek obsahovat GPS modul *NEO-6M* a kompas *HMC5883L*, jakožto záložní zdroje dat pro lokalizaci prostředku. Při testování se však ukázalo, že údaje z GPS modulu jsou natolik nepřesné, že nepřinášejí prakticky žádnou hodnotu a tak tato zařízení nakonec přidána nebyla.

3.3 Řídicí subsystém

Návrh řídicího subsystému byl původně zaměřen na využití průmyslového počítače POC-200 jakožto hlavní řídicí jednotky, jež bude vykonávat řídicí program. Ta měla být, vzhledem k níže popsaným nedostatkům rozhraní, doplněna o sekundární řídicí jednotku, jejímž úkolem byla komunikace s periferiemi prostředku.

Při implementaci se však objevily problémy v komunikaci mezi těmito jednotkami, jež vyústily v rozhodnutí nahradit obě jednotky jedinou, která bude schopna plnit úkoly obou dvou zároveň. K tomu účelu byl zvolen jednodeskový počítač Raspberry Pi 4, jež sice nedisponuje odolností na průmyslové úrovni, ale zato obsahuje vhodná rozhraní v dostatečné počtu a též dosahuje výrazně nižší spotřeby.

Následuje detailní popis obou řešení, včetně popisu výše zmíněných problémů. Dále je v této kapitole popsána aplikace, jež je na řídicí jednotce provozována.

3.3.1 Původní řešení

Jak již bylo řečeno, jako jádro řídicího subsystému byl původně určen průmyslový počítač POC-200 od výrobce Neousys Technology (viz kapitolu 1.4). Ten disponuje čtyřmi USB porty, čtyřmi sériovými (UART) porty a jedním DIO portem se čtyřmi vstupními a výstupními kanály. Vzhledem k poměrně nízkému rozšíření USB rozhraní u běžně dostupných senzorů a nízkému počtu možných připojených zařízení u rozhraní UART (obvykle se jedná o komunikaci jeden na jednoho), nejsou tato rozhraní vzhledem k požadavkům prostředku příliš vhodná. DIO port i přes svou univerzálnost rovněž nepředstavuje vhodné řešení, přetrvává zde totiž problém s nízkým počtem možných připojených zařízení, který je navíc doplněn o nestandardní softwarovou podporu.

Jako nejpraktičtější východisko bylo zvoleno přidání druhé řídicí jednotky (SCU), jejímž účelem je pouze komunikace se senzorickým a pohonným subsystémem. K tomu byl zvolen mikrokontrolér ATmega328P-PU od společnosti Microchip Technology. Jedná se o 8bitový mikrokontrolér s architekturou RISC, který patří do rodiny procesorů AVR. Disponuje 32 KiB programové flash paměti, 2 KiB SRAM a 1 KiB EEPROM. Z hlediska rozhraní obsahuje 23 digitálních vstupně-výstupních pinů (z toho 6 s A/D převodníkem, což znamená, že může pracovat jak v digitálním, tak analogovém režimu), sběrnici TWI, UART a SPI. Jeho velkou výhodou je, že se používá ve známých vývojových deskách Arduino UNO [84], díky čemuž má širokou podporu jak po stránce softwaru, tak informačních zdrojů.

K připojování senzorů byla zvolena sběrnice TWI, která nabízí dostatečné množství prostoru jak pro současné, tak i v budoucnu přidané senzory. Rovněž nabízí poměrně jednoduchou obsluhu pomocí nativní knihovny Wire a je často používána u běžně

dostupných senzorů. Jedinými výjimkami jsou kontaktní senzory (nárazníky) a Hallovy sondy, které jsou připojeny přímo na kontrolér, jako zdroje přerušení. U kontaktních senzorů byla univerzálnost obětována ve prospěch rychlosti reakce, jelikož ta je v situaci přímého kontaktu s překážkou klíčová. U Hallových sond bylo takto rozhodnuto ve prospěch jednoduchosti systému, jelikož sondy jsou přímou součástí použitých motorů a tedy jejich odebrání není předpokládáno.

Řízení motorů probíhá také přímo, připojením řídicích prvků na PWM piny kontroléru. Detailněji je toto připojení popsáno v podkapitole 3.3.1.1.

Mikrokontrolér SCU užívá Arduino bootloader, a tedy jeho řídicí program je standardní Arduino program. Jeho jádrem je knihovna *Protorduino*, jež byla naprogramována pro užití v SCU. Knihovna slouží k simulaci multithredingu na jednojádrovém procesoru a je inspirována konceptem protothreadů [85]. Princip činnosti knihovny spočívá v periodickém vykonávání definovaných úkolů, vždy po uplynutí zadané doby tak, že se navzájem nikterak neovlivňují. Program toho využívá k periodickému čtení dat ze senzorů, která jsou následně odesílána do hlavní jednotky. Čtení probíhá skrze TWI rozhraní, za pomoci knihovny Wire. Úlohy pro jednotlivé senzory jsou definovány dynamicky, tedy na základě údajů z hlavní řídicí jednotky, díky čemuž není nutné při přidání či odebrání senzoru zasahovat do kódu SCU.

Vedle Protorduino úloh jsou součástí programu dvě rutiny přerušení. První, využívající hardwarové přerušení, obsluhuje kontaktní senzory. Při detekci kontaktu dochází k okamžitému zastavení motorů a informování hlavní jednotky. Druhá rutina využívá pro nedostatek hardwarových přerušení softwarová a slouží k shromažďování pulzů z Hallových sond. Ty se ukládají do proměnné a výsledek se periodicky odesílá do hlavní jednotky. Posledním úkolem programu je na základě příkazů z hlavní řídicí jednotky nastavovat směr a rychlost otáčení motorů. Jakým způsobem tato komunikace mezi jednotkami probíhá je popsáno dále, v podkapitole 3.3.1.2.

3.3.1.1 Zapojení řídicí jednotky subsystémů

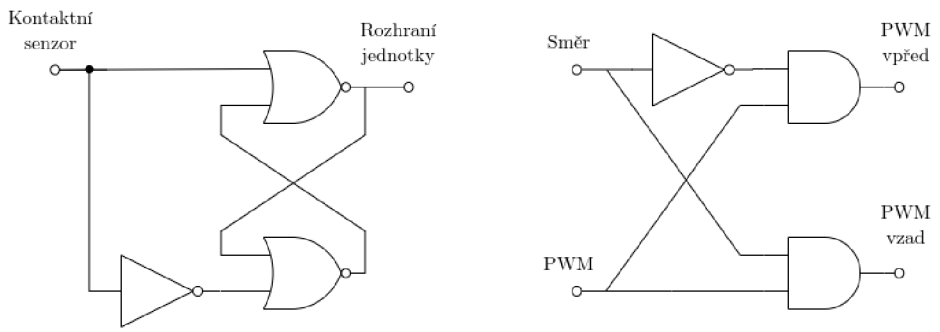
Schéma zapojení a z něj vycházející deska plošných spojů byly navrženy v programu KiCad. Vzhledem k poměrně velkému počtu komponent a propojů bylo zvoleno hierarchické schéma, kde jsou dílčí skupiny komponent umístěny na zvláštních listech a poté v hlavním listu referencovány pomocí vývodů. Návrh je díky tomu přehlednější a lépe škálovatelný.

Centrálním prvkem schématu je mikrokontrolér ATmega328P-PU, jež je doplněn o piezoelektrický krystal s rezonanční frekvencí 16 MHz, sloužící spolu se dvěma keramickými 22 pF kondenzátory, jako externí oscilátor. Dále se zde nachází spínač pro restartování kontroléru, doplněný o pull-up rezistor s hodnotou 10 $k\Omega$ a také dva

keramické blokovací kondenzátory s hodnotou 100 nF.

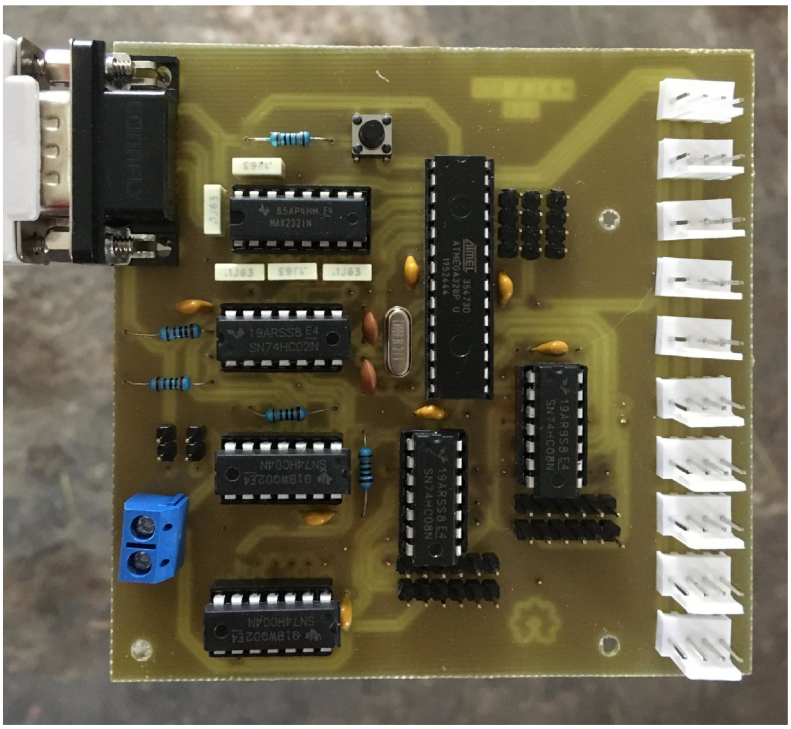
Z mikrokontroléru je vyvedena sběrnice TWI, která jej spojuje s deseti konektory Molex KK-254 pro připojení senzorů. Dále sběrnice UART, vedoucí skrz integrovaný obvod MAX232 do D-sub konektoru pro připojení k hlavní řídicí jednotce. Obvod MAX232 slouží k transformaci RS-232 signálů (z hlavní řídicí jednotky) na TTL (pro ATmega328P-PU) a naopak. Vedle sběrnic je vyvedeno též šest GPIO vstupních pinů jako zdroje přerušení programu. Dva z nich mají hardwarovou podporu přerušení a jsou využity k připojení předního a zadního kontaktního senzoru. Zbylé čtyři mají softwarovou podporu přerušení a slouží k příjmu dat z Hallových sond. Dalších osm GPIO pinů je využito jako výstupní, pro připojení k H-můstkům motorů. Jsou rozděleny do dvojic, z nichž každá obsahuje jeden PWM pin určující rychlost otáčení motoru a jeden obyčejný pin pro určení směru otáčení.

Vstupní piny pro kontaktní senzory jsou připojeny každý přes jeden bistabilní klopný obvod, jež zajišťuje tzv. debounce, čili odstranění zákmitů při sepnutí tlačítek v senzoru. Tyto BKO jsou realizovány pomocí dvojice NOR hradel (integrovaný obvod 74HC02) a jednoho invertoru (integrovaný obvod 74HC04). Výstupní piny pro H-můstky jsou připojeny přes demultiplexor, jež na základě signálu ze směrového pinu přepíná signál z PWM pinu na odpovídající vstup H-můstku. Demultiplexor je realizován pomocí dvou AND hradel (integrovaný obvod 74HC08) a jednoho invertoru (opět integrovaný obvod 74HC02). Všechny dříve zmíněné integrované obvody jsou podobně jako hlavní mikropočítač doplněny o blokovací kondenzátory.



Obrázek 3.18: Schéma zapojení klopného obvodu (vlevo) a demultiplexoru (vpravo) *(zdroj: vlastní)*

Na základě schématu byla navržena a na frézce LPKF E44 vyfrézována deska plošných spojů. Pro snazší výrobu byla navržena jako jednovrstvá s tím, že propoje, jež nelze na jedné vrstvě realizovat, jsou nahrazeny drátovými propoji.



Obrázek 3.19: Deska plošných spojů řídicí jednotky subsystémů *(zdroj: vlastní)*

3.3.1.2 Komunikace mezi jednotkami

Pro propojení řídicích jednotek byla zvolena sběrnice UART. Hlavním důvodem pro to byla především snadná implementace na straně ATmega328P-PU, jež má přímou softwarovou podporu a z hlediska hardwaru vyžaduje pouze převod mezi RS-232 a TTL signály.

Dále byl navržen a implementován jednoduchý protokol, jež staví na UART komunikaci a rozšiřuje ji o vícebytové strukturované pakety, kontrolu integrity přenášených dat, ochranu před přetečením UART bufferu a možnost opakovaného odeslání paketu při chybě přenosu. Protokol byl nazván *Simple Reliable Serial Communication (SRSC)*, jeho definice je k dispozici zde [86] a referenční implementace zde [87] (Java) a zde [88] (C++ pro Arduino).

Princip funkce protokolu spočívá v navázání spojení a následné výměně paketů s předem definovaným typem. Typy mohou být registrovány před zahájením komunikace manuálně na obou stranách, případně po navázání spojení pomocí dedikovaných paketů (takto lze registrovat typ z jednoho zařízení na druhém). Poté lze mezi zařízeními libovolně posílat registrované pakety s daty či příkazy.

