

Aplikace pro řízení robotického auta A Robotic Car Driving Application

Jan Andrysek

Bakalářská práce
2023



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Jan Andryšek
Osobní číslo: A20056
Studijní program: B0613A140020 Softwarové inženýrství
Forma studia: Kombinovaná
Téma práce: Aplikace pro řízení robotického auta
Téma práce anglicky: A Robotic Car Driving Application

Zásady pro vypracování

1. Popište robotické auto PiCar-4WD a možnosti jeho řízení.
2. Charakterizujte Raspberry Pi a jeho propojení s robotickým autem.
3. Navrhněte a naprogramujte mobilní řídicí aplikaci pro řízení robotického auta.
4. Aplikace bude mít grafické rozhraní pro sestavení řídicích příkazů.
5. Koncepce aplikace bude navržena tak, aby bylo možné celý systém použít pro propagaci logického programování.

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

Seznam doporučené literatury:

1. Sunfounder: piCar-4wd Car Kit for Raspberry Pi [online]. Shenzhen: © SunFounder, 2022 [cit. 2022-11-24]. Dostupné z: <https://docs.sunfounder.com/projects/picar-4wd/en/latest/>
2. Raspberrypi: raspberry Pi OS: datasheet: configuration: the config.txt file : documentation [online]. Cambridge: ©Raspberrypi, 2022 [cit. 2022-11-21]. Dostupné z: <https://www.raspberrypi.com>
3. Android Developers: develop android games: android studio: docs [online]. Mountain View: Mountain View, 2022 [cit. 2022-11-24]. Dostupné z: <https://developer.android.com>
4. Blockly: samples: references: guides [online]. Mountain View: Mountain View, 2022 [cit. 2022-11-24]. Dostupné z: <https://developers.google.com/blockly>
5. HORTON, John. Android Programming for Beginners. 3rd edition. Birmingham: ©Packt Publishing, 2021. ISBN 9781800563438.

Vedoucí bakalářské práce: **Ing. Pavel Pokorný, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**



doc. Ing. Jiří Vojtěšek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 24.5.2023

.....
podpis studenta

ABSTRAKT

Tato bakalářská práce představuje vývoj webové aplikace, která umožňuje uživatelům vizuálně programovat model auta RpiCar-4WD. Aplikace je navržena tak, aby mohla být hostována na počítači Raspberry Pi 4B, který je součástí modelu auta. Tento počítač zprostředkovává komunikaci mezi webovou aplikací a periferiemi modelu auta, což umožňuje uživatelům vytvářet a spouštět programy pro řízení modelu auta a interakci s prostředím. Aplikace je navržena tak, aby umožňovala programování pomocí bloků reprezentujících jednotlivé akce, což usnadňuje uživatelům tvorbu komplexních programů. Tato práce je zaměřena na propagaci logického myšlení a programování na akcích pořádaných technologickou firmou ZF Engineering, a je určena pro nadšence v oblasti robotiky a vývojáře, kteří se zajímají o autonomní vozidla a vizuální programování.

Klíčová slova: Blockly, SunFounder PiCar-4WD, Raspberry Pi 4B, JavaScript, webový server.

ABSTRACT

This bachelor's thesis presents the development of a web application that allows users to visually program the RpiCar-4WD model car. The application is designed to be hosted on a Raspberry Pi 4B computer, which is part of the car model. This computer facilitates the communication between the web application and the peripherals of the car model, enabling users to create and run programs for controlling the car model and interacting with the environment. The application is designed to allow programming using blocks representing individual actions, which simplifies the creation of complex programs for users. This work is focused on promoting logical thinking and programming at events organized by the technology company ZF Engineering and is intended for enthusiasts in the field of robotics and developers interested in autonomous vehicles and visual programming.

Keywords: Blockly, SunFounder PiCar-4WD, Raspberry Pi 4B, JavaScript, web server.

Na tomto místě bych rád upřímně poděkoval panu Ing. Pavlu Pokornému, Ph.D., za jeho odborné vedení, cenné rady a neustálou podporu během mého studia a při vypracování této bakalářské práce. Jeho odbornost, zkušenosti a trpělivost mi byly neocenitelným zdrojem inspirace a motivace.

Poděkování patří také firmě ZF Engineering s.r.o. za zakoupení a poskytnutí vybavení nezbytného pro tento projekt.

Dále bych chtěl poděkovat všem členům akademického týmu a spolužákům za spolupráci a podnětné diskuse, které mi pomohly prohloubit mé znalosti a dovednosti v oboru.

Taktéž bych rád vyjádřil svou vděčnost rodině a přátelům za jejich podporu, trpělivost a pochopení během celé doby mého studia.

V neposlední řadě děkuji všem, kteří se přímo či nepřímo podíleli na úspěšném dokončení mé bakalářské práce a kteří přispěli svým časem, znalostmi nebo materiálním zázemím.

Bez Vaší pomoci bych toho nemohl dosáhnout. Děkuji.

OBSAH

ÚVOD	8
I TEORETICKÁ ČÁST	9
1 ROBOTICKÉ AUTO PICAR – 4WD	10
1.1 OBECNÉ INFORMACE	10
1.1.1 Sunfounder	11
1.2 STAVEBNICE MODELU AUTA.....	11
1.3 MOŽNOSTI ŘÍZENÍ.....	12
1.3.1 Originální aplikační řešení	13
1.3.2 Možnosti rozšíření.....	14
1.3.2.1 Aplikace pro chytré telefony.....	14
1.3.2.2 Integrace herního ovladače	14
1.3.2.3 Infračervené dálkové ovládání.....	15
1.3.2.4 Hlasové ovládání.....	15
1.3.2.5 Vizuální programování pomocí Blockly.....	15
2 RASPBERRY PI	16
2.1 ARCHITEKTURA ARM.....	16
2.2 RASPBERRY PI 4B	16
2.2.1 Základní parametry	17
2.2.2 Konektivita čipu BCM2711	18
2.3 PROPOJENÍ RPI A 4WD HAT	19
2.3.1 HAT obecně	19
2.3.2 4WD HAT	19
2.3.3 Mechanické propojení	20
2.3.4 Elektrické propojení a komunikace.....	21
2.3.4.1 GPIO	22
2.3.4.2 Sériová sběrnice I ² C (I2C).....	22
2.3.4.3 Knihovna smbus	24
3 NÁSTROJE PRO TVORBU MOBILNÍCH APLIKACÍ	25
3.1 STĚŽEJNÍ KNIHOVNY	25
3.1.1 Blockly	26
3.1.2 JS-interpreter	26
3.2 ANDROID STUDIO.....	27
II PRAKTICKÁ ČÁST	28
4 SESTAVENÍ ROBOTY	29
5 ÚPRAVY HARDWARU	30
5.1 NAPÁJECÍ SOUSTAVA.....	30
5.1.1 Řešení nabíjení	31
6 NÁVRH APLIKACE	35
6.1 POŽADAVKY NA PROJEKT	35
7 REALIZACE APLIKACE	37
7.1 ANALÝZA PYTHON SKRIPTŮ	37
7.1.1 Podrobnější rozbor	38

7.1.1.1	setup.py	38
7.1.1.2	start.py.....	38
7.1.1.3	web_server.py	39
7.1.1.4	Součásti knihovny picar_4wd.....	40
7.2	WEBOVÁ APLIKACE.....	42
7.2.1	Představení vizuální stránky aplikace	42
7.2.2	Komunikace se serverem	45
7.2.2.1	requireSocket.js.....	46
7.2.2.2	responseSocket.js	46
7.2.2.3	manual.js	47
7.2.3	Implementace grafického rozhraní pro programování	47
7.2.3.1	Definice bloků.....	50
7.2.3.2	Generování a spuštění kódu.....	52
8	SOUČASNÝ STAV, VYUŽITÍ A PLÁNY DO BUDOUCNA	57
8.1	SOUČASNÝ STAV	57
8.1.1	Ukázka programu	57
8.1.2	Toolbox	58
8.1.3	Hlídaní spojení a asistence při skladbě bloků	59
8.2	VYUŽITÍ APLIKACE V PROPAGACI.....	61
8.3	NÁVRHY NA VYLEPŠENÍ	61
	ZÁVĚR	62
	SEZNAM POUŽITÉ LITERATURY.....	63
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	66
	SEZNAM OBRÁZKŮ	69
	SEZNAM KÓDŮ	70
	SEZNAM TABULEK.....	71
	SEZNAM PŘÍLOH.....	72

ÚVOD

Dlouhodobý nedostatek IT specialistů na trhu práce postihuje mnoho subjektů, včetně firmy ZF Engineering Plzeň s.r.o.¹ Tato bakalářská práce, která vznikla ve spolupráci s touto firmou, se snaží tuto situaci zlepšit. Zaměřuje se na rozvoj logického myšlení, které je klíčovým předpokladem pro všechny vědecké a technické obory. Cílem je vytvořit aplikaci s grafickým rozhraním, které umožňuje uživatelům sestavit vlastní funkční aplikaci pomocí programových bloků. Takto sestavenou aplikaci lze poté spustit a využít pro řízení robotického modelu auta

V práci je využita stavebnice a zdrojové kódy od výrobce Sunfounder [1], který se specializuje na poskytování cenově dostupných komponentů a nástrojů pro elektroniku, DIY projekty a internet věcí (*IoT*). Po několika hardwarových úpravách se tato sada stala ideální pro naplnění požadavků našeho projektu.

Cílem této práce je vytvořit aplikaci implementující rozhraní grafického programování. Uživatelé tak mohou pomocí graficky odlišných, do sebe zapadajících bloků sestavit funkční program, který po spuštění řídí dálkově ovládaný model auta. Tato práce představuje potenciál robotiky a programování a zároveň tato pole propaguje mezi širokou veřejností. Aplikace je navržena tak, aby byla intuitivní a jednoduchá na ovládání, s cílem dovést i naprosto začátečníka k funkčnímu výsledku.

Vývoj webových mobilních aplikací prudce pokračuje a s každým rokem neustále narůstá nejen jejich počet, ale také možnosti a výkon, které nabízí. Chytrý mobilní telefon se stal nezbytným prostředníkem mezi lidmi a různorodými technologiemi, které jsou implementovány v různých aplikacích pro práci, vzdělání, zdraví nebo zábavu. Jako ideální forma aplikace byla zvolena responzivní webová aplikace, která je především optimalizována pro mobilní zařízení, jako jsou smartphony a tablety. Tento typ aplikace je nejvhodnější pro propagační účely a umožňuje snadný přístup a použití na mobilních zařízeních.

¹ <https://www.zf.com/czech/cs/home/home.html>

I. TEORETICKÁ ČÁST

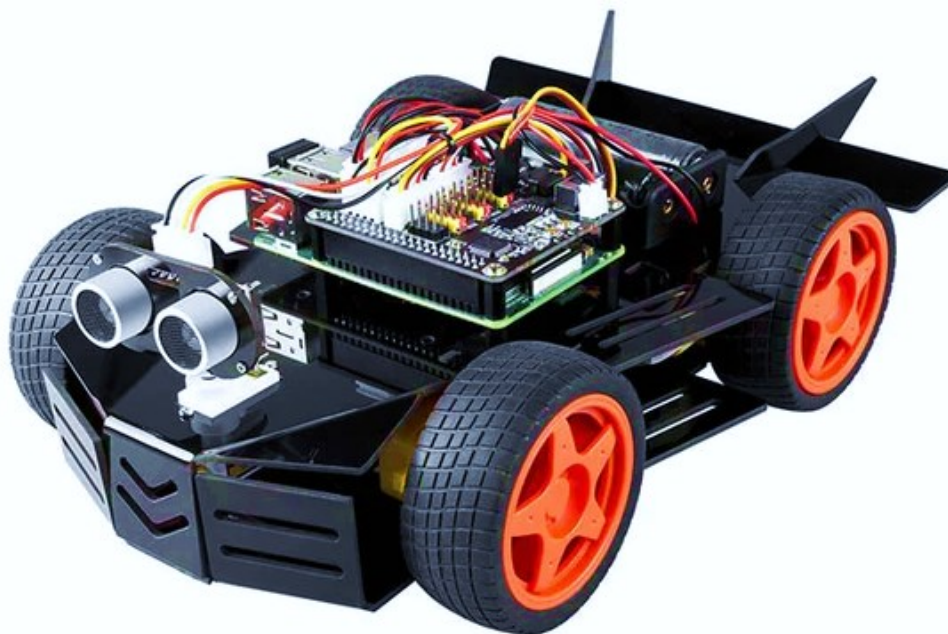
1 ROBOTICKÉ AUTO PICAR – 4WD

Tato kapitola popisuje stavebnici PiCar-4WD i jejího výrobce Sunfounder, její obsah, možnosti řízení a také software dodaný spolu se stavebnicí.

1.1 Obecné informace

SunFounder PiCar-4WD (obrázek 1), je malý model automobilu, který je cenově dostupný a kompatibilní s Raspberry Pi3 modelem B/B+ a Raspberry Pi 4 modelem B (zkráceně RPi), který ale není součástí stavebnice. Model auta je vybaven čtyřmi nezávisle poháněnými koly a servomotorem. Dále obsahuje rozšiřující modul, který zvyšuje počet vstupů a výstupů jednodeskového počítače Raspberry Pi. Je vybaven senzory, které robotu umožňují získávat informace o svém okolí. Je to ultrazvukový měřič vzdálenosti, 3 infračervené senzory umožňující snímat odstíny šedé a dvěma optickými senzory otáček umístěny na zadních kolech modelu. Robot může být ovládán aplikací, která je dodávána se stavebnicí. Tato aplikace slouží jako dálkové ovládání a je jí také možné použít ke spouštění několika autonomních režimů.[1]

Tento robot je vhodný pro zájemce o automatizaci a robotiku a může být také použit pro výuku programování a řízení robotů. PiCar-4WD lze naprogramovat k provádění různých úkolů a pohybů, což uživatelům poskytuje platformu pro experimentování a zkoumání světa robotiky.



Obrázek 1 Raspberry Pi Smart Car Kit (PiCar-4WD) [2]

1.1.1 Sunfounder

Společnost SunFounder, založená v roce 2012, se zaměřuje na vývoj a výrobu elektronických součástek, robotických stavebnic a dalších výukových produktů v oblasti STEM (věda, technologie, inženýrství a matematika). Hlavním cílem této společnosti je podporovat začátečníky, studenty a nadšence v oboru elektroniky a robotiky tím, že jim poskytne dostupné a snadno použitelné nástroje a materiály.

Se sídlem v Hongkongu, je společností specializující se na vývoj a výrobu elektronických projektů i výukových materiálů pro programování, robotiku a internet věcí (IoT). Nabízí širokou škálu stavebnic pro začátečníky i pokročilé uživatele, včetně kitů pro stavbu robotů, senzorických modulů, desek s mikrokontrolery a dalších komponentů pro vytváření interaktivních projektů. Vedle produkce se také věnuje edukaci a poskytuje školení v oblasti programování, robotiky a IoT. Jejich produkty jsou oblíbené zejména v akademickém prostředí, na středních a základních školách, ale také mezi hobbyisty a tvůrci DIY projektů. Díky široké nabídce svých produktů si společnost SunFounder získala důvěru mnoha uživatelů po celém světě. [18]

1.2 Stavebnice modelu auta

Stavebnice modelu auta s pohonem všech kol obsahuje komponenty jako je podvozek, stejnosměrné motory, kola, řídicí desky a další nezbytné díly a spojovací materiál pro stavbu modelu auta. Tyto komponenty (obr. 2) se skládají do jednoho celku. [5]

Sestavení modelu není příliš složité, takže ho zvládne každý, kdo má základní mechanické a elektronické dovednosti. Stavebnice obsahuje všechny potřebné stavební součásti (obrázek 2), elektronické díly a nářadí v podobě šroubováku. Neobsahuje však jednodeskový počítač Raspberry Pi, micro SD kartu a dva lithium-iontové články 18650 potřebné k napájení počítače a především čtyř stejnosměrných motorů. Tento typ článků je doporučen výrobcem. Motory tvoří většinu celkové spotřeby energie. [5]

V přílohách P2 až P4 jsou obrázky detailu všech součástí stavebnice [5]. Zde následuje jejich výčet [3]:

- 4WD HAT modul / 4WD HAT Module x 1
- 2-kanálový foto-snímač (modul měření rychlosti) / 2-ch Photo-interrupter Module (Velocity Measurement Module) x 1
- 3-kanálový senzor odstínu šedi / 3-ch Grayscale Sensor Module x 1
- 3-kanálový senzor odstínu šedi / 3-ch Grayscale Sensor Module x 1

některé další možnosti rozšíření funkčnosti, jelikož samotný počítač Rpi i 4WD HAT nabízí velké možnosti pro další rozšíření. Samotná přítomnost RPi dělá tuto stavebnici velmi silnou, co se možností jejího rozšíření týče. Dále budou zmíněny některé tyto možnosti.