Protokol během komunikace kontroluje integritu přenášených dat. Slouží k tomu kontrolní součet, jež je součástí všech paketů (vyjma potvrzovacích, viz tabulka 3.3). Při obdržení paketu je prováděna kontrola a neodpovídá-li obsah paketu kontrolnímu

součtu, paket je zahozen (náhrada poškozeného paketu je řešena níže, v sekci Struktura paketu). Protokol dále provádí prevenci zaplnění bufferu sériového rozhraní. Využívá k tomu čítač o velikosti bufferu, jehož hodnota je snižována při odeslání datového paketu o jeho velikost a naopak zvyšována při přijetí potvrzovacího paketu. Klesne-li hodnota čítače pod velikost aktuálního paketu, je odesílání pozdrženo, dokud není hodnota čítače opět zvýšena.

Struktura paketu je znázorněna na obrázku 3.20. Význam jednotlivých polí je následující:

- *ID* slouží k identifikaci vícenásobného přijetí `CRITICAL` paketu a odlišení sérií `REPEATED` paketů.
- *Typ* určuje o jaký paket se jedná. Jsou s ním spojeny následující vlastnosti:
  - *Závažnost paketu* udává, do jaké míry je zajištěno jeho doručení:
    - `NORMAL` není v případě poškození doručován znovu.
    - `REPEATED` je pro zvýšení pravděpodobnosti úspěšného doručení odeslán 3×.
    - `CRITICAL` je periodicky odesílán, dokud odesílatel neobdrží potvrzení o úspěšném doručení (tato metoda se nehodí pro použití na mikrokontrolérech, kvůli vyšší paměťové náročnosti). Implementace musejí podporovat přijímání, nikoliv však odesílání (např. Arduino implementace z výše popsaného důvodu odesílání nepodporuje).
  - *Délka dat*, udávaná v bytech.
- *Kontrolní součet* slouží ke kontrole integrity paketu. Je počítán metodou jedničkového doplňku.
- *Data* paketu. Byty jsou řazeny podle konvence little-endian.

ID *1 b*	Identifikátor typu *7 b*	Kontrolní součet *1 B*	Data *0-n B*

Obrázek 3.20: Struktura paketu *(zdroj: vlastní)*

Protokol sám definuje pět typů paketů, které slouží pro jeho interní činnost. Jejich popis a vlastnosti jsou stručně uvedeny v tabulce 3.3.

Název	Identifikátor	Kontrolní součet	Závažnost	Délka [B]	Popis
CONNECT	0x01	Ano	`NORMAL`	6	Slouží k navázání spojení a informování zařízení o velikosti bufferu.
CONNEFIRM	0x02	Ano	`NORMAL`	6	Je odesílán jako odpověď na *CONNECT*. Potvrzuje navázání spojení a rovněž předává informaci o velikosti bufferu.
ACCEPTACK	0x03	Ne	`NORMAL`	2	Potvrzení přijetí paketu závažnosti `NORMAL` nebo `REPEATED`. Nese počet přijatých bytů.
CRITACK	0x04	Ne	`NORMAL`	2	Potvrzení přijetí paketu závažnosti `CRITICAL`. Nese ID přijatého paketu.
REGTYPE	0x05	Ano	`REPEATED`	4	Registruje nový typ paketu. Přenáší identifikátor, závažnost a délku dat nového paketu.

Tabulka 3.3: Seznam rezervovaných typů paketů

Minimální rozhraní pro implementace je rovněž protokolem definováno. Každá implementace musí obsahovat definované metody, může je však přizpůsobovat vlastnostem cílové platformy či jazyka a doplňovat o dodatečné metody. Základní popis těchto metod obsahuje tabulka 3.4.

Využití protokolu v řídicích aplikacích jednotek je následující. Hlavní řídicí jednotka obsahuje modul, který využívá Java implementaci protokolu, odesílá skrze ní příkazy SCU a přijímá od ní data ze senzorů. Naproti tomu SCU využívá Arduino C++ implementaci jako stěžejní prvek celého programu, jelikož každá jeho činnost zahrnuje odesílání či přijímání SRSC zpráv.

Název	Návratový typ	Argumenty	Výjimky	Popis
Srsc	-	communicator: `object`	-	Konstruktor. *communicator* je reference na objekt reprezentující rozhraní využívané protokolem.
begin	`bool`	bufferSize: `u32`	-	Neblokující pokus o navázání spojení. *bufferSize* znamená velikost bufferu tohoto zařízení. Metoda vrací informaci o úspěchu navázání spojení.
begin	`void`	bufferSize: `u32` delay: `u32`	-	Blokující pokus o navázání spojení (pokusy se opakují, dokud není spojení navázáno). *bufferSize* má shodný význam jako v předešlém případě a *delay* určuje počet milisekund mezi jednotlivými pokusy.
registerPacketType	`void`	identifier: `u8` payloadSize: `u8` severity: `enum`	-	Registrace nového typu paketu. Argumenty odpovídají parametrům registrovaného typu.
registerPacket-TypeOnBoth	`void`	identifier: `u8` payloadSize: `u8` severity: `enum`	-	Registrace nového typu paketu na obou zařízeních. Argumenty mají shodný význam jako v předešlém případě.
writePacket	`void`	packetType: `u8`	UnknownPacket-TypeException, SerialBuffer-FullException, MissingPayload-Exception	Odeslání paketu bez dat. *packetType* je identifikátor typu paketu.
writePacket	`void`	packetType: `u8` payload: `u8[]`	UnknownPacket-TypeException, SerialBuffer-FullException	Odeslání paketu s daty. *packetType* má shodný význam jako v předešlém případě a *payload* reprezentuje data přenášená paketem.
registerOnPacket-ArrivedCallback	`void`	callback: `function`	-	Registruje funkci, jež se vykoná při přijetí paketu.

Tabulka 3.4: Metody SRSC rozhraní

3.3.1.3 Problémy původního řešení

V průběhu vývoje byly testovány jednotlivé komponenty jak po stránce hardwarové, tak softwarové, aby se při finálním zapojení minimalizovalo riziko chyby. I přes to se však objevil problém s navázáním UART spojení mezi jednotkami, který se nepodařilo odstranit a ani s jistotou odhalit jeho zdroj (pravděpodobně se však jedná o chybu v DPS jednotky). Proto bylo rozhodnuto nahradit obě jednotky jedinou, a tím se dalším komunikačním problémům vyhnout.

3.3.2 Výsledné řešení

Jak jíž bylo řečeno, jako náhradní řídicí jednotka byl zvolen jednodeskový počítač Raspberry Pi 4 model B. Jedná se o jednu generaci starší zařízení než je Raspberry Pi 5, popsané v podkapitole 2.1.2.2. Rozdíly mezi nimi jsou minimální, RPi 4 obsahuje:

- starší SoC Broadcom BCM2711
- pomalejší RAM LPDDR4-3200
- pouze jeden displej podporuje 4kp60
- neobsahuje PCIe slot a dedikovaný I/O kontrolér
- průměrná spotřeba energie je nižší, díky čemuž nevyžaduje aktivní chlazení

Konkrétně čtvrtá generace byla vybrána především kvůli aktuální dostupnosti a nižší spotřebě (ostatní rozdíly nejsou vzhledem k aplikaci příliš relevantní). Z obecného hlediska bylo Raspberry Pi zvoleno především kvůli následujícím faktorům:

- 40 pinové rozhraní, obsahující běžně používané sběrnice a dostatek GPIO pinů s podporou PWM a přerušení
- velmi dobrá softwarová podpora práce s tímto rozhraním (knihovny *pigpio*, *Pi4J*, atd.)
- klidový příkon 2,7 W [89]
- kvalitní dokumentace a mnoho informačních zdrojů třetích stran
- dedikovaná distribuce GNU/Linux - *Raspberry Pi OS*

Zapojení bylo kvůli nedostatku času (k výměně jednotek došlo poměrně pozdě) realizováno na nepájivém poli. Jednotlivá propojení jsou popsána v tabulce 3.5.

Softwarové řešení je popsáno v následující kapitole.

Raspberry Pi pin (BCM)	Připojené zařízení
5 V	Napájecí linka TWI senzorů, Hallových sond, řídicích obvodů H-můstků a kontaktních senzorů
GND	Společná zem TWI senzorů, Hallových sond, řídicích obvodů H-můstků a kontaktních senzorů
GPIO 0	PWM vstup předního pravého motoru pro pohyb dozadu
TWI SDA (GPIO 2)	Datové piny všech TWI senzorů
TWI SCL (GPIO 3)	Hodinové piny všech TWI senzorů
GPIO 4	Hallova sonda předního levého motoru
GPIO 5	PWM vstup zadního levého motoru pro pohyb dopředu
GPIO 6	PWM vstup zadního levého motoru pro pohyb dozadu
GPIO 9	PWM vstup předního levého motoru pro pohyb dozadu
GPIO 10	PWM vstup předního levého motoru pro pohyb dopředu
GPIO 11	PWM vstup předního pravého motoru pro pohyb dopředu
GPIO 13	PWM vstup zadního pravého motoru pro pohyb dopředu
GPIO 17	Hallova sonda předního pravého motoru
GPIO 19	PWM vstup zadního pravého motoru pro pohyb dozadu
GPIO 22	Hallova sonda zadního pravého motoru
GPIO 23	Přední kontaktní senzor
GPIO 24	Zadní kontaktní senzor
GPIO 27	Hallova sonda zadního levého motoru

Tabulka 3.5: Seznam propojení řídicí jednotky

3.3.3 Řídicí aplikace

Řídící aplikace je dostupná z gitového repozitáře [90] na platformě Codeberg [91], pod licencí *MIT* [92]. Jedná se o aplikaci napsanou v jazyce Java (konkrétně JDK 17), jejímž jádrem je knihovna *mals-intermodule-broker (MIB)*. Ta původně vznikala jako součást řídicí aplikace, ale časem byla vyčleněna do zvláštního projektu, za účelem zvýšení znovupoužitelnosti kódu. Následuje popis knihovny a jednotlivých modulů, ze kterých se aplikace skládá.

3.3.3.1 Knihovna `mals-intermodule-broker`

MIB je knihovna pro usnadnění komunikace mezi částmi aplikace, které provádějí samostatnou činnost, během které produkují (a také konzumují) data užitečná pro ostatní části. Je napsána v jazyce Java (JDK 17) a je dostupná z repozitáře [93] pod licencí *LGPLv2.1* [94]. Z hlediska návrhu je knihovna inspirována protokolem *MQTT*. Tedy, jejím jádrem je *broker*, který přijímá zprávy publikované připojenými *moduly* a následně je předává dalším modulům, které odebírají téma daných zpráv. Komunikační model je tedy *publisher-subscriber* [95].

Aplikace, jež knihovnu využívá, pracuje primárně se dvěma třídami - `TaskModule` a `MalsIntermoduleBroker`:

- Abstraktní třída `TaskModule` reprezentuje výše zmiňovanou samostatně činnou část programu. Uživatel ji využije jako předka pro svou vlastní implementaci, ve které musí přetížit metody `setup()` a `loop()`. Metoda `setup()` slouží k úvodnímu nastavení modulu a je vykonána pouze jednou, na začátku jeho činnosti. Naproti tomu metoda `loop()` je vykonávána periodicky a slouží pro umístění výkonného kódu modulu. Tento styl rozvržení modulu byl zvolen z důvodu, že je poměrně snadno pochopitelný a dobře známý mezi začátečníky, jelikož jej využívá platforma Arduino. [96] Každý `TaskModule` má přístup k instanci třídy `MalsIntermoduleBroker`, u které je registrován a může na ní volat níže popsané metody.

- Třída `MalsIntermoduleBroker` slouží jako prostředník (broker) pro vzájemnou komunikaci modulů. Ty se k ní při vytvoření registrují pomocí metody `registerModule(newModule)` a poté skrze ní komunikují s využitím následujících metod:

  - `subscribe(topic, subscriber)` - Pomocí této metody se modul přihlásí k odběru zpráv (dává sám sebe jako parametr `subscriber`) se zadaným tématem (`topic`). Metoda má rovněž variantu, která přijímá místo jednoho tématu celý list témat (a tedy vytváří odběr pro každé z nich).

  - `unsubscribe(topic, subscriber)` - Slouží ke zrušení odběru zadaného tématu.

  - `publish(topic, payload, priority)` - Odeslání zprávy s daným tématem (`topic`), obsahem (`payload`) a prioritou (`priority`). Obsahem zprávy může být číslo, textový řetězec, boolovská hodnota, JSON objekt, případně nic (zpráva nenese data).

Výše zmíněná témata jsou textové řetězce, které odpovídají následujícímu formátu:

- téma může začínat znakem "/"
- jednotlivé úrovně tématu jsou vždy odděleny znakem "/"
- obsahem úrovní témat mohou být jakékoliv znaky ze znakové sady UTF-16, kromě znaku "/"

Při přihlašování k odběru platí ještě dvě další pravidla:

- je-li jediným znakem úrovně znak "+", jedná se o takzvanou *jednoúrovňovou divokou kartu*, což znamená, že jsou brány v potaz všechny varianty, které jsou na této úrovni dostupné
- obdobně, je-li jediným znakem "#", jedná se o *víceúrovňovou divokou kartu*, a tedy v potaz jsou brány všechny varianty na této i následujících úrovních

Činnost brokeru samotného je následovná: nejprve si ukládá reference na registrované moduly, které po jeho spuštění (zavolání metody `start()`) rovněž spouští. Následně čeká na volání výše zmíněných komunikačních metod. Je-li zavolána metoda `subscribe(topic, subscriber)`, ukládá si referenci na odebírající modul v prefixovém stromu (též známém pod anglickým označením *trie*) [97], pod cestou odpovídající parametru `topic`. Dojde-li k zavolání metody `unsubscribe(topic, subscriber)`, je modul z prefixového stromu odebrán a případné prázdné uzly smazány. Zavoláním metody `publish(topic, payload, priority)` je interně vytvořen objekt typu `Message`, který obsahuje všechny parametry předané metodě a ten je uložen do prioritní fronty zpráv, odkud je, v momentě kdy se dostane na řadu, odebrán a odeslán všem odběratelům zadaného tématu.