1.3.1 Originální aplikační řešení

Model je možno řídit originální aplikací dodávanou výrobcem anebo také existuje možnost spuštění Python skriptů, které po dobu spuštění převzou kontrolu nad modelem. Tyto skripty je také možné spustit z příkazové řádky přímo v konzole RPi.

Součástí je také webová aplikace, která představuje nejjednodušší a praktické řešení jakékoliv interakce s modelem auta. Zdrojové kódy [16] jsou volně přístupné a je možné je využít pro jakékoliv projekty, samozřejmě bez záruky. Ty jsou velmi jednoduché, ale ne vždy fungují dle očekávání. Nicméně jako základní kostra pro kreativní tvorbu jsou naprosto dostačující.

Tato aplikace umí, kromě klasického dálkového ovládní pomocí šipek, spustit Python skripty s jejichž pomocí lze auto řídit. Aplikace (obr. 3) se skládá ze 3 obrazovek a to uvítací, hlavní a nastavení.



Obrázek 3 Hlavní stránka originální aplikace SunFounder [16]

Webový server, vytvořený s využitím Pythonu a jeho standardních knihoven (*http.server*), představuje jednoduchý, ale efektivní nástroj pro ovládání hardwaru. Tento konkrétní server umožňuje interaktivní ovládání modelu auta prostřednictvím webového rozhraní.

Výhody:

- Jeho univerzální design zajišťuje širokou dostupnost na různých platformách
- Není třeba instalovat další software do zařízení uživatele.
- Jednoduchá rozšiřitelnost funkcionality.

Nevýhody:

- Potenciální problémy s latencí kvůli závislosti na připojení Wi-Fi.
- Omezené možnosti přizpůsobení ve srovnání s nativními aplikacemi.

1.3.2 Možnosti rozšíření

Jelikož je ve stavebnici použit počítač RPi, existuje řada způsobů, jakými lze ovládat model auta Sunfounder. S ohledem na budoucí rozvoj a adaptabilitu projektu, je důležité zohlednit i další potenciální možnosti rozšíření. Tyto alternativy mohou poskytnout cenné perspektivy pro budoucí iterace projektu, přizpůsobení se změnám v technologii nebo rozšíření funkcionality modelu auta Sunfounder.

1.3.2.1 Aplikace pro chytré telefony

Vývoj vlastní aplikace pro chytré telefony pro platformy Android nebo iOS, která umožňuje ovládání automobilu prostřednictvím Bluetooth nebo Wi-Fi.

1. Výhody

- Přizpůsobitelné rozhraní s funkcemi specifickými pro danou platformu.
- Může pracovat s připojením Bluetooth i Wi-Fi.

2. Nevýhody

- Potřeba samostatného vývoje pro platformy Android a iOS.
- Vyžaduje instalaci aplikace v zařízení uživatele.

1.3.2.2 Integrace herního ovladače

Bezdrátové ovládání modelu auta pomocí herního ovladače, například ovladače PlayStation nebo Xbox, připojením k Raspberry Pi pomocí Bluetooth nebo kabelu USB.

1. Výhody:

- Je to efektivní způsob ovládání.
- Možnost využití analogových joysticků.

2. Nevýhody:

- Pouze ovládání pohybu.

1.3.2.3 Infračervené dálkové ovládání

V úvahu přichází přidání infračerveného (IR) přijímače k modelu auta a použití standardního IR dálkového ovladače k odesílání příkazů.

1. Výhody:

- Jednoduché řešení bezdrátového přenosu.

2. Nevýhody:

- Pouze ovládání pohybu.
- Prakticky nepříliš dobře využitelné.

1.3.2.4 Hlasové ovládání

Integrace knihoven nebo služeb pro rozpoznávání hlasu, jako je Google Assistant nebo Amazon Alexa, aby bylo možno model automobilu ovládat hlasem.

1. Výhody:

- Implementace a zpracování hlasu můžou výborně posloužit pro výuku.

2. Nevýhody:

- Nevhodné do hlučných prostor.
- Nutnost kvalitního mikrofону, který není součástí stavebnice

1.3.2.5 Vizualní programování pomocí Blockly

Začlenění vizuálního programovacího prostředí, jako je knihovna Blockly (popis viz kapitola 3.1.1), do ovládání modelu vozu, které umožňuje uživatelům vytvářet vlastní programy uspořádáním bloků představujících různé funkce a logické konstrukce.[12]

1. Výhody:

- Implementace a může výborně posloužit pro výuku.
- Usnadnění už i tak poměrně komplexní problematiky ovládání.
- Učení formou hry.

2. Nevýhody:

- Příliš vysoká míra abstrakce kódu.

2 RASPBERRY PI

Raspberry Pi (dále RPi) je malý, cenově dostupný a energeticky úsporný jednodeskový počítač, který byl původně navržen pro podporu výzkumu a výuky informatiky ve školách. Obecně je to řada populárních jednodeskových (SBC single board computer) počítačů, do které spadají ty, mající v názvu slovo Pi. Tyto počítače se začaly objevovat přibližně od března roku 2012, ale historie sahá až do roku 2006 na místo Cambridge univerzity a ihned po uvedení na trh byla zaznamenána obrovská poptávka.[22]

Tento nový typ SBC byl revoluční v jednoduchosti připojení dalších periférií přes rozhraní jako je USB, LAN, HDMI a také dostatečný výkon, modifikovatelnost a nízkou cenu. Tuto revoluci ve své podstatě zahájila firma Raspberry Pi Foundation [6] a od té doby již mnoho dalších počítačů velmi podobných RPi, takzvaných klonů.

Tyto klony se často zaměřují na řešení omezení a nedostatků původních modelů, které mohou být aktualizovány s poměrně dlouhými intervaly. Nicméně, tato snaha vylepšit původní modely může vést k výrobě produktů s nižší kvalitou, což se projevuje ve formě snížené spolehlivosti a komplikací s kompatibilitou v porovnání s originálními modely. Tento problém je často způsoben především použitím jiného typu ARM čipu, což může přinášet nesrovnalosti v chování a kompatibilitě s originálním hardwarem.

V tomto projektu se konkrétně pracuje s verzí *RPi 4 Model B 2GB RAM*.