Aplikace *mals-mcu* zavádí celkem sedm vlastních modulů, které jsou popsány níže.

3.3.3.2 Modul pro hardwarovou abstrakci (HAL)

Tento modul slouží k ovládání hardwarového rozhraní řídicí jednotky. V systému nahradil *Modul komunikace s řídicí jednotkou subsystémů* a efektivně převzal jeho činnost, tedy přijímání dat ze senzorů a odesílání příkazů motorům. Interně se však moduly podstatně liší, jelikož HAL modul komunikuje s komponentami přímo, kdežto SCU modul skrze řídicí jednotkou subsystémů.

K přístupu k hardwarovým komponentám modul využívá knihovnu Pi4J [98]. Ta tyto komponenty zabaluje do nativních Java objektů, které modul využívá ve vlastních reprezentacích senzorů a motorů. Ty dále obsahují se zařízením spojená MIB témata a veškeré další konstrukce, potřebné pro komunikaci se zařízením. V případě motorů jsou dále k dispozici metody, které umožňují nastavení rychlosti a směru otáčení a u senzorů pak metoda, která funguje jako reakce na přijatá data. Modul obsahuje tyto reprezentace:

- `Motor`
- `Hall`
- `Tactile`
- `TwiDevice` - abstraktní třída definující společné prvky pro všechny senzory komunikující přes TWI; její implementace jsou:
  - `Lidar`
  - `Ssa`

V metodě `setup()` dochází k inicializaci reprezentaci a zadání odběrů na MIB témata. V metodě `loop()` jsou čteny zprávy z MIB fronty a na jejich základě předávány příkazy `Motor` reprezentacím. `Hall` a `Tactile` reprezentace pouze pasivně čekají na signál z přidružených zařízení, na jehož základě publikují odpovídající MIB zprávy a dále `TwiDevice` reprezentace obsahují vlastní vlákno, které po inicializaci periodicky vyžaduje ze zařízení data a po jejich obdržení je publikuje skrz MIB.

3.3.3.3 Modul komunikace s řídicí jednotkou subsystémů

Modul komunikace s řídicí jednotkou subsystémů sloužil původně k překladu a inicializaci přenosu příkazů od ostatních modulů směrem k SCU a také naměřených senzorických dat v opačném směru. Využíval k tomu knihovnu SRSC, která je blíže popsána v kapitole 3.3.1.2. S odstraněním SCU byl též odebrán. Stručný popis jeho činnosti je uveden dále.

Modul si udržuje informaci o senzorech připojených k SCU a obdrží-li skrze SRSC kanál zprávu nesoucí data z některého z nich, transformuje ji na MIB zpávu, kterou odešle brokeru. Podobný postup je aplikován při přijetí MIB zprávy s řídicím příkazem pro SCU. Ta je přeložena na odpovídající SRSC zprávu a odeslána. Zpracování MIB zpráv probíhá v metodě `loop()`, zatímco zpracování SRSC v callbacku předávaném knihovně metodou `registerOnPacketArrivedCallback`

`(callback)`. Vzhledem k tomu, že data ze senzorů přicházejí ve formě pole bytů, je nutné před odesláním provádět konverzi na odpovídající formát, který je definován typem senzoru. K tomu slouží handlery ve třídě `SrscPayloadProcessor`.

3.3.3.4 Modul komunikace protokolem WebSocket

Cílem tohoto modulu je poskytnout asynchronní rozhraní řídícího subsystému, pro připojení klientské aplikace. K tomu využívá protokol WebSocket, definovaný jako RFC6455 [99], a to prostřednictvím knihovny *Java-WebSocket* [100]. Princip činnosti modulu je poměrně jednoduchý a spočívá v překladu MIB volání (*publish*, *subscribe*, *unsubscribe*) na zprávy přenositelné skrze WebSocket spojení a naopak. Díky tomu umožňuje začlenění klienta do systému, jako by byl standardním modulem.

Zprávy směřující ke klientu mohou být pouze produktem volání *publish*, jejich formát tedy odpovídá JSON reprezentaci MIB zpráv (viz fragment 3.1). Zprávy od klientské aplikace mohou odpovídat všem MIB voláním, proto obsahují navíc hlavičku pro identifikaci typu zprávy a zpráva samotná je umístěna do pole `payload`. Tělo (payload) zprávy typu *publish* je shodné s výše popsanou zprávou ke klientu. Zprávy typu *subscribe* a *unsubscribe* obsahují krom hlavičky jen téma, ke kterému má být přihlášen či odhlášen odběr (viz fragment 3.2).

```
1 {
2     "topic": "some/topic",
3     "payload": {
4         ...
5     },
6     "priority": "NORMAL"
7 }
```

Fragment 3.1: Příklad odchoží WebSocket zprávy

```
1 {
2     "type": "SUBSCRIBE",
3     "payload": {
4         "topic": "some/topic"
5     }
6 }
```

Fragment 3.2: Příklad příchozí WebSocket zprávy typu SUBSCRIBE

Vnitřně modul využívá knihovní třídu `WebSocketServer` (dědí z ní třída `Server`), která poskytuje k přetížení metody pro reakci na události týkající se WebSocket spojení. Nejdůležitější z nich je metoda `onMessage()`, která je zavolána v momentě úspěšného přijetí zprávy z klienta. Modulová implementace této metody zprávu parsuje a podle jejího typu volá příslušnou MIB metodu s daty získanými ze zprávy. Dále stojí za zmínku metoda `onOpen()` (volá se při otevření WebSocket spojení), jejíž implementace kontroluje, zdali je při zavolání již aktivní spojení (tzn. je připojen jiný klient), a pokud ano, nové spojení je ihned ukončeno. Takto je zamezeno tomu, aby docházelo k nežádoucím zásahům do řízení z více klientů najednou.

Využití zděděných metod modulu je následující:

- `setup()` provádí pouze nastavení a spuštění instance třídy `Server`
- `loop()` zpracovává přijaté MIB zprávy, vytváří z nich JSON objekty a ty odesílá prostřednictvím instance třídy `Server` klientu

Modul sám není přihlášen k odběru žádných MIB zpráv. K přihlášení dochází až po přijetí zprávy typu `SUBSCRIBE` od připojeného klienta.

3.3.3.5 Moduly řízení a detekce kolizí

Moduly autonomního řízení, řízení sledováním linky, manuálního řízení a detekce kolizí slouží k vykonávání procesů popsaných v kapitole 3.5. Jejich činnost (s výjimkou manuálního řízení) lze shrnout následovně: po spuštění modul čeká na data ze senzorů (autonomní na pulzy z Hallových sond, sledování linky na snímky z kamery a detekce kolizí na měření z LiDARu, sonarů a kontaktních senzorů), po jejich přijetí je začne zpracovávat a po dokončení zpracování odesílá příkazy zbytku systému (řízení změny rychlostí motorů a detekce kolizí oznámení o kolizi). Poté opět čeká na přijetí dalších dat.

Jak již bylo řečeno, modul manuálního řízení je výjimkou. Jeho činnost totiž nespočívá v čekání na senzorická data, ale na příkazy od uživatele. V ostatních ohledech se však shoduje s předešlými.

3.4 Klientská aplikace

Obsluha prostředku je realizována prostřednictvím webové aplikace naprogramované v jazyce TypeScript s využitím frameworku Svelte. Ten by zvolen především kvůli rychlosti načítání stránek, což je způsobeno maximalizací optimalizace kódu při kompilaci (v prohlížeči se zpracovává pouze to, co je skutečně nutné) a jednoduchosti

tvorby vlastních komponent. Zdrojový kód aplikace je dostupný v repozitáři [101] pod licencí MIT.

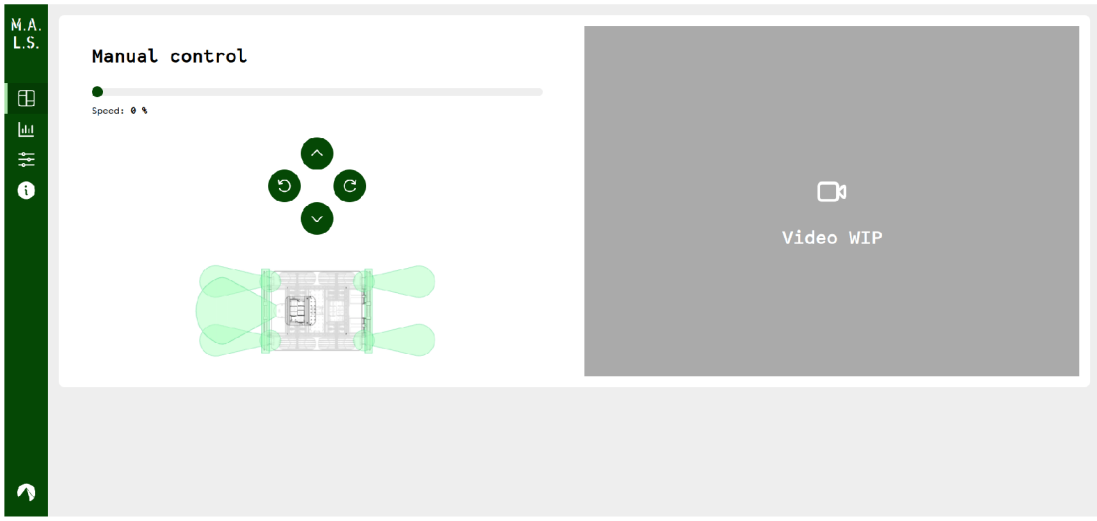
Aplikace obsahuje čtyři obrazovky: řízení, monitoring, nastavení a informace o aplikaci, které jsou popsány níže. Společným prvkem všech obrazovek je postranní menu panel, který obsahuje tlačítka pro přepínání mezi obrazovkami a přechod do repozitáře se zdrojovým kódem. Na pozadí aplikace využívá WebSocket API pro připojení k řídicímu programu. To je zapouzdřeno v modulu, který napodobuje rozhraní a činnost MIB brokeru a umožňuje tak obrazovkám komunikovat s moduly řídicí aplikace. Obrazovky se mohou přihlašovat k odběru MIB témat a publikovat zprávy, jako kdyby se jednalo o standardní moduly. Spojení s řídicí aplikací se navazuje při otevření aplikace a udržuje se otevřené do jejího zavření.

Stav aplikace je udržován pomocí tzv. *stores*, což jsou sdílené objekty, přístupné ze všech jejích součástí, které existují po celou dobu připojení klienta. Jsou v nich uloženy konfigurační hodnoty metod řízení, aktivní nastavení a výše zmíněný WebSocket modul.

3.4.1 Obrazovka řízení

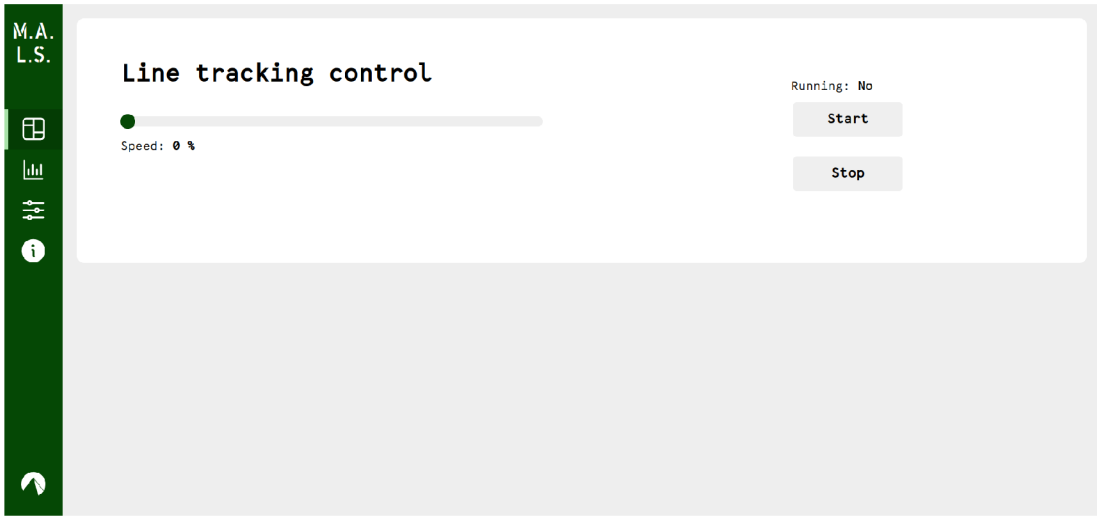
Úvodní obrazovkou po zapnutí aplikace je obrazovka určená k řízení prostředku. Ta může mít tři podoby, podle aktuálně zvolené metody řízení (viz podkapitola 3.4.3). Výchozím režimem je manuální řízení, kde může uživatel nastavovat rychlost jízdy pomocí posuvníku a určovat směr jízdy tlačítky se šipkami. K pohybu prostředku dochází při držení stisku tlačítka - při stisknutí je odeslána zpráva s informací o směru pohybu a s aktuální hodnotou rychlosti; při uvolnění stisku je odeslána zpráva s příkazem zastavit. Význam šipek je následující: šípky dopředu a dozadu spouští otáčení všech motorů v požadovaném směru stejnou rychlostí, kdežto levá a pravá šipka spouští otáčení na místě, kdy jsou všechny motory hnány stejnou rychlostí, ale levá strana opačným směrem než pravá.