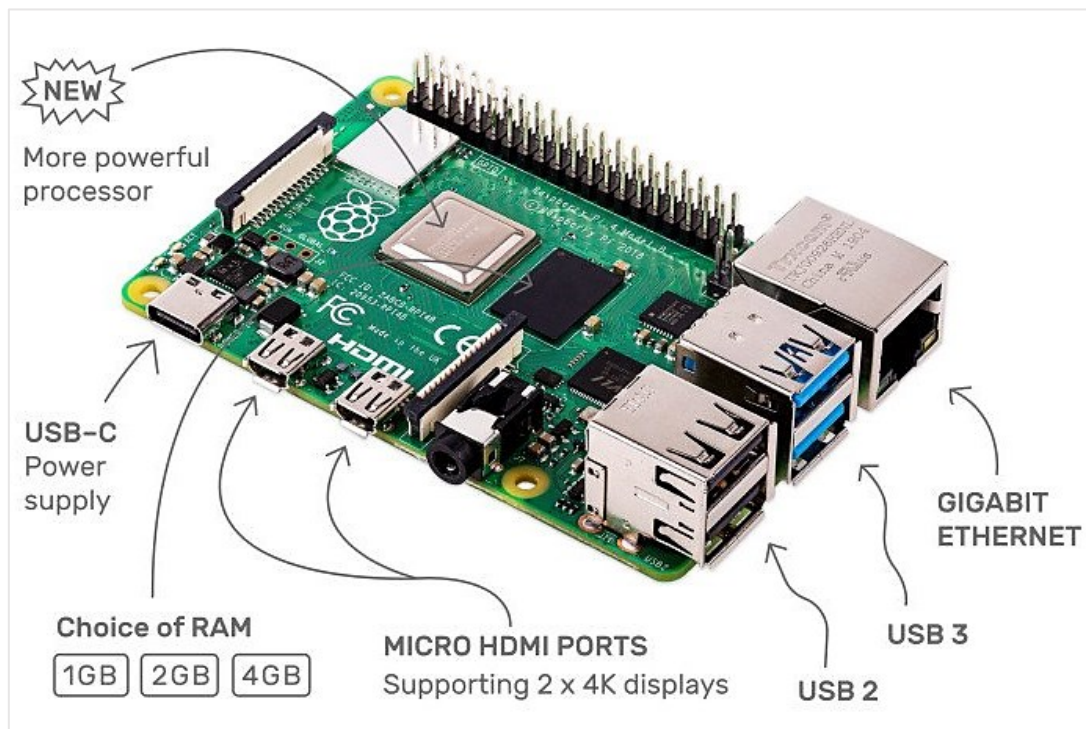
2.1 Architektura ARM

Architektura ARM, kterou vyvinula společnost ARM Holding (ARM Limited), je hojně využívána při výrobě procesorů pro běžné jednodeskové počítače. Jednou z charakteristických vlastností této architektury je používání redukované instrukční sady RISC, což umožňuje jednodušší konstrukci procesoru s menším počtem elektronických prvků na křemíku a především s nižší spotřebou elektrické energie. Tato nízká spotřeba umožňuje využití procesorů s architekturou ARM v mnoha oblastech spotřební a mobilní elektroniky, jako jsou mobilní telefony, tablety, MP3 přehrávače a mnoho dalšího.[25]

2.2 Raspberry Pi 4B

Raspberry Pi 4B (nebo také Raspberry Pi 4 Model B, obr. 3) je nejnovější verzí malého jednodeskového počítače, který se stal velmi populárním díky své nízké ceně, výkonnosti a jednoduchému použití. Raspberry Pi 4B je vybaven vylepšeným čtyřjádrovým procesorem

ARM Cortex-A72 s taktom 1,5 GHz a 2 nebo 4 GB operační paměti. Dále obsahuje dva mini HDMI porty, Gigabit Ethernet, Wi-Fi a Bluetooth, USB 3.0 a USB 2.0 porty a slot na microSD kartu pro uložení operačního systému a dat. RPi 4B je vhodný pro různé projekty a aplikace, jako jsou domácí multimediální centra, IoT projekty, výpočetní servery, robotiku, herní konzole. Lze na něm provozovat různé operační systémy, včetně Raspberry Pi OS, Ubuntu a mnoho dalšího.



Obrázek 4 Rpi4 B [7]

2.2.1 Základní parametry

Přehled všech základních parametrů jednodeskového počítače RPi [11]:

- **Procesor:** Broadcom BCM2711, čtyřjádrový ARM Cortex-A72 64bitový procesor s taktom 1,5 GHz
- **Operační paměť (RAM):** 2 GB až 8 GB LPDDR4-3200 SDRAM (v závislosti na verzi)
- **Síťové rozhraní:** Gigabit Ethernet, 2,4 GHz a 5 GHz IEEE 802.11ac bezdrátové rozhraní, Bluetooth 5.0, BLE
- **Porty USB:** 2x USB 3.0 a 2x USB 2.0
- **Porty video výstupu:** 2x micro HDMI (až do 4Kp60 podpory)
- **Audio:** 2-pólový kombinovaný konektor pro stereo výstup a mikrofonní vstup, analogový audio výstup na 3,5 mm jack
- **Uložiště:** MicroSD slot pro načtení operačního systému a dat
- **Napájení:** USB-C konektor 5V/3A (minimálně pro bezproblémový provoz)
- **GPIO:** 40pinový GPIO header kompatibilní s předchozími verzemi Raspberry Pi
- **PoE-capable:** Vyžaduje připojení PoE HAT

2.2.2 Konektivita čipu BCM2711

Jelikož možnosti samotného jednodeskového počítače RPi vychází z možností [8] samotného čipu BCM2711 jsou zde zmíněny i periferie tohoto čipu, ke kterým může ARM bezpečně přistupovat:

- Časovače
- řadič přerušení
- GPIO
- USB
- PCM / I2S
- Řadič DMA
- I2C mastery
- SPI mastery
- PWM
- UART

Jedním z důležitých faktorů pro jednodeskové počítače, jako je RPi, je konektivita. RPi nabízí mnoho možnosti konektivity, aby bylo možné snadno komunikovat s externími zařízeními a sítěmi. Některé z hlavních konektivit Raspberry Pi jsou:

- **Ethernet:** 1x Gigabit Ethernet port (RJ45) pro připojení kabelové sítě pro přístup k internetu nebo k jiným síťovým zařízením.
- **Wi-Fi:** 2,4 GHz a 5 GHz IEEE 802.11 ac bezdrátové rozhraní pro připojení k bezdrátovým sítím. Podporuje standardy 802.11 b/g/n/ac.
- **Bluetooth:** Bluetooth 5.0 rozhraní pro bezdrátovou komunikaci s jinými zařízeními, jako jsou klávesnice, myši, sluchátka a další. Podporuje profil A2DP a AVRCP pro přehrávání audia.
- **USB:** 2x USB 3.0 porty a 2x USB 2.0 porty pro připojení externích zařízení, jako jsou klávesnice, myši, tiskárny, USB disky a další. USB 3.0 porty umožňují rychlost přenosu dat až 5 Gbit/s.
- **GPIO:** 40pinový GPIO header pro propojení s externími zařízeními a senzory. Podporuje rozhraní I2C, SPI a UART.
- **Kamery a displeje:** 2x micro HDMI porty pro připojení dvou displejů s rozlišením až do 4Kp60. Podporuje HDMI-CEC a HDCP 2.2. 2-pólový kombinovaný konektor pro připojení kamerového modulu.
- **Audio:** 2-pólový kombinovaný konektor pro stereo výstup a mikrofonní vstup, analogový audio výstup na 3,5 mm jack konektor.
- **Slot na paměťovou kartu:** MicroSD slot pro načtení operačního systému a dat.

Tento detailní výpis konektivity Raspberry Pi 4B ukazuje, jak může být tento malý jednodeskový počítač využit v mnoha různých aplikacích. Díky své široké konektivě je Raspberry Pi 4B nepochybně vhodný pro vývoj IoT (internet věcí) projektů, domácí automatizace, výpočetní servery, multimediální centra a další.

2.3 Propojení RPi a 4WD HAT

Tato kapitola se zaměřuje na popis propojení mezi jednodeskovým počítačem RPi a rozšiřující deskou 4WD-HAT, což bylo již dříve zmíněno. Diskutovány jsou zde jak hardwarové, tak i softwarové aspekty tohoto propojení. Zvláštní pozornost je věnována komunikaci I²C mezi těmito dvěma komponenty, stejně jako knihovně smbus, která poskytuje nástroje pro řízení tohoto komunikačního rozhraní. Cílem této kapitoly je poskytnout čtenáři komplexní pohled na fungování celého systému.

2.3.1 HAT obecně

HAT (*Hardware Attached on Top*) je označení pro rozšiřující desky navržené pro RPi. Jsou navrženy tak, aby byly kompatibilní s rozložením vstupně-výstupních pinů, jinými slovy GPIO (*General Purpose Input/Output*, popis viz kapitola 2.3.4.1), na RPi a připojují se přímo na vrch jednodeskového počítače. HATy usnadňují přidávání funkcí obecně a rozšiřují možnosti RPi pro různé aplikace a projekty. [26]

HATy musí splňovat určitá pravidla a specifikace stanovené Raspberry Pi Foundation, což zahrnuje rozměry, kompatibilitu se základní deskou Raspberry Pi a identifikaci EEPROM. EEPROM na HATu uchovává informace o desce a konfiguraci, což umožňuje Raspberry Pi automaticky rozpoznat připojený HAT a nastavit potřebné ovladače a parametry.