Obrazovka dále obsahuje indikátor kolizí, což je zjednodušená vizualizace prostředku s přibližnými oblastmi snímání senzorů. Ty jsou ve výchozím stavu zelené, dojde-li však k detekci potenciální kolize či pádu, zbarví se červeně. Cílem tohoto indikátoru je poskytnout uživateli dodatečné informace o okolí prostředku a předejít tak potenciálnímu přehlédnutí rizikové situace. Na pravé polovině obrazovky je prostor pro budoucí umístění obrazu z kamery, čímž se umožní manuální ovládání i mimo oblast přímého vizuálního kontaktu.



Obrázek 3.21: Obrazovka manuálního řízení *(zdroj: vlastní)*

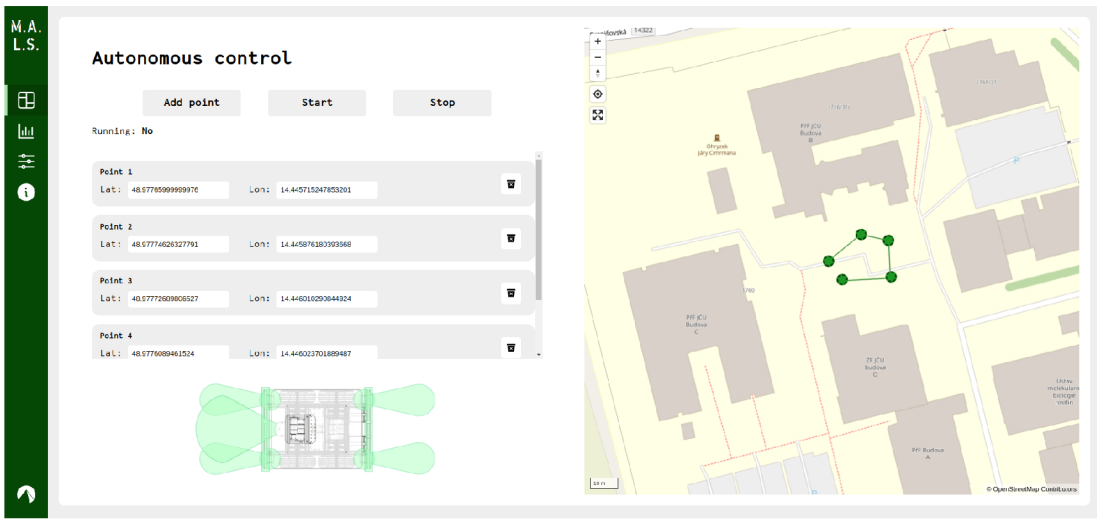
Druhým režimem je řízení sledováním linky. Zde obrazovka obsahuje posuvník pro nastavení rychlosti jízdy a tlačítka pro spuštění a zastavení jízdy. Dále je zde indikátor aktivní jízdy.



Obrázek 3.22: Obrazovka řízení sledováním linky *(zdroj: vlastní)*

Posledním režimem je autonomní jízda podle souřadnic. Tato obrazovka se skládá z mapy zobrazující trasu, seznamu bodů trasy, indikátoru kolizí a tlačítek pro přidání nového bodu trasy, spuštění jízdy a jejího přerušení. Mapa je vytvořena pomocí knihovny *svelte-maplibre* a na začátku obsahuje dva body (start a cíl cesty). Těmi lze po mapě pohybovat a měnit tak souřadnice cesty. Seznam bodů obsahuje souřadnice bodů z mapy, možnost upravit jejich hodnotu a též je odstranit (vždy však musí zůstat minimálně dva body). Indikátor kolizí funguje shodně s tím použitým v obrazovce ma-

nuálního řízení. Tlačítka umožňují spustit a zastavit řízení a přidat nový bod do trasy. Ten je umístěn poblíž posledního bodu a uživatel jej může dále přemístit na požadované souřadnice. Je-li řízení spuštěno, dochází k zamknutí trasy a nelze přidávat ani odebírat body a též nelze měnit jejich pozice.



Obrázek 3.23: Obrazovka autonomní jízdy podle souřadnic *(zdroj: vlastní)*

3.4.2 Obrazovka monitoringu

Ke sledování aktuálního stavu senzorů prostředku slouží obrazovka monitoringu. Ta obsahuje moduly reprezentující jednotlivé senzory a zobrazující jejich naměřené hodnoty. Obsah modulu závisí na typu senzoru, který reprezentuje:

- *LiDAR* - aktuální hodnota vzdálenosti od překážky a graf[1] s předešlými hodnotami, rozloženými podle úhlu natočení LiDARu
- *Sonarová soustava* - aktuální hodnota vzdálenosti od překážky a od země; graf se třiceti předešlými hodnotami obou veličin
- *kontaktní senzor* - informace o kontaktu

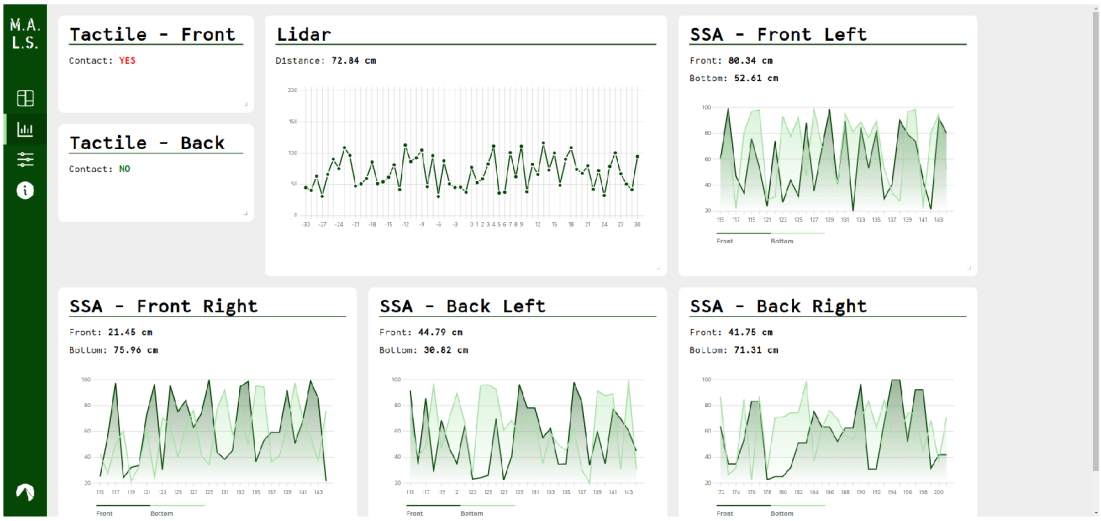
Samotné senzory jsou definovány jako JSON objekty a jejich seznam je umístěn v souboru *monitoring.json*. Objekty obsahují název senzoru, unikátní identifikátor, typ, seznam MIB témat s daty senzoru a informace o rozměrech a fyzickém umístění senzoru v rámci obrazovky. Některé typy senzorů přidávají vlastní informace, jako např. počet vzorků zobrazených v grafu. Příklad objektu senzoru lze vidět v následujícím fragmentu.

[1] Grafy jsou vytvořeny pomocí knihovny *svelte-frappe-charts*.

```
{
    "metadata": {
        "id": 4545,
        "name": "SSA - Front Left",
        "category": 1,
        "type": "SSA",
        "topics": {
            "front": "scu.twiSensors.ssaFlf",
            "bottom": "scu.twiSensors.ssaFlb"
        }
    },
    "boardItem": {
        "x": 6,
        "y": 0,
        "w": 3,
        "h": 5,
        "min": {
            "w": 2,
            "h": 5
        },
        "max": {
            "h": 5
        }
    },
    "measurements": 30
}
```

Fragment 3.3: Příklad definice senzoru

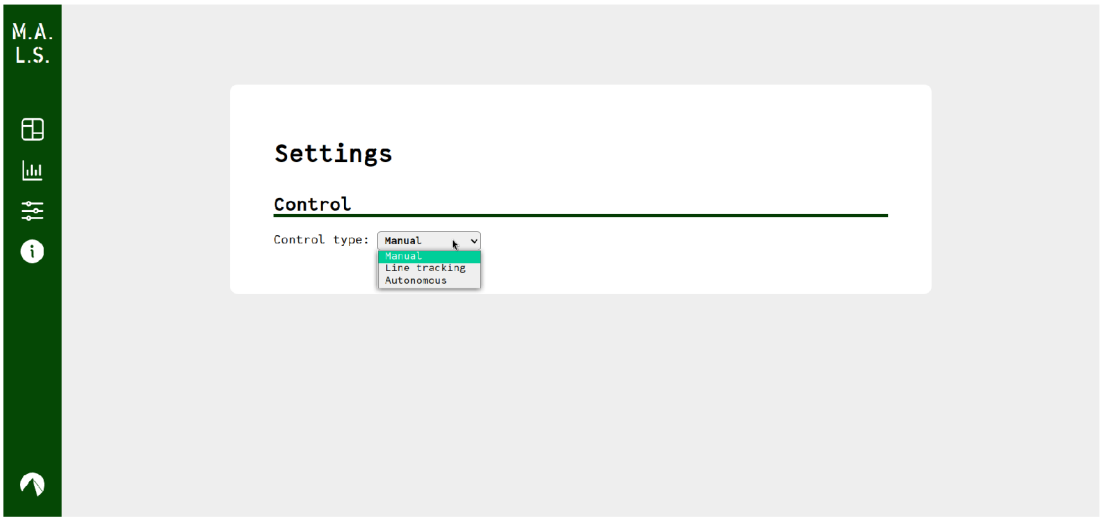
Obrazovka po načtení vykresluje postupně jednotlivé senzory a zadává příkazy k odběru jejich témat. Poté čeká na příchozí zprávy s naměřenými vzorky a zobrazuje je.



Obrázek 3.24: Obrazovka monitoringu *(zdroj: vlastní)*

3.4.3 Obrazovka nastavení

Nastavení aplikace je momentálně velmi jednoduché, obsahuje pouze jednu položku, a to typ řízení. Její změnou dojde k přepnutí aktivního režimu řízení na první obrazovce a rovněž je vyslán požadavek na příslušný endpoint API řídicí aplikace, který změní aktivní metodu řízení v ní.

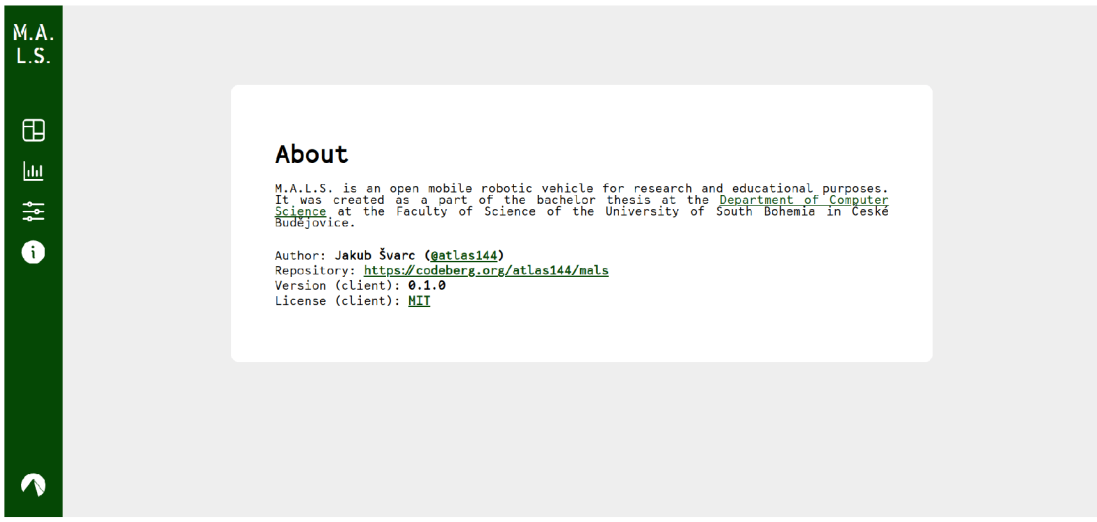


Obrázek 3.25: Obrazovka nastavení *(zdroj: vlastní)*

3.4.4 Obrazovka informací o aplikaci

Poslední obrazovka obsahuje krátký popis celého projektu a základní informace o něm. Vzhledem k tomu, že neobsahuje žádné proměnlivé prvky, je celý její obsah vytvořen

při kompilaci a server ji posílá jako obyčejnou statickou stránku.



Obrázek 3.26: Obrazovka informací o aplikaci *(zdroj: vlastní)*

3.5 Řízení prostředku

Dílčím cílem práce byl návrh a implementace dvou metod řízení prostředku, a to řízení čtením linky, doplněné o detekci značek a samostatná jízda po trase zadané souřadnicemi. Obě metody byly implementovány a doplněny dále o metodu manuálního řízení, pro snadnější manipulaci s prostředkem. Následuje popis všech tří metod.