Existuje mnoho různých HATů pro různé účely, jako jsou například pro ovládání motorů, připojení displejů, přehrávání zvuku, komunikace přes GSM, GPS, PoE, TV nebo LoRa-WAN (*Long Range Wide Area Network*), řízení servomotorů a mnoho dalších. HATy mohou být využity v širokém spektru projektů a aplikací, od robotiky přes domácí automatizaci až po IoT (*Internet of Things*) a výukové nástroje.

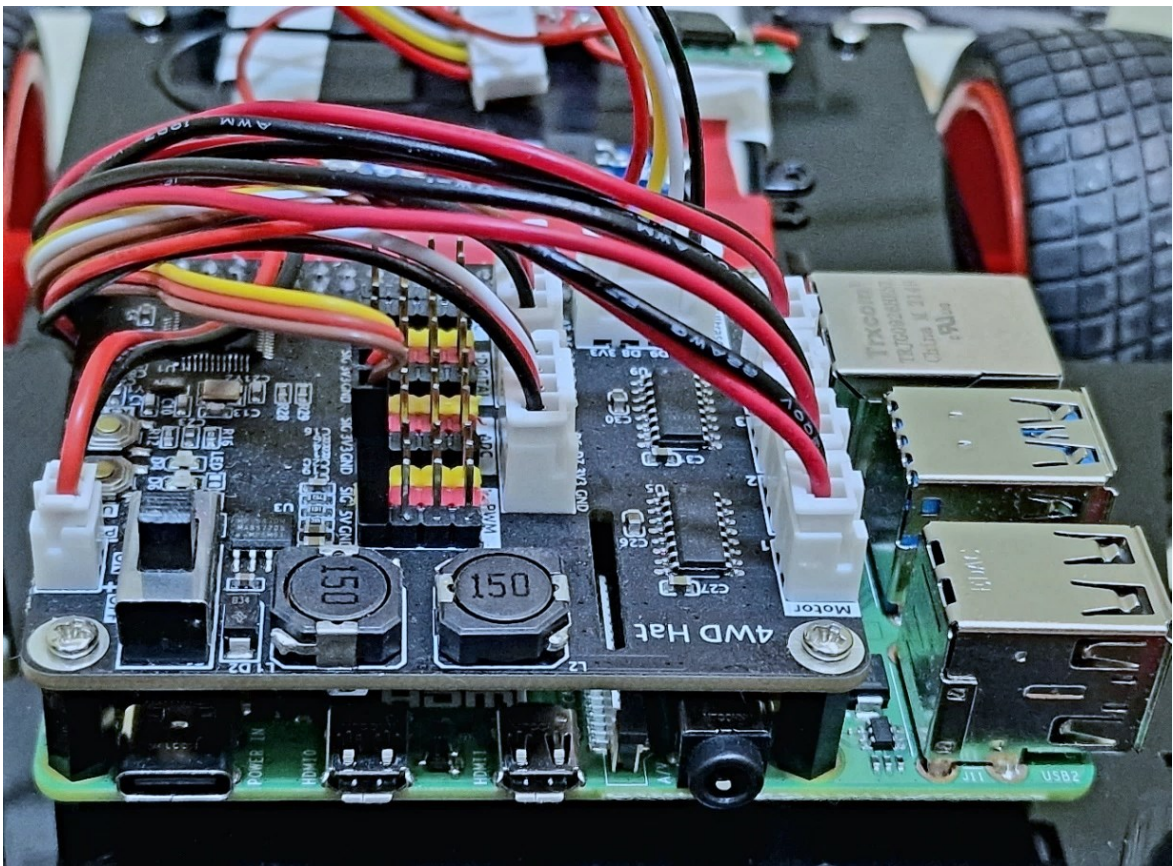
2.3.2 4WD HAT

4WD HAT je rozšiřující modul pro RPi od Sunfounderu, který slouží k řízení a komunikaci s jednotlivými komponenty stavebnice PiCar-4WD. HAT zajišťuje řízení pohonu kol, servomotorů, senzorů a dalších periferií modelu auta.

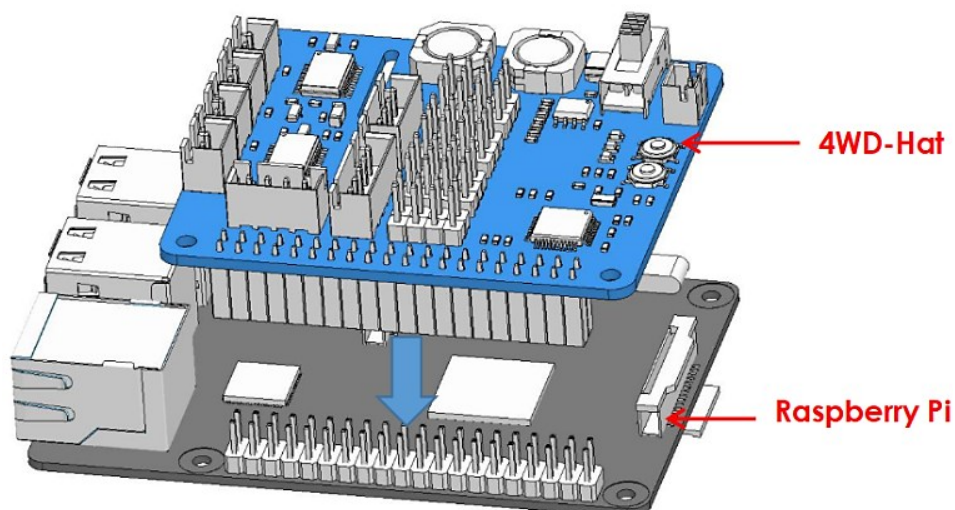
Raspberry Pi se připojuje k 4WD HAT pomocí GPIO pinů a komunikuje s HATem prostřednictvím sériového portu, v tomto případě I²C. Tímto způsobem lze ovládat jednotlivé funkce a prvky modelu auta, jako je například pohon, řízení, odhad vzdálenosti pomocí ultrazvukových senzorů nebo LED osvětlení apod. 4WD HAT tedy umožňuje snadnou integraci hardwarových komponent PiCar-4WD s RPi a zjednodušuje tak vývoj software pro řízení a ovládání modelu auta. [1]

2.3.3 Mechanické propojení

Deska je 4WD HAT je speciálně navržena pro počítače RPi a proto je možné i přímo napojit na 40 pinový konektor RPi. Rozměrově jej částečně kopíruje – vlastnost HATů, aby ve spojeném stavu vznikl kompaktní celek (obrázek 5 a 6).



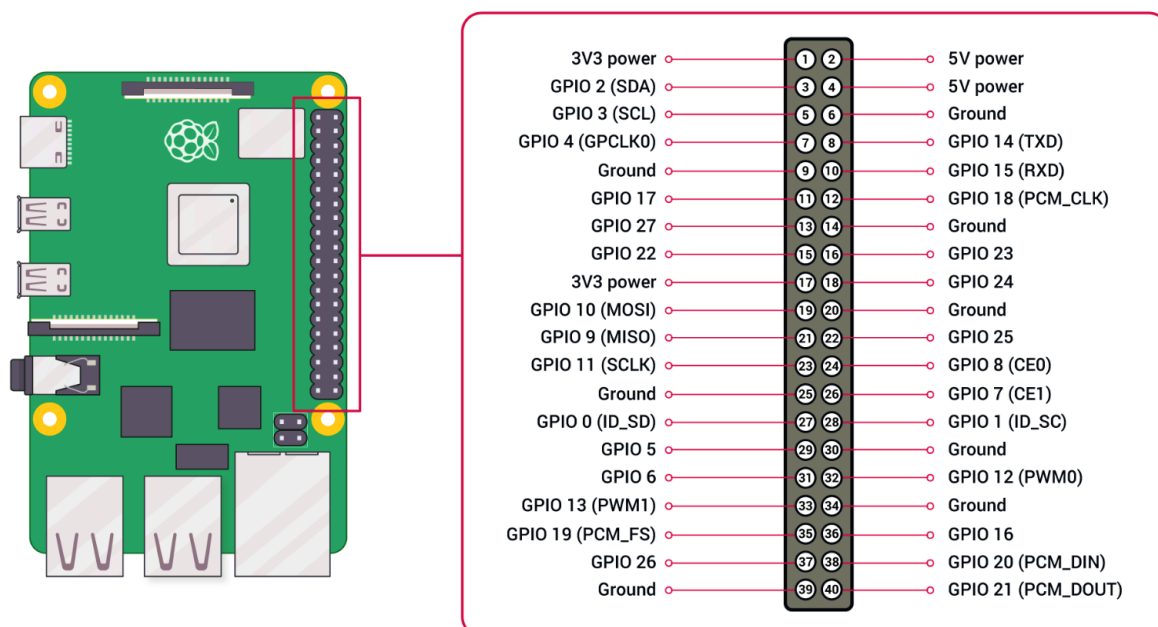
Obrázek 5 RPi a 4WD HAT složený stav



Obrázek 6 Vizualizace spojení Rpi a 4WD HAT[1]

2.3.4 Elektrické propojení a komunikace

Robotické auto může být propojeno s Raspberry Pi prostřednictvím několika rozhraní. Nejčastěji se používají GPIO piny (obrázek 7, popis viz následující podkapitola), které umožňují komunikaci mezi Raspberry Pi a různými součástkami auta, jako jsou motory, senzory nebo kamery. GPIO piny mohou být konfigurovány jako vstupní nebo výstupní a jsou schopné poskytovat nebo přijímat digitální signály.