3.5.1 Sledování linky a čtení značek

Metoda sledování linky je nejsnáze implementovatelná pomocí tří (případně dvou) reflexních optočlenů, které jsou umístěny na spodní straně prostředku, co nejblíže nad zemí. Ty snímají odrazivost povrchu pod sebou a podle ní je dále určeno, zdali se optočlen nachází nad linkou či nad jejím okolím. Směr pohybu prostředku se určuje podle toho, které optočleny se aktuálně nacházejí nad linkou. Tento způsob je jednoduchý a potřebné senzory levné, nelze jej však bez přídavných senzorů kombinovat s čtením značek.

Z toho důvodu byl zvolen přístup, využívající ke snímání povrchu kameru, která dokáže detekovat linku i značky. Zvolená kamera je popsána v podkapitole 3.2.5. Ke zpracování obrazu z kamery je využita knihovna *OpenCV*. Jako symboly, které prostředek detekuje, byly zvoleny různobarevné kruhové značky, kde význam je dám právě barvou.

Řízení funguje následovně: nejprve dojde k pořízení, uložení a načtení snímku do paměti (uložení je nutnou součástí pořízení snímku - vlastnost knihovny *libcamera*, která ovládá kameru). Poté je snímek převeden na maticovou reprezentaci a začíná hledání symbolů. To se provádí postupně třikrát pro různé barvy (červenou, modrou a zelenou). Nejprve je snímek transformován na HSV barevný model a pomocí metody `inRange` převeden na černobílý (pouze pixely, jejichž barva spadá do zadaného HSV rozsahu, jsou bílé, ostatní černé). Poté je na takto předzpracovaný snímek použita *Houghova transformace* pro detekci kruhů (metoda `HoughCircles`).

Je-li kruh nalezen, dochází k zastavení prostředku a informování spolupracujícího zařízení o detekování značky. Poté prostředek čeká, dokud neobdrží odpověď, signalizující konec činnosti druhého zařízení. Po obdržení pokračuje dál v cestě, ale dokud nezmizí ze zorného pole kamery přečtený symbol, jsou všechny symboly ignorovány (aby se předešlo opětovné detekci téhož symbolu).

Po detekci symbolů dochází ke spuštění detekce linky, která je inspirována [102]. Začíná opět převedením snímku na černobílý, pomocí metody `inRange`, tentokrát však podle jasu pixelů. Následuje detekce obrysů metodou `findContours`. Poté je vybrán největší z nalezených a ten je dále považován za *cestu*. Z té je vypočítán střed a podle jeho horizontální pozice určen směr následujícího pohybu. Následuje vyslání MIB zpráv, které odpovídajícím způsobem nastaví rychlost motorů.

Tento postup se cyklicky opakuje během celého procesu řízení. Uživatel jej může pozastavit, opět spustit a nastavit rychlost jízdy.

3.5.2 Autonomní jízda podle souřadnic

Druhou zadanou metodou řízení je autonomní jízda podle daných souřadnic. Tu lze pojmout celou řadou způsobů (např. [103], [104], [105]), zde byl použit přístup, který klade důraz na kvalitu vstupů zadaných operátorem. Ten na začátku procedury definuje dílčí cíle cesty, kterými prostředek postupně projíždí a je velmi vhodné, aby se takto zvolená cesta vyhýbala známým (statickým) překážkám, zvláště pak těm komplexním. Takto je pro prostředek jednodušší dosáhnout optimální cesty a vyhnout se uváznutí způsobenému nezdařilými pokusy o vyhnutí se komplexním statickým překážkám.

Struktura řídicího procesu do značné míry odpovídá struktuře popsané v kapitole 2.2, tedy počáteční fází je *získávání informací* o prostředku a jeho okolí, z nich dále čerpá *lokalizace* prostředku, jejíž výsledky spolu s výsledky *tvorby mapy* zpracovává proces *plánování cesty*. Nakonec jsou výstupy plánování cesty spolu s výstupy *detekce překážek a prevence pádu* převedeny *řízením pohybu* na rychlosti motorů. Následuje detailní popis těchto fází.

Získávání informací o prostředku a jeho okolí probíhá prostřednictvím senzorického subsystému. Konkrétně o interní snímání se starají Hallovy sondy a o externí LiDAR, sonarové soustavy a kontaktní senzory. Data z nich přicházejí bez předešlého zpracování. Popis činnosti jednotlivých senzorů je uveden v kapitole 3.2.

Lokalizace se provádí pomocí údajů z Hallových sond metodou *odometrie*. Jedná se tedy o relativní lokalizaci, jejíž obecný princip je popsán v podkapitole 2.2.2. Konkrétní postup výpočtu, převzatý z [106] následuje. Na úvod je nutno zmínit, že pro zjednodušení výpočtu je uvažováno pouze jedno kolo na každé straně prostředku. Toto zjednodušení je možné díky tomu, že obě kola na shodné straně jsou v každém okamžiku poháněna stejnou rychlostí.

Základem výpočtu je převodový poměr impulzů z Hallovy sondy na lineární posun kola c. Ten lze vypočítat takto:

$$c = \frac{\pi d}{nC}, \tag{3.7}$$

kde d je průměr kola, n převodový poměr převodovky mezi motorem a kolem a C počet pulzů z Hallovy sondy na otáčku motoru. S využitím c je vypočítán přírůstek ujeté vzdálenosti ve snímací periodě I zvlášť pro kola na levé $\Delta U_{L,\ I}$ a na pravé $\Delta U_{R,\ I}$ straně takto:

$$\Delta U_{L/R,\ I} = cN_{L/R,\ I}, \tag{3.8}$$

přičemž $N_{L/R,\ I}$ znamená počet impulzů levé, respektive pravé sondy během periody I. Z nich je dále vypočítán přírůstek ujeté vzdálenosti středu prostředku ΔU_I a přírůstek změny orientace prostředku $\Delta\theta_I$:

$$\Delta U_I = \frac{U_{L,\ I} + U_{R,\ I}}{2} \tag{3.9}$$

$$\Delta\theta_I = \frac{U_{L,\ I} - U_{R,\ I}}{b}, \tag{3.10}$$

kde b je rozchod nápravy. Nakonec je vypočítána nová relativní pozice x_I, y_I a orientace θ_I prostředku:

$$x_I = x_{I-1} + \Delta U_I cos\theta_I \tag{3.11}$$

$$y_I = y_{I-1} + \Delta U_I sin\theta_I \tag{3.12}$$

$$\theta_I = \theta_{I-1} + \Delta\theta_I. \tag{3.13}$$

Tvorba mapy je zde značně zjednodušena. K tvorbě dochází pouze na začátku, a to z cesty zadané operátorem. Překážky potkané po cestě nejsou přidávány, zpracovávají se pouze v daném okamžiku. Veškerá činnost zde tedy spočívá ve sledování pokroku v dosahování dílčích cílů cesty.

Plánování cesty je rovněž zjednodušeno. Vzhledem k tomu, že podstatnou část plánování - zpracování detekovaných překážek, zde přebírá kontrolér pohybu (viz dále), je činnost plánovače omezena na výpočet odchylky vzdálenosti Δd a natočení $\Delta\theta$ od následujícího dílčího cíle, a to takto:

$$\Delta d = \sqrt{(x_G - x_I) + (y_G - y_I)} \tag{3.14}$$

$$\Delta\theta = arctg\frac{y_G - y_I}{x_G - x_I} - \theta_I, \tag{3.15}$$

kde x_I a y_I jsou souřadnice aktuální relativní pozice, x_G a y_G souřadnice nejbližšího dílčího cíle a θ_I aktuální relativní orientace.

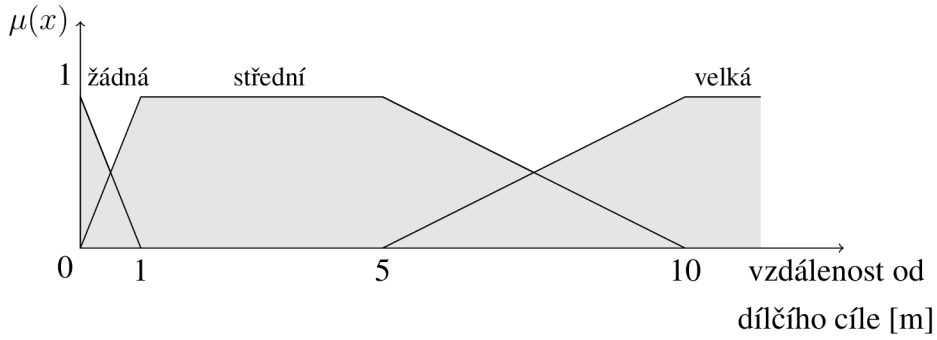
Řízení pohybu provádí fuzzy kontrolér, jehož princip je popsán v podkapitole 2.2.5. Užitá implementace vychází z kontroléru popsaného v článku [107]. Kontrolér je vytvořen pomocí knihovny *jFuzzyLogic* popsané v pracích [108] a [109]. Ta využívá pro definici komponent kontroléru jazyk FCL, definovaný standardem IEC 61131-7.

```
1  FUZZIFY angle
2      TERM back := (-180, 1) (-90, 0) (90, 0) (180, 1);
3      TERM left := (-180, 0) (-90, 1) (0, 0);
4      TERM front := (-90, 0) (0, 1) (90, 0);
5      TERM right := (0, 0) (90, 1) (180, 0);
6  END_FUZZIFY
7
8  ...
9
10 RULE 1 : IF distance IS none THEN leftSpeed IS stop,
      rightSpeed IS stop;
11 RULE 2 : IF angle IS left AND distance IS small THEN
      leftSpeed IS forSlow, rightSpeed IS backSlow;
```

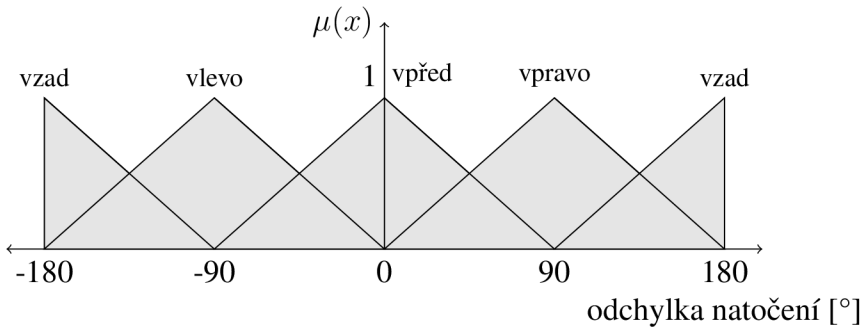
Fragment 3.4: Ukázka definice funkcí příslušnosti a fuzzy pravidel v jazyce FCL

Vstupní proměnné kontroléru jsou *odchylka vzdálenosti od dílčího cíle*, *odchylka natočení* a *vzdálenost od překážky vpředu* a *vzadu*. Výstupní proměnné jsou *střídy*

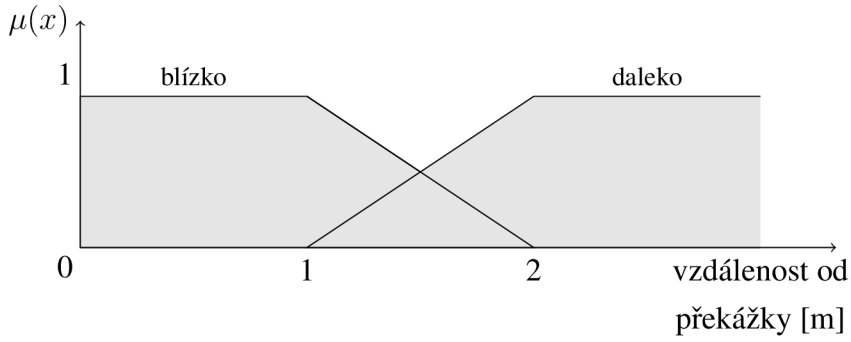
PWM signálu pro motory na levé a na pravé straně. Jejich funkce příslušnosti $\mu(x)$ lze vidět na obrázcích 3.27 až 3.30:



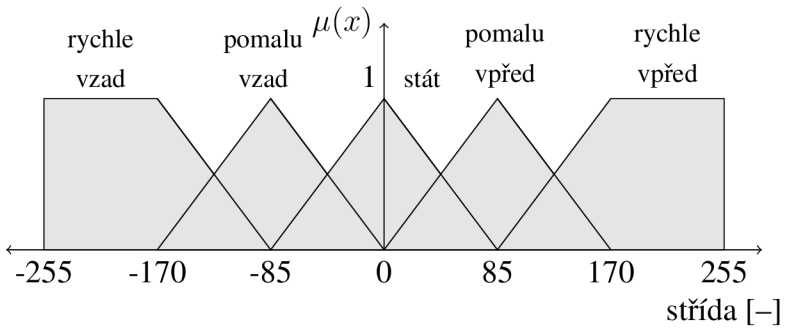
Obrázek 3.27: Funkce příslušnosti pro vzdálenost od dílčího cíle *(zdroj: vlastní)*



Obrázek 3.28: Funkce příslušnosti pro odchylku natočení *(zdroj: vlastní)*



Obrázek 3.29: Funkce příslušnosti pro vzdálenost od překážky *(zdroj: vlastní)*



Obrázek 3.30: Funkce příslušnosti pro střídy PWM signálu motorů *(zdroj: vlastní)*

Defuzzyfikace se provádí výpočtem těžiště t_I pomocí výpočtů těžišť dílčích elementů plochy pod křivkou:

$$t_I = \frac{\sum_{k=1}^{n} t_{Ik} \cdot S_k}{\sum_{k=1}^{n} S_k}, \tag{3.16}$$

kde n je počet fuzzy množin počítané veličiny (tedy pro střídu PWM signálu $n = 5$), t_k je těžiště plochy množiny k a S_k její obsah. Výsledkem procesu jsou hodnoty střídy PWM signálu, jež jsou dále odeslány motorům.

3.5.3 Manuální řízení

Pro zjednodušení přesouvání prostředku byla přidána třetí metoda řízení, a to řízení manuální. Metoda je navržena tak, aby umožnila jednoduché a přesné umístění prostředku pomocí příkazů od operátora. Lze ji využít např. pro přistavení prostředku k lince při řízení první popsanou metodou či k nastavení prostředku do výchozí pozice metody druhé. Operátor má k dispozici příkazy pro pohyb vpřed a vzad, natočení vlevo a vpravo a nastavení rychlosti pohybu.

3.5.4 Detekce kolizí a prevence pádu

Paralelně k řízení dochází v samostatném modulu k detekci potenciálních kolizí a pádů. Činnost detekce spočívá v přijímání dat z LiDARu, sonarů a kontaktních senzorů, jejich kumulaci a následném zpracování. Konkrétní postup je specifický pro každý typ senzoru:

- U *LiDARu* dochází po přijetí zprávy s daty nejprve k přepočtu přímé vzdálenosti od čela prostředku (pomocí goniometrických funkcí), ta je následně porovnána z prahovou vzdáleností (30 cm) a pokud je nižší, hodnota je přidána do akumulátoru. Nashromáždí-li se takto pět po sobě jdoucích hodnot (opatření proti falešným detekcím), situace je vyhodnocena jako blížící se kolize a systém je o tom informován MIB zprávou.
- Činnost *sonaru* je podobná činnosti LiDARu, rozdílem je, že na začátku nedochází k přepočtu vzdálenosti, ale využívá se rovnou přijatá hodnota. Vzdálenost čelních sonarů je shodná s LiDARem, u spodních (prevence pádů) je to 15 cm.
- U *kontaktních senzorů* dochází k reakci okamžitě, tedy hned po přijetí zprávy.

Reakce na detekovanou kolizi je poté v režii jednotlivých metod řízení.

3.6 Testování

Testování probíhalo během celého procesu návrhu prostředku, díky čemuž bylo možné ihned reagovat na případné problémy a předejít tak jejich kumulaci. Tento postup se osvědčil především u návrhu 3D tištěných dílů, kde byly často nejprve tištěny dílčí části komponent a teprve po jejich úspěšném otestování i celé komponenty. Takto se předešlo plýtvání materiálem, které by způsobil tisk nevyhovujících komponent.

Při návrhu elektronických součástí předcházelo tvorbě DPS vždy testování obvodu na nepájivém poli. Deska byla vyfrézována vždy až po kontrole funkčního napájení, absence zkratů a zkoušce celkové funkčnosti obvodu. Díky tomu nebylo nutné žádnou desku vyrábět vícekrát. Výjimkou je řídicí jednotka subsystémů, u které testování proběhlo v pořádku, ale výsledná DPS nebyla funkční.

Po dokončení konstrukce byla testována jízda po různých typech terénů. Jednalo se o asfalt, dlažební kostky, rovný travnatý terén a členitý travnatý terén. Jízda na všech typech terénu probíhala bez větších obtíží. Dále byla testována jízda se zátěží simulující robotické rameno, která též proběhla v pořádku.

Během provozu byla orientačně měřena teplota řídicí jednotky, H-můstků a napěťového měniče. Ani v jednom případě nedošlo k přiblížení k hraniční teplotě a tedy nebylo nutné přidávat dodatečné chlazení.

Bezpečnost byla testována simulování přítomnosti překážek v bezprostřední blízkosti prostředku. Nejprve byl vyvinut tlak na přední a zadní kontaktní senzor, na což prostředek bezprostředně zareagoval. Dále byla použita dřevěné deska k postupnému přibližování k LiDARu a čelním sonarům. Po překročení zadané prahové vzdálenosti došlo vždy k detekci překážky. Průměrná odchylka od skutečné vzdálenosti k desce byla u LiDARu 1,3 cm a u sonarů 3,4 cm. Poté byl podobný postup aplikován na spodní sonary, zde se ale deska od snímačů naopak oddalovala. Výsledky byly téměř totožné s čelními sonary.

Z hlediska zabezpečení systému byla testována ochrana proti zásahu nežádoucí osoby do řízení prostředku. Bylo-li již aktivní WebSocket spojení s klientem, server úspěšně odmítl druhé spojení, čímž nežádoucí zásah znemožnil.

4. Závěr

Cílem práce bylo navrhnout a vytvořit prototyp kolového robotického prostředku a jeho řízení. Vzniklé řešení je univerzální, rozšiřitelné a umožňuje převážet jiná zařízení (např. robotické rameno). Je schopno základní autonomní jízdy podle horizontálního značení a GPS souřadnic, při které dochází ke kontrole kolizí a pádů, na které prostředek reaguje. V neposlední řadě umožňuje ruční ovládání operátorem, které předešlé dvě metody doplňuje.

Řídicí software je navržen tak, aby umožnil jednoduchou výměnu a rozšíření funkcionality, díky čemuž lze prostředek využít např. k testování různých řídicích algoritmů. Podobný princip byl aplikován při návrhu senzoriky, kde je k připojování senzorů zvolena univerzální škálovatelná sběrnice, doplněná o modulární softwarovou reprezentaci.

Fyzicky je řešení navrženo tak, aby jej bylo možno co nejjednodušeji replikovat, a to s maximálním využitím běžně dostupných nástrojů. Jedinými specializovanými nástroji použitými při výrobě jsou 3D tiskárna a laserový plotr. Obě dvě zařízení lze nahradit zadáním poptávky u firem, které se prací s nimi zabývají, případně využitím veřejných dílen (tzv. *hackerspaců*), jež je obvykle mají k dispozici. Pro výrobu dílů řezaných plotrem lze též využít alternativních materiálů, které umožňují jednoduché zpracování běžnými nástroji, jako např. dříve zmiňovaná dřevotříska. Replikaci a též potenciálnímu vylepšení řešení napomáhá i fakt, že veškeré zdrojové kódy, schémata elektroniky, návrhy DPS a zdrojové CAD soubory jsou publikovány pod otevřenou licencí (*MIT*, *LGPL v2.1* a *CERN OHL SR v2.0*) na platformě Codeberg a kdokoliv je tak může využít, upravit a dále sdílet.

4.1 Budoucí rozvoj

Vzhledem k tomu, že řešení bylo od začátku navrhováno jako univerzální platforma pro co nejširší využití, je zde velký prostor pro budoucí rozvoj.

Jak již bylo naznačeno výše, řízení prostředku je možno doplnit o další metody, proto jednou ze zajímavých možností budoucího rozvoje je testování pokročilých metod řízení s využitím komplexnějších senzorů. Další možností je kooperace s dalšími zařízeními, jako je například dříve zmíněné robotické rameno.

Z hlediska kvality softwaru by bylo žádoucí posílit a sjednotit logování událostí v systému, jelikož to momentálně poskytuje pouze kusé informace. Dále by bylo vhodné umožnit definování senzorů mimo zdrojový kód, pomocí konfiguračních souborů.

Seznam použité literatury

1. MADAN, Anusha; GANESH, Arati; N, Preethi; BANDEKAR, Sahana; NAGASHREE, A.N. Mobile Covid Sanitization Robot. In: *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*. 2021, s. 131–136. Dostupné z DOI: 10.1109/IRIA53009.2021.9588769.
2. YANG, Geng; LV, Honghao; ZHANG, Zhiyu; YANG, Liu; DENG, Jia; YOU, Siqi; DU, Juan; YANG, Huayong. Keep Healthcare Workers Safe: Application of Teleoperated Robot in Isolation Ward for COVID-19 Prevention and Control. *Chinese Journal of Mechanical Engineering*. 2020, roč. 33, č. 1, s. 47. ISSN 2192-8258. Dostupné z DOI: 10.1186/s10033-020-00464-0.
3. LEE, Min-Fan Ricky; CHEN, Yi-Ching Christine. COVID-19 Pandemic Response Robot. *Machines*. 2022, roč. 10, č. 5. ISSN 2075-1702. Dostupné z DOI: 10.3390/machines10050351.
4. CHEN, Xiaoqi; CHEN, Y.Q.; CHASE, James. Mobiles Robots - Past Present and Future. In: 2009. ISBN 978-953-307-001-8. Dostupné z DOI: 10.5772/6986.
5. ŠÍMA, Michal. *Plánování cesty robota ve spojitém prostředí* [online]. Brno, 2006 [cit. 2023-01-06]. Dostupné z: https://docplayer.cz/918524-Diplomova-prace-planovani-cesty-robota-ve-spojitem-prostredi.html. Dipl. pr. Vysoké učení technické v Brně, Fakulta strojního inženýrství. Vedoucí práce Jiří DVOŘÁK.
6. KUBÍČEK, Michal. *Všesměrové roboty – návrh konstrukce* [online]. Brno, 2012 [cit. 2023-01-06]. Dostupné z: https://core.ac.uk/display/30292377. Bak. pr. Vysoké učení technické v Brně, Fakulta strojního inženýrství. Vedoucí práce Radomil MATOUŠEK.
7. TAJTI, Ferenc; SZAYER, Géza; KOVÁCS, Bence; KORONDI, Péter. Robot base with holonomic drive. *IFAC Proceedings Volumes*. 2014, roč. 47, č. 3, s. 5715–5720. ISSN 1474-6670. Dostupné z DOI: https://doi.org/10.3182/20140824-6-ZA-1003.00785. 19th IFAC World Congress.

8. KAO, I-Hsi; SU, Jian-an; PERNG, Jau-Woei. Design of Outdoor Autonomous Moble Robot. *CoRR*. 2022, roč. abs/2201.12605. Dostupné z arXiv: 2201.12605.

9. RASAM, HIMANSHU R. Review on Land-Based Wheeled Robots. *MATEC Web of Conferences* [online]. 2016, roč. 53, 01058–p.3 [cit. 2023-01-06]. Dostupné z DOI: 10.1051/matecconf/20165301058.

10. BUILDEX.CZ. *Náhradní kolo k rudlu G21* [online]. 2024-04-12 [cit. 2024-04-12]. Dostupné z: https://www.buildex.cz/nahradni-kolo-k-rudlu-g21.

11. DGQ. *DGQ 5" Caster wheels Heavy Duty Premium Commercial Grade Non-Marking Set of 4 (2 Rigid, 2 Swivel with Brakes)* [online]. 2024-04-12 [cit. 2024-04-12]. Dostupné z: https://www.amazon.com/DGQ-Caster-Premium-Commercial-Non-Marking/dp/B07856M8MS.

12. TĂTAR, Mihai Olimpiu; CIREBEA, Claudiu; MÂNDRU, Dan. Structures of the Omnidirectional Robots with Swedish Wheels. In: *Mechatronic Systems and Materials IV*. Trans Tech Publications Ltd, 2013, sv. 198, s. 132–137. Solid State Phenomena. Dostupné z DOI: 10.4028/www.scientific.net/SSP.198.132.

13. SONI, Sanket; MISTRY, Trilok; HANATH, Jayesh. Experimental Analysis of Mecanum wheel and Omni wheel. *IJISET - International Journal of Innovative Science, Engineering & Technology*. 2014, roč. 1, č. 3, s. 292–295. ISSN 2348-7968. Dostupné také z: https://www.ijiset.com/v1s3/IJISET_V1_I3_50.pdf.

14. MR.WHO. *Mecanum Wheel* [online]. 2024-01-12 [cit. 2024-04-12]. Dostupné z: https://grabcad.com/library/mecanum-wheel-119.

15. *Rotacaster Omni-Directional Wheels* [online]. 2024-04-12 [cit. 2024-04-12]. Dostupné z: https://www.materialshandling.com.au/products/rotacaster-omnidirectional-wheels/.

16. WAIBEL, Markus. *Project Rezero: Ball Balancing Robot With Style* [online]. 2010-06-08 [cit. 2024-04-12]. Dostupné z: https://spectrum.ieee.org/project-rezero-ball-balancing-elegance.

17. GÜVEN, Yılmaz; COŞGUN, Ercan; KOCAOĞLU, Sıtkı; GEZICI, Harun; YILMAZLAR, Eray. Understanding the Concept of Microcontroller Based Systems To Choose The Best Hardware For Applications. *Research Inventy: International Journal of Engineering And Science*. 2017, roč. 7, s. 38.

18. *ATmega48A/PA/88A/PA/168A/PA/328/P*. 2020. Dostupné také z: `https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf`.

19. OOMLOUT. *ATmega328P-PU* [online]. 2009-10-09 [cit. 2024-04-12]. Dostupné z: `https://commons.wikimedia.org/wiki/File:ATMEGA328P-PU.jpg`.