Obrázek 7 Přehled 40-pinového headeru [9]

Jak již bylo uvedeno dříve, jednodeskový počítač je osazen rozšiřující deskou 4WD HAT, která pomocí sériové sběrnice I²C (popis viz kapitola 2.3.4.2), která je součástí GPIO,

komunikuje s počítačem RPi. Pro práci s touto sběrnici je využita knihovna *smbus* (popis viz kapitola 2.3.4.3).

2.3.4.1 *GPIO*

GPIO (General Purpose Input/Output) představuje všeobecný vstupní/výstupní systém, jehož součástí jsou rozhraní, která jsou využívána v rámci mnoha mikrokontrolerů a mikropočítačů, jako je Raspberry Pi. Toto rozhraní zprostředkovává komunikaci s různými externími zařízeními a moduly.

Rozhraní GPIO je na RPi umístěno na 40pinovém konektoru. Piny (kontakty) tohoto rozhraní lze konfigurovat buď jako vstupy nebo výstupy. V režimu vstupu mohou piny číst data z různých senzorů, tlačítek a dalších zařízení. V režimu výstupu mohou piny ovládat zařízení jako LED diody, motory a podobně. Některé z pinů také podporují speciální funkční režimy, jako jsou I2C, SPI, UART a další, což umožňuje komunikaci s širokou škálou zařízení. [9]

2.3.4.2 *Sériová sběrnice I²C (I2C)*

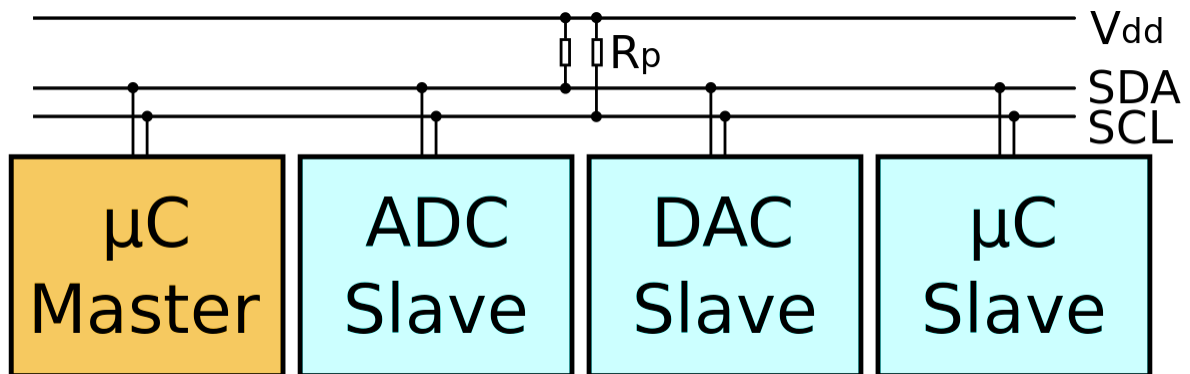
I²C (Inter-Integrated Circuit), původně IIC, je sériový komunikační protokol vyvinutý společností Philips. V českém prostředí se často označuje jako "í-dva-cé". Někteří výrobci, například Atmel, označují tuto sběrnici jako TWI (Two Wire Interface) a to kvůli ochranné známce Philips.[10]

I²C se liší od SPI komunikace tím, že používá pouze dva vodiče:

- datový signál (SDA)
- hodinový signál (SCL)

Má jeden nebo více master obvodů a připojené periferní obvody. Každý periferní obvod má svou sedmibitovou adresu (0 až 127). Vývody SDA a SCL jsou navrženy s otevřeným kolektorem, což znamená, že potřebují pull-up rezistor pro klidový stav logické 1.

Master zahajuje komunikaci pomocí startovního signálu a následně vysílá hodiny a nastavuje data na SDA. Přenos dat začíná sedmibitovou adresou a bitem R/W, který určuje, zda následuje čtení nebo zápis do zařízení. Zařízení s rozpoznanou adresou potvrdí připravenost komunikovat pomocí pulsu ACK (potvrzení). Poté následuje vlastní přenos dat, který řídí master.[10]

Obrázek 8 Blokové schéma sběrnice I²C [10]

I²C umožňuje některé pokročilé funkce, jako desetibitové adresy nebo „natažení času“, pokud je periferní zařízení pomalé, uzemní hodinový signál a master chvíli počká. Základní rychlost přenosu je 100 kHz, ale některá zařízení podporují rychlosti až 400 kHz nebo 3,4 MHz. Ultra-fast zařízení komunikují jednosměrně a nepoužívají pull-up rezistory.[10]

Tabulka 1 Přehled a porovnání SPI a I²C [10]

Vlastnosti	SPI	I2C
Rychlost	Teoreticky neomezená, prakticky omezená fyzikálními limity	100 kbit/s 400 kbit/s 3,4 Mbit/s
Počet vodičů	1x hodinový signál 2x datový signál 1x řídicí vodič pro každé zařízení 1x Společná zem	1x hodinový signál 1x datový signál 1x Společná zem
Počet připojených zařízení	Teoreticky neomezený, prakticky omezený potřebou mít pro každé zařízení jeden řídicí signál vodič CS (Chip Select)	Teoreticky je počet zařízení připojitelných k I2C sběrnici neomezen. Avšak v praxi je omezen sedmibitovým adresováním, což umožňuje připojit maximálně 128 zařízení. Při použití více obvodů stejného typu je nutné zajistit, aby každý z nich měl svou jedinečnou adresu. Pokud konkrétní obvod nepodporuje nastavení různých adres, nelze takové obvody připojit více najednou.

2.3.4.3 Knihovna *smbus*

SMBus (*System Management Bus*) je jednoduchý dvoukabelový sběrný systém, který je obecně používán pro komunikaci mezi různými perifériemi. Je založen na sběrnici I²C, pro správnou funkci je nutné tuto sběrnici na RPi povolit následovně:

```
sudo raspi-config
```

Otevře se okno, pokračuje se výběrem interfacing options a I²C.

Pro programování v Pythonu na RPi je k dispozici knihovna *smbus*, která umožňuje komunikaci se zařízeními připojenými k I²C rozhraní.[27]

Pro instalaci knihovny *smbus*, můžeme použít následující příkaz v terminálu, obdobný postup pro instalaci lze použít i jiné knihovny:

```
sudo apt-get install python3-smbus
```

nebo

```
sudo apt-get install python-smbus
```

Po instalaci je možno tuto knihovnu importovat do jakéhokoliv python scriptu pomocí:

```
import smbus
```

Knihovna *smbus* poskytuje řadu funkcí, které můžeme použít k odesílání a přijímání dat přes I²C rozhraní. Například pro vytvoření instance SMBus pro komunikaci s I²C zařízeními na sběrnici 1, lze použít:

```
bus = smbus.SMBus(1)
```

Poté můžeme číst a zapisovat data na konkrétní adrese pomocí funkcí *read_byte_data*, *write_byte_data* a dalších.

Knihovna defaultně na RPi využívá tyto digitální piny:

- GPIO 2 (BCM 2) pro SDA (Serial Data)
- GPIO 3 (BCM 3) pro SCL (Serial Clock)

Pro více podrobností o použití knihovny *smbus* v Pythonu doporučuji nahlédnout do dokumentace nebo do některých návodů dostupných online. Tato knihovna je jednoduchým wrapperem Linuxových funkcí I²C/SMBus ioctl (input/output control) volání a nemá proto žádnou oficiální dokumentaci. Nicméně, lze nalézt řadu komunitních průvodců, tutoriálů a příkladů kódu, které vysvětlují způsob použití knihovny *smbus* v Pythonu.

3 NÁSTROJE PRO TVORBU MOBILNÍCH APLIKACÍ

Aplikace je vytvořena pomocí kombinace HTML a JavaScriptu, které jsou běžné technologie používané pro vývoj webových a mobilních aplikací.

HTML (*Hyper Text Markup Language*) je základní stavební kámen webových stránek a je používán pro strukturování a prezentaci obsahu na webu. Kód ukazuje strukturu HTML stránky, která obsahuje různé prvky, jako jsou tlačítka a divy, pro interakci s uživatelem a zobrazení informací.

JavaScript je programovací jazyk, který se používá pro vytvoření interaktivních prvků na webových stránkách. V tomto kódu je JavaScript použit pro definování chování robotického auta, jako je pohyb vpřed, vzad, otočení doleva a doprava a zastavení.

Kód také zahrnuje využití knihovny Blockly od společnosti Google. Tato knihovna umožňuje vizuální programování pomocí bloků a je ideální pro výuku základů programování a robotiky.

Dále je použit webserver pro komunikaci mezi mobilní aplikací a robotickým autem. Webserver přijímá pokyny od mobilní aplikace a předává je robotickému autu. Tento způsob komunikace umožňuje vzdálené ovládání robota a jeho interakci s uživatelem. Pro více informací je zde² dostupná dokumentace.

Výběr HTML, JavaScriptu a webového serveru pro tento projekt je zdůvodněn jejich širokou dostupností, flexibilitou a schopností pracovat na různých platformách. Tyto technologie umožňují vývoj robustní a univerzální mobilní aplikace pro řízení robotického auta, která může být použita na různých zařízeních, jak již bylo zmíněno v kapitole možnosti řízení, bez potřeby instalace specifického softwaru.

3.1 Stěžejní knihovny

Knihovna v programování je soubor kódu nabízející specifické funkce, umožňující znovu použití ve více programech. Mohou zahrnovat různé funkce, např. vykreslování grafiky či manipulaci s daty. Zásadní vlastnosti knihovny jsou znovupoužitelnost a abstrakce, což ulehčuje její použití bez detailního pochopení jejího fungování. Knihovny se dělí na statické a

² <https://docs.python.org/3/library/http.server.html>

dynamické, přičemž každý typ má své výhody a nevýhody. Použití knihovny šetří čas a úsilí, protože nabízí předpřipravená řešení. Distribuce knihovny zahrnuje hlavičkové a objektové soubory. [17]

3.1.1 Blockly

Blockly[12] je open-source knihovna pro vývoj nejen webových aplikací určených pro blokové programování. Byla vyvinuta společností Google a je určena k vytváření vlastních vizuálních programovacích jazyků (VPLs). Umí exportovat bloky do mnoha programovacích jazyků, včetně těchto populárních možností:

- JavaScript
- Python
- PHP
- Lua
- Dart

Principem blokového programování je, že uživatelé mohou vytvářet programy tím, že "poskládají" bloky kódu, místo toho, aby ručně psali syntakticky přesné textové kódy. Každý blok představuje určitou funkci nebo příkaz, který lze použít v programu. Bloky mohou být například cykly, podmínky, matematické operace, a tak dále.

Blockly se obecně používá k výuce programování a počítačového myšlení, ale může být také použit v jakémkoli projektu, kde je třeba umožnit uživatelům tvořit vlastní skripty nebo algoritmy bez nutnosti znát detaily programovacího jazyka.

Generování kódu v knihovně Blockly převádí vizuální bloky na spustitelný kód, jako je JavaScript. Tento kód je poté možné spustit v prostředí podporujícím JavaScript, jako je webový prohlížeč nebo serverové prostředí Node.js. Interpretace JS kódu vyžaduje externí nástroj, například Node.js nebo webový prohlížeč. Pro výukové účely a krokování kódu může být použita externí knihovna, jako je js-interpreter, která umožňuje bezpečné spuštění JS kódu, krokování a definování vlastních funkcí. Blockly a js-interpreter jsou tedy dvě samostatné knihovny použité společně v tomto projektu - Blockly pro generování kódu a js-interpreter pro jeho bezpečné spuštění a krokování. [15]

3.1.2 JS-interpreter

Knihovna js-interpreter[21] je open-source projekt vytvořený Neilem Fraserem, který je rovněž jedním z hlavních přispěvatelů do knihovny Blockly. JS Interpreter je navržen tak, aby

poskytoval bezpečné a omezené prostředí pro spouštění JS kódu. Jeho použití je také zmíněno v oficiální dokumentaci Blockly.

Knihovna js-interpreter implementuje vlastní JavaScriptový engine, který je schopen interpretovat JS kód krok za krokem. Toto je výhodné pro výukové a testovací účely, kde může být užitečné sledovat, jak se kód vykonává v reálném čase. Js-nterpreter také poskytuje schopnost definovat vlastní funkce a proměnné, které mohou být poté použity v interpretovaném kódu. To umožňuje vytvářet personalizované a bezpečné prostředí pro spouštění kódu, což je zvláště důležité v případech, kdy je kód generován nebo poskytován uživateli, jako je tomu v případě aplikace, která využívá knihovnu Blockly. Použití knihovny js-interpreter v kontextu tohoto projektu umožňuje bezpečné a ovladatelné spouštění JavaScriptového kódu generovaného z Blockly bloků. Tímto způsobem je možné nejen vykonávat instrukce pro řízení robotického auta, ale také poskytnout uživatelům možnost sledovat, jak se jejich program vykonává krok za krokem.[15] Pro více informací o js-interpretru je zde³ dostupná dokumentace.

3.2 Android Studio

Je od Googlu navržené a udržované integrované vývojové prostředí pro Android aplikace[13]. Poskytuje nástroje jako editor kódu, debugger a emulátor, a podporuje jazyky jako Java, Kotlin a C++[14]. Robustně integruje Gradle pro automatizaci vývoje a uživatelské rozhraní umožňuje snadnou úpravu layoutů. Android Studio také poskytuje nástroje pro analýzu výkonu, testování a ladění aplikací[13]. Kromě toho existují i jiné frameworky pro vývoj mobilních aplikací jako React Native, Flutter, Xamarin, NativeScript a Ionic, s různými výhodami a nevýhodami pro různé projekty a týmy[14].

³ <https://neil.fraser.name/software/JS-Interpreter/docs.html>

II. PRAKTICKÁ ČÁST

4 SESTAVENÍ ROBOTA

Robotické auto RpiCar-4WD (obrázek. 9) a jednodeskový počítač RPi zakoupila a poskytla firma ZF Engineering Plzeň.



Obrázek 9 Krabice stavebnice PiCar-4WD

Sestavení robota bylo realizováno dle pokynů v oficiální příručce [20]. Přestože některé součásti mohly během procesu sestavování působit pružně a křehce, po úplném sestavení se robot jeví jako robustní celek. Je důležité podotknout, že přestože byla procedura sestavování bez větších komplikací, v rámci této práce se nebudeme věnovat detailnímu popisu tohoto procesu.

V průběhu sestavování bylo identifikováno několik klíčových postřehů. Jedním z nich je použití malých kleští při manipulaci s miniaturními maticemi. Použití tohoto nástroje se ukázalo být neocenitelné pro důkladné dotažení matic, což je nezbytné pro udržení jejich pevnosti. Bez pevného utažení mohou tyto malé matice po krátkém čase povolít, což by mohlo mít za následek nestabilitu celého systému. Především uchycení motorů je nejvíce náchylné. Tento postřeh nám přinesl důležité pochopení pro zajištění dlouhodobé stability a funkčnosti robota.

Celkově lze říci, že proces sestavení robota představuje důležitou fázi vývoje, která ilustruje, jak jednotlivé, možná na první pohled křehké komponenty, mohou ve společném fungování vytvářet odolný a robustní systém.