20. *Arduino Uno Rev3* [online]. 2024-04-12 [cit. 2024-04-12]. Dostupné z: `https://store.arduino.cc/products/arduino-uno-rev3`.

21. *AVR® Microcontrollers Peripheral Integration: Quick Reference Guide*. 2022. Dostupné také z: `https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/Brochures/AVR-Microcontrollers-Peripheral-Integration-30010135.pdf`.

22. *ATtiny85* [online]. 2023-12-28 [cit. 2023-12-28]. Dostupné z: `https://www.microchip.com/en-us/product/attiny85`.

23. *Tensilica Xtensa Controllers* [online]. 2023-12-12 [cit. 2023-12-12]. Dostupné z: `https://www.cadence.com/en_US/home/tools/ip/tensilica-ip/tensilica-xtensa-controllers-and-extensible-processors.html`.

24. *Memory Types - ESP32* [online]. 2023-12-09 [cit. 2023-12-11]. Dostupné z: `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/memory-types.html`.

25. *ESP32 Series: Datasheet*. Shanghai, China, 2023. Dostupné také z: `https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf`.

26. *ULP Coprocessor Programming - ESP32* [online]. 2023-12-09 [cit. 2023-12-12]. Dostupné z: `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/ulp.html`.

27. KRENT, Brian. *Espressif ESP32 Chip Function Block Diagram* [online]. 2018-09-03 [cit. 2024-04-12]. Dostupné z: https://en.wikipedia.org/wiki/File:Espressif_ESP32_Chip_Function_Block_Diagram.svg.

28. *ESP DevKits* [online]. 2023-12-13 [cit. 2023-12-13]. Dostupné z: https://www.espressif.com/en/products/devkits.

29. *ESP IoT Development Framework* [online]. 2023-12-13 [cit. 2023-12-13]. Dostupné z: https://www.espressif.com/en/products/sdks/esp-idf.

30. BABIUCH, Marek; FOLTYNEK, Petr; SMUTNÝ, Pavel. Using the ESP32 Microcontroller for Data Processing. In: 2019, s. 1–6. Dostupné z DOI: 10.1109/CarpathianCC.2019.8765944.

31. ESPRESSIF SYSTEMS. *ESP32-DevKitC-32E* [online]. 2024-04-12 [cit. 2024-04-12]. Dostupné z: https://cz.mouser.com/ProductDetail/Espressif-Systems/ESP32-DevKitC-32E?qs=GedFDFLaBXFpgD0kAZWDrQ%3D%3D.

32. *ESP Product Selector* [online]. 2023-05-10 [cit. 2023-12-16]. Dostupné z: https://products.espressif.com/#/product-selector.

33. MOONRINTA, Sirawit. *Comparison table for ESP8266/ESP32/ESP32-S2/ESP32-S3/ESP32-C3/ESP32-C6* [online]. 2023-12-17 [cit. 2023-12-17]. Dostupné z: https://gist.github.com/sekcompsci/2bf39e715d5fe47579fa184fa819f421.

34. *ESP32-S3: Technical Reference Manual*. Shanghai, China, 2023. Dostupné také z: https://www.espressif.com/sites/default/files/documentation/esp32-s3_technical_reference_manual_en.pdf.

35. *ESP32-C6: Technical Reference Manual*. Shanghai, China, 2023. Dostupné také z: https://www.espressif.com/sites/default/files/documentation/esp32-c6_technical_reference_manual_en.pdf.

36. ONLOGIC. *What is an Industrial PC?* [Online]. 2023-12-18 [cit. 2023-12-18]. Dostupné z: https://www.onlogic.com/company/io-hub/what-is-an-industrial-pc/.

37. BARNATT, Christopher. *Single Board Computers (SBCs)* [online]. 2023-11-08 [cit. 2023-12-18]. Dostupné z: https://www.explainingcomputers.com/sbc.html.

38. *Raspberry Pi 5 mechanical drawing*. 2023-10. Tech. zpr. Raspberry Pi Ltd. Dostupné také z: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-mechanical-drawing.pdf.

39. *BCM2712: Raspberry Pi Documentation - Processors* [online]. 2023-12-21 [cit. 2023-12-29]. Dostupné z: https://www.raspberrypi.com/documentation/computers/processors.html#bcm2712.

40. *Raspberry Pi Documentation - Raspberry Pi 5* [online]. 2023-12-21 [cit. 2023-12-28]. Dostupné z: https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html.

41. RASPBERRY PI LTD. *Raspberry Pi Documentation: GPIO and the 40-pin header* [online]. 2024-03-20 [cit. 2024-04-12]. Dostupné z: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html.

42. LTD, Raspberry Pi. *Raspberry Pi Zero 2 W* [online]. 2023-12-25 [cit. 2023-12-29]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/.

43. *Raspberry Pi OS* [online]. 2023-12-21 [cit. 2023-12-29]. Dostupné z: https://www.raspberrypi.com/documentation/computers/os.html#introduction.

44. NEOUSYS MARKETING TEAM. *POC-200 BayTrail Ultra-Compact Fanless Rugged Computer* [online]. 2023-12-29 [cit. 2023-12-29]. Dostupné z: https://www.neousys-tech.com/en/product/product-lines/industrial-computers/ultra-compact/poc-200#overview.

45. NEOUSYS TECHNOLOGY. *POC-200: BayTrail Ultra-Compact Fanless Rugged Computer* [online]. 2024-04-12 [cit. 2024-04-12]. Dostupné z: https://www.neousys-tech.com/en/product/product-lines/industrial-computers/ultra-compact/poc-200.

46. *POC-200 Series: User's Manual*. 2014. Dostupné také z: https://www.neousys-tech.com/Resource/Product_Document/POC-200/POC-200-series-User-Manual-RevA1-1.pdf.

47. *Universal asynchronous receiver/transmitter (UART)*. 2006. Dostupné také z: https://www.nxp.com/docs/en/data-sheet/SCC2691.pdf.

48. PEŇA, Eric; LEGASPI, Mary Grace. UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter. *Analog Dialogue*. 2020, roč. 54. Dostupné také z: https://www.analog.com/media/en/analog-dialogue/volume-54/number-4/uart-a-hardware-communication-protocol.pdf.

49. *What is TWI? How to Configure the TWI for I2C Communication*. 2018-01. Tech. zpr. Microchip Technology Inc. Dostupné také z: https://ww1.microchip.com/downloads/en/DeviceDoc/90003181A.pdf.

50. *I^2C-bus specification and user manual*. 2021. Dostupné také z: https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

51. *7-bit 8-bit and 10-bit I2C Slave Addressing* [online]. 2023-11-18 [cit. 2023-11-18]. Dostupné z: https://www.totalphase.com/support/articles/200349176-7-bit-8-bit-and-10-bit-i2c-slave-addressing/.

52. DHAKER, Piyu. Introduction to SPI Interface. *Analog Dialogue*. 2018, roč. 52. Dostupné také z: https://www.analog.com/media/en/analog-dialogue/volume-52/number-3/introduction-to-spi-interface.pdf.

53. *SPI Block Guide*. 2003. Dostupné také z: https://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf.

54. *CAN Specification: Version 2.0*. 1991. Dostupné také z: http://esd.cs.ucr.edu/webres/can20.pdf.

55. GAGE, Douglas W. *Network Protocols for Mobile Robot Systems* [online]. 1997-10 [cit. 2024-01-04]. Dostupné z: https://apps.dtic.mil/sti/tr/pdf/ADA422500.pdf.

56. KAZALA, R.; TANEVA, A.; PETROV, M.; PENKOV, St. Wireless Network for Mobile Robot Applications. *IFAC-PapersOnLine*. 2015, roč. 48, č. 24, s. 231–236. ISSN 2405-8963. Dostupné z DOI: https://doi.org/10.1016/j.ifacol.2015.12.088. 16th IFAC Conference on Technology, Culture and International Stability TECIS 2015.

57. ANDREASSON, Henrik; GRISETTI, Giorgio; STOYANOV, Todor; PRETTO, Alberto. *Sensors for Mobile Robots*. arXiv, 2022. Dostupné z DOI: 10.48550/ARXIV.2206.03223.

58. TZAFESTAS, Spyros G. Mobile Robot Control and Navigation: A Global Overview. *Journal of Intelligent & Robotic Systems*. 2018, roč. 91, č. 1, s. 35–58. ISSN 1573-0409. Dostupné z DOI: 10.1007/s10846-018-0805-9.

59. LEUNG, Keith; LEUNG, Kit; CLARK, Christopher; HUISSOON, J.P. Monocular Vision based Particle Filter Localization in Urban Environments. 2007. Dostupné také z: https://www.researchgate.net/publication/253172719_Monocular_Vision_based_Particle_Filter_Localization_in_Urban_Environments.

60. WANG, Jiefei; HERATH, Damith. How to Move? Control, Navigation and Path Planning for Mobile Robots. In: *Foundations of Robotics: A Multidisciplinary Approach with Python and ROS*. Ed. HERATH, Damith; ST-ONGE, David. Singapore: Springer Nature Singapore, 2022, s. 205–238. ISBN 978-981-19-1983-1. Dostupné z DOI: 10.1007/978-981-19-1983-1_8.

61. ZOHAIB, Mohammad; PASHA, Mustafa; RIAZ, Raja Ali; JAVAID, Nadeem; ILAHI, M.; KHAN, Rahim. Control Strategies for Mobile Robot With Obstacle Avoidance. 2013, roč. 3, s. 1027–1036. Dostupné také z: https://www.researchgate.net/publication/237054021_Control_Strategies_for_Mobile_Robot_With_Obstacle_Avoidance.

62. ZOHAIB, Mohammad; PASHA, Mustafa; JAVAID, Nadeem; IQBAL, Jamshed. IBA: Intelligent Bug Algorithm – A Novel Strategy to Navigate Mobile Robots Autonomously. In: 2014, sv. 414, s. 291–299. ISBN 978-3-319-10987-9. Dostupné z DOI: 10.1007/978-3-319-10987-9_27.

63. MORIN, Pascal; SAMSON, Claude. Motion Control of Wheeled Mobile Robots. In: *Springer Handbook of Robotics*. Ed. SICILIANO, Bruno; KHATIB, Oussama. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, s. 799–826. ISBN 978-3-540-30301-5. Dostupné z DOI: 10.1007/978-3-540-30301-5_35.

64. *IG42-DB4: 4WD All Terrain Heavy Duty Robot Platform* [online]. 2022-03-17 [cit. 2024-01-19]. Dostupné z: https://www.superdroidrobots.com/store/product=1477.

65. BAE, Beomsu; LEE, Dong-Hyun. Design of a Four-Wheel Steering Mobile Robot Platform and Adaptive Steering Control for Manual Operation. *Electronics*. 2023, roč. 12, č. 16. ISSN 2079-9292. Dostupné z DOI: 10.3390/electronics12163511.

66. IŠA, Jiří. *Freyja* [online]. 2024-01-21 [cit. 2024-01-21]. Dostupné z: https://robotika.cz/robots/freyja/cs.

67. EUROPEAN ROBOTICS. *ENRICH: The European Robotics Hackathon* [online]. 2023-06-16 [cit. 2024-01-23]. Dostupné z: https://enrich.european-robotics.eu/.

68. DEFENSE ADVANCED RESEARCH PROJECTS AGENCY (DARPA). *DARPA Triage Challenge* [online]. 2024-01-18 [cit. 2024-01-23]. Dostupné z: https://triagechallenge.darpa.mil/.

69. FDATA CO., Ltd. *Outdoor mobile robot platform A012 Omnidirectional chassis with payload 50kg* [online]. 2024-01-21 [cit. 2024-01-21]. Dostupné z: https://www.fdatarobot.com/products/outdoor-mobile-robot-platform-a012-omnidirectional-chassis-with-payload-50kg.html.

70. AGILEX ROBOTICS TEAM. *Hunter SE: User Manual*. 2021. Dostupné také z: https://static.generation-robots.com/media/user-manual-agilex-robotics-hunter-se-en.pdf.

71. AGILEX ROBOTICS TEAM. *Scout 2.0: User Manual*. 2021. Dostupné také z: https://static.generation-robots.com/media/user-manual-agilex-robotics-scout2.0-en.pdf.

72. ROBOTNIK AUTOMATION S.L. *SUMMIT-XL STEEL* [online]. 2024-01-22 [cit. 2024-01-22]. Dostupné z: https://robotnik.eu/products/mobile-robots/summit-xl-steel-en/.

73. FOX, C.W.; EVANS, M.H.; LEPORA, N.F.; PEARSON, M.J.; HAM, A.; PRESCOTT, T.J. *12th TAROS: Conference Towards Autonomous Robotic Systems*. CrunchBot: a mobile whiskered robot platform. Ed. GROSS, R.; ALBOUL, L.; MELHUISH, C.; WITKOWSKI, M.; PRESCOTT, T.J.; PENDERS, J. Springer, 2011. Lecture Notes in Computer Science, č. 6856. Dostupné také z: https://eprints.whiterose.ac.uk/155179/. © 2011 Springer-Verlag. This is an author-produced version of a paper subsequently published in LNCS. Uploaded in accordance with the publisher's self-archiving policy.

74. BARROS, Rhaian; FILHO, Jorge; NETO, Joao; NASCIMENTO, Tiago. An Open-Design Warehouse Mobile Robot. In: 2020, s. 1–6. Dostupné z DOI: 10.1109/LARS/SBR/WRE51543.2020.9307137.

75. SILVER_A. *Remote Controlled 6WD All Terrain Robot: 10 Steps (with Pictures)* [online]. 2024-01-22 [cit. 2024-01-22]. Dostupné z: https://www.instructables.com/Remote-Controlled-6WD-All-Terrain-Robot/.

76. SILVER_A. *4WD All Terrain Arduino Robot for Everyone* [online]. 2024-01-22 [cit. 2024-01-22]. Dostupné z: https://www.instructables.com/4WD-All-Terrain-Arduino-Robot-for-Everyone/.

77. *Kombi stojka 30x30 - ELOX: Hliníkové profily standardní* [online]. 2024-04-08 [cit. 2024-04-08]. Dostupné z: https://www.alunet.cz/index.php?route=product/product&product_id=1203.

78. NOVÁK, Petr. Dimenzování motoru. In: *Mobilní roboty*. 1. vydání. Praha: BEN - technická literatura, 2005, s. 79–88.

79. *Gears - Gear Efficiency* [online]. 2022-07-26 [cit. 2024-04-09]. Dostupné z: https://roymech.org/Useful_Tables/Drive/Gear_Efficiency.html.

80. BERÁNEK, Karel. *Modifikace 3D tiskárny Prusa i3 MK3 na uzavřený koncept s dvojnásobnou tiskovou plochou* [online]. České Budějovice, 2019 [cit. 2023-01-06]. Dostupné z: https://dspace.jcu.cz/bitstream/handle/20.500.14390/42734/BP-kabe.pdf?sequence=1. Bak. pr. Ústav Fyziky, Přírodovědecká fakulta, Jihočeská univerzita v Českých Budějovicích. Vedoucí práce Jakub GEYER.

81. SPARKFUN ELECTRONICS. *TFMini Plus - Micro LiDAR Module - SEN-15179* [online]. 2021-03-09 [cit. 2024-04-12]. Dostupné z: https://www.sparkfun.com/products/15179.

82. BENEWAKE (BEIJING) CO. LTD. *Product Manual of TFmini Plus: TFmini Plus LiDAR Module*. [B.r.]. Dostupné také z: https://cdn.sparkfun.com/assets/1/4/2/1/9/TFmini_Plus_A02_Product_Manual_EN.pdf.

83. *Kamera OdSeven HD OV5647 5Mpx* [online]. 2030-11-05 [cit. 2024-04-11]. Dostupné z: https://botland.cz/fotoaparaty-pro-raspberry-pi/5619-kamera-odseven-hd-ov5647-5mpx-pro-raspberry-pi-5904422333850.html.

84. *UNO R3* [online]. 2023-03-10 [cit. 2023-03-11]. Dostupné z: https://docs.arduino.cc/hardware/uno-rev3.

85. DUNKELS, Adam; SCHMIDT, Oliver; VOIGT, Thiemo; ALI, Muneeb. Protothreads: simplifying event-driven programming of memory-constrained embedded systems. In: *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*. Boulder, Colorado, USA: Association for Computing Machinery, 2006, s. 29–42. SenSys '06. ISBN 1595933433. Dostupné z DOI: 10.1145/1182807.1182811.

86. ŠVARC, Jakub. *Simple Reliable Serial Communication protocol (SRSC)*. 2023. Dostupné také z: https://codeberg.org/atlas144/srsc.

87. ŠVARC, Jakub. *SRSC Java*. 2024. Dostupné také z: https://codeberg.org/atlas144/srsc-java.

88. ŠVARC, Jakub. *SRSC Arduino*. 2024. Dostupné také z: https://codeberg.org/atlas144/srsc-arduino.

89. GEERLING, Jeff. *Power Consumption Benchmarks* [online]. 2021-10-30 [cit. 2024-04-04]. Dostupné z: https://www.pidramble.com/wiki/benchmarks/power-consumption.

90. ŠVARC, Jakub. *mals-mcu* [online]. 2023-03-18 [cit. 2023-03-18]. Dostupné z: https://codeberg.org/atlas144/mals-mcu.

91. CODEBERG.ORG. *Codeberg.org* [online]. 2023-03-18 [cit. 2023-03-18]. Dostupné z: https://codeberg.org/about.

92. *The MIT License* [online]. [B.r.] [cit. 2024-04-26]. Dostupné z: https://opensource.org/license/mit.

93. ŠVARC, Jakub. *mals-intermodule-broker* [online]. 2023-03-18 [cit. 2023-03-18]. Dostupné z: https://codeberg.org/atlas144/mals-intermodule-broker.

94. *GNU Lesser General Public License* [online]. 1999. Ver. 2.1 [cit. 2023-03-18]. Dostupné z: https://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html.

95. EUGSTER, Patrick; FELBER, Pascal; GUERRAOUI, Rachid; KERMARREC, Anne-Marie. The Many Faces of Publish/Subscribe. *ACM Comput. Surv.* 2003, roč. 35, s. 114–131. Dostupné z DOI: 10.1145/857076.857078.

96. *Arduino Sketches* [online]. 2023-03-15 [cit. 2023-03-19]. Dostupné z: https://docs.arduino.cc/learn/programming/sketches#setup-and-loop.

97. DE LA BRIANDAIS, Rene. File Searching Using Variable Length Keys. In: *Papers Presented at the the March 3-5, 1959, Western Joint Computer Conference*. San Francisco, California: Association for Computing Machinery, 1959, s. 295–298. IRE-AIEE-ACM '59 (Western). ISBN 9781450378659. Dostupné z DOI: 10.1145/1457838.1457895.

98. THE PI4J PROJECT. *Pi4J* [online]. 2024-03-27 [cit. 2024-03-27]. Dostupné z: https://pi4j.com/.

99. MELNIKOV, Alexey; FETTE, Ian. *The WebSocket Protocol* [RFC 6455]. RFC Editor, 2011. Request for Comments, č. 6455. Dostupné z DOI: 10.17487/RFC6455.

100. TOOTALLNATE. *Java-WebSocket* [online]. 2020-03-16 [cit. 2023-03-19]. Dostupné z: https://tootallnate.github.io/Java-WebSocket/.

101. ŠVARC, Jakub. *mals-client* [online]. 2024-03-18 [cit. 2024-03-18]. Dostupné z: https://codeberg.org/atlas144/mals-client.

102. ENGINEERKID1. *OpenCV Based Line Following Robot* [online]. 2024-04-06 [cit. 2024-04-06]. Dostupné z: https://www.instructables.com/OpenCV-Based-Line-Following-Robot/.

103. ABD HAMID, Mohamad hanif; ADOM, Abdul; ABDUL RAHIM, Norasmadi; FAZALUL RAHIMAN, Mohd Hafiz. Navigation of mobile robot using Global Positioning System (GPS) and obstacle avoidance system with commanded loop daisy chaining application method. In: 2009, s. 176–181. Dostupné z DOI: 10.1109/CSPA.2009.5069211.

104. LOOSE, Harald; ZASEPA, Marcin; PIERZCHALA, Przemyslaw; RITTER, Robert. GPS Based Navigation of an Autonomous Mobile Vehicle. *Solid State Phenomena*. 2009, roč. 147-149, s. 55–60. Dostupné z DOI: 10.4028/www.scientific.net/SSP.147-149.55.

105. NAKAMURA, Ryota; KOBAYASHI, Kazuyuki. A Remote Control System for Waypoint Navigation Based Mobile Robot Using JAUS. In: *2018 57th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. 2018, s. 1751–1756. Dostupné z DOI: 10.23919/SICE.2018.8492554.

106. BORENSTEIN, Johann; FENG, L. UMBmark : a method for measuring, comparing, and correcting dead-reckoning errors in mobile robots. In: 1994. Dostupné také z: https://api.semanticscholar.org/CorpusID:56599256.

107. WONDOSEN, Assefinew; SHIFERAW, Dereje. *Fuzzy Logic Controller Design for Mobile Robot Outdoor Navigation*. 2024. Dostupné z arXiv: 2401.01756 [cs.RO].

108. CINGOLANI, Pablo; ALCALÁ-FDEZ, Jesús. jFuzzyLogic: a Java Library to Design Fuzzy Logic Controllers According to the Standard for Fuzzy Control Programming. 2013, s. 61–75.

109. CINGOLANI, Pablo; ALCALA-FDEZ, Jesus. jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation. In: *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*. 2012, s. 1–8.

Seznam obrázků

2.1 Příklad pevného, orientovatelného centrického a orientovatelného mimostředného kola *(zdroj: převzato z [10] a [11]; upraveno)* 10
2.2 Příklad švédského kola, poly-wheel a sférického kola *(zdroj: převzato z [14], [15] a [16]; upraveno)* 11
2.3 ATmega328 *(zdroj: převzato z [19])* 13
2.4 Arduino UNO Rev3 *(zdroj: převzato z [20])* 13
2.5 Blokové schéma mikrokontroléru ESP32 *(zdroj: převzato z [27])* . . . 14
2.6 Vývojová deska ESP32-DevKitC-32E *(zdroj: převzato z [31])* 15
2.7 40 pinové GPIO rozhraní *(zdroj: převzato z [41])* 17
2.8 POC-200 *(zdroj: převzato z [45])* 18
2.9 Robot IG42-DB4 *(zdroj: převzato z [64])* 31
2.10 Akademický robot z Kumoh National Institute of Technology *(zdroj: převzato z [65])* 32
2.11 Robot Freyja *(zdroj: převzato z [66])* 33

3.1 Průřez Kombi profilu *(zdroj: vlastní)* 36
3.2 Úvodní návrh modelu *(zdroj: vlastní)* 37
3.3 Model po odebrání vnitřní platformy *(zdroj: vlastní)* 38
3.4 Finální návrh modelu *(zdroj: vlastní)* 39
3.5 Kryty náboje *(zdroj: vlastní)* 40
3.6 Model pohonu jednoho kola *(zdroj: vlastní)* 41
3.7 Diagram napájení komponent *(zdroj: vlastní)* 42
3.8 Šablony pro přípravu vrtání *(zdroj: vlastní)* 43
3.9 Uchycení hmoždinek v profilu *(zdroj: vlastní)* 44
3.10 Výsledná konstrukce *(zdroj: vlastní)* 44
3.11 LiDAR TFMini-Plus *(zdroj: převzato z [81])* 45
3.12 Schéma zapojení LiDARu *(zdroj: vlastní)* 46
3.13 Deska plošných spojů a pouzdro LiDARu *(zdroj: vlastní)* 47
3.14 Deska plošných spojů sonarové soustavy *(zdroj: vlastní)* 48
3.15 Pouzdro sonarové soustavy *(zdroj: vlastní)* 49
3.16 Detail DPS kontaktního senzoru *(zdroj: vlastní)* 49
3.17 Kamera pro sledování linky *(zdroj: [83])* 50
3.18 Schéma zapojení klopného obvodu (vlevo) a demultiplexoru (vpravo) *(zdroj: vlastní)* 53
3.19 Deska plošných spojů řídicí jednotky subsystémů *(zdroj: vlastní)* . . . 54

3.20 Struktura paketu *(zdroj: vlastní)* 55
3.21 Obrazovka manuálního řízení *(zdroj: vlastní)* 66
3.22 Obrazovka řízení sledováním linky *(zdroj: vlastní)* 66
3.23 Obrazovka autonomní jízdy podle souřadnic *(zdroj: vlastní)* 67
3.24 Obrazovka monitoringu *(zdroj: vlastní)* 69
3.25 Obrazovka nastavení *(zdroj: vlastní)* 69
3.26 Obrazovka informací o aplikaci *(zdroj: vlastní)* 70
3.27 Funkce příslušnosti pro vzdálenost od dílčího cíle *(zdroj: vlastní)* . . . 74
3.28 Funkce příslušnosti pro odchylku natočení *(zdroj: vlastní)* 74
3.29 Funkce příslušnosti pro vzdálenost od překážky *(zdroj: vlastní)* 75
3.30 Funkce příslušnosti pro střídy PWM signálu motorů *(zdroj: vlastní)* . 75

Seznam tabulek

2.1 Porovnání vlastností mikrokontrolérů ESP [32][33][34][35] 16
2.2 Porovnání vybraných KRP 35

3.1 Parametry motorů PG420-24-49-BE 39
3.2 Výpočty převodového poměru (tučně jsou zvýrazněny měněné parametry); hodnoty μ a θ jsou převzaty z [78] a η_P z [79] 41
3.3 Seznam rezervovaných typů paketů 56
3.4 Metody SRSC rozhraní 57
3.5 Seznam propojení řídicí jednotky 59

Seznam fragmentů kódu

3.1 Příklad odchoží WebSocket zprávy 63
3.2 Příklad příchozí WebSocket zprávy typu SUBSCRIBE 63
3.3 Příklad definice senzoru 68
3.4 Ukázka definice funkcí příslušnosti a fuzzy pravidel v jazyce FCL . . 73

Seznam příloh

Přílohy jsou umístěny v doprovodném archivu *Svarc_Jakub_2024_BP_prilohy.zip*. Jedná se o:

- Zdrojové kódy řídicí aplikace, klientské aplikace, firmwaru senzorů a vlastních knihoven.
- Schémata elektroniky a návrhy desek plošných spojů.
- CAD soubory a technické výkresy.
- Fotografie výsledného řešení a klientské aplikace.
- Uživatelskou dokumentaci.