5 ÚPRAVY HARDWARU

Jelikož se jedná o stavebnici, často se najdou oblasti, které buďto částečně nebo zcela nevyhovují požadavkům. Tato kapitola je tedy věnována opravě hardwarových nedostatků spojených s dosavadním řešením. Tyto nedostatky vyvstali během samotného vývoje softwaru.

Upozorňuji, že manipulace s bateriovými články li-ion/li-pol, ke které bude docházet v následující kapitole může být nebezpečná.

5.1 Napájecí soustava

Standartní součástí balení robotického auta je li-ion baterie 18650. Ta má ovšem své nevýhody, a to především jejich nízkou proudovou odolnost a vysoké nároky stav článků. Jejich vysoký vnitřní odpor způsobuje významný pokles napětí baterie, zejména při rozjezdu na vyšší rychlost, od 50% a více. Tento pokles napětí může vyvolat resetování počítače Rpi, což vyžaduje čekání na celý bootovací proces, který trvá téměř minutu to je pro jakékoliv použití nepřijatelné. Tento problém je podrobněji popsán a řešen v následující podkapitole.

Existuje několik možných řešení tohoto problému:

1. Přidání více článků 18650 do jednotlivých sérií baterie.
2. Použití zcela jiných článků, jako jsou li-pol/liPo (lithium-polymerové), a vytvoření zcela nové baterie.

Byla vybrána druhá možnost, tedy použití li-pol článků. Tato volba zahrnuje implementaci vlastního BMS (*Battery Management System*) a nabíjecí kaskády s využitím portu USB-C pro nabíjení a technologie PD (*Power Delivery*) s napětím 9V.

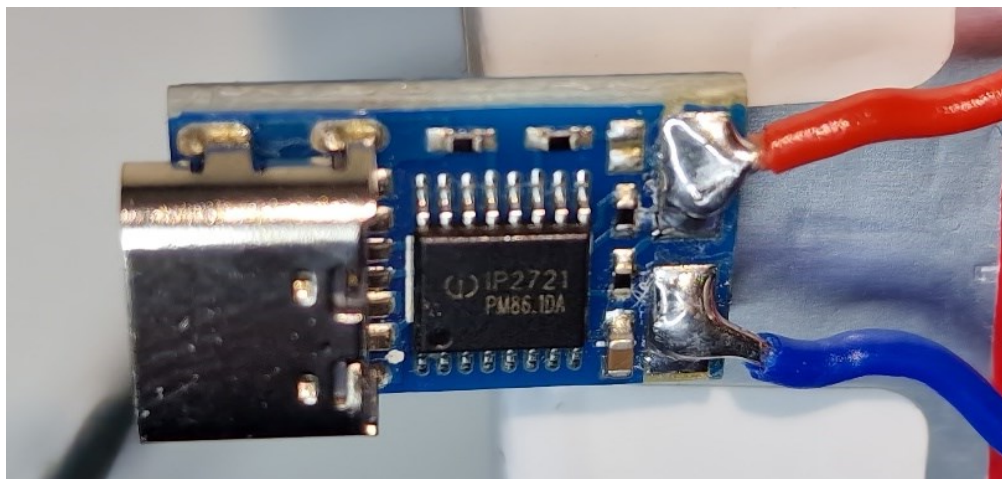
Dalším problémem je, že nabíjení těchto článků není uživatelsky pohodlné a je nepraktické. Každé nabíjení vyžaduje vyjmutí článků z držáku, což vylučuje možnost nabíjení přímo na modelu, a to každý separátně, protože v modelu auta jsou zapojeny do série. Články musí být nabíjeny pomocí externí modelářské nabíječky nebo zdroje s režimem konstantní proud-konstantní napětí (CC-CV).

Tento problém může být vyřešen použitím zcela odlišného typu článků, konkrétně lithium-polymerových (Li-Po) článků. Ty se liší svým tvarem a obecně mají o něco vyšší kapacitu než články 18650.

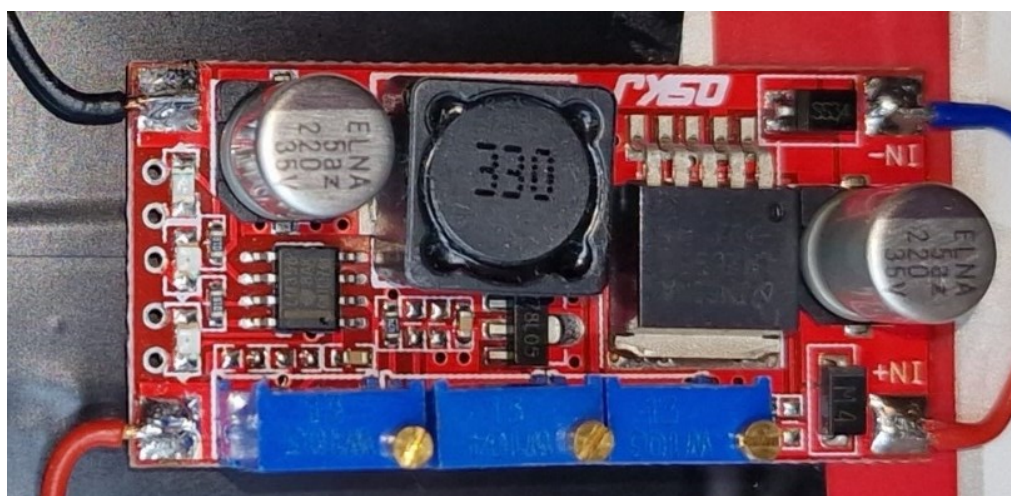
5.1.1 Řešení nabíjení

Původní napájecí systém modelu RPi-Car byl založen na dvou li-ion článcích 18650 zařazených v sérii, což poskytovalo maximální napětí 8,4V v nabitém stavu a napětí okolo 6V při vybití. Problémy s tímto spojené byly uvedeny v předešlé kapitole.

Pro řešení těchto problémů bylo napájení upraveno tak, aby umožnilo nabíjení přes konektor USB-C. Nabíječky a powerbanky podporující protokol PD (Power Delivery) jsou schopny dodávat i napětí 9V, což je dostatečné pro potřeby RPi-Car. Za konektorem USB-C je umístěn PD chip, který zajistí vyžádání napětí 9V. Bez tohoto chipu by na výstupu nabíječek bylo buďto 0V u novějších modelů anebo 5V u starších typů, což by pro potřeby RPiCar-4WD bylo zcela nedostatečné. Na obrázku 10 je zleva: USB-C konektor, následuje integrovaný obvod (IC) IP2721, vývody kladného a záporného pólu napájení.



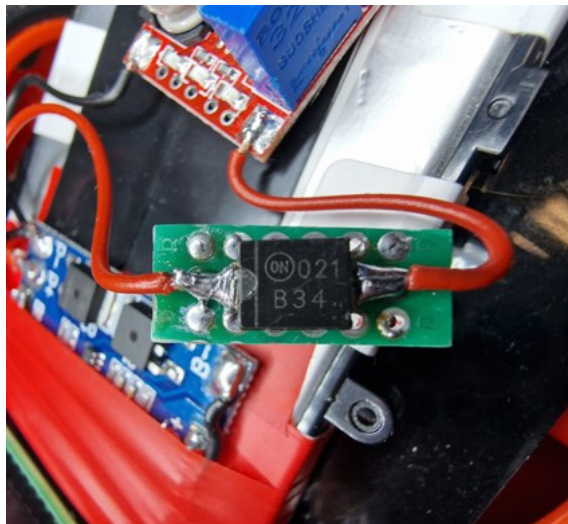
Obrázek 10 USB-C, IC IP2721 pro fyzickou vrstvu USB



Obrázek 11 DC-DC step down zdroj s CC-CV

Napětí 9 V z USB-C je pak pomocí měniče stejnosměrného napětí/proudu (DC-DC) sníženo na maximální nabíjecí hodnotu 8,7 V (Obrázek 11). Tato hodnota získána součtem napětí $4,2+4,2+0,29$ jde o dvakrát maximální napětí článku zvýšený (přibližně) o úbytek na diodě, vysvětleno dále. Tento měnič má možnost nastavení maximálního nabíjecího proudu (CC) na 1 A.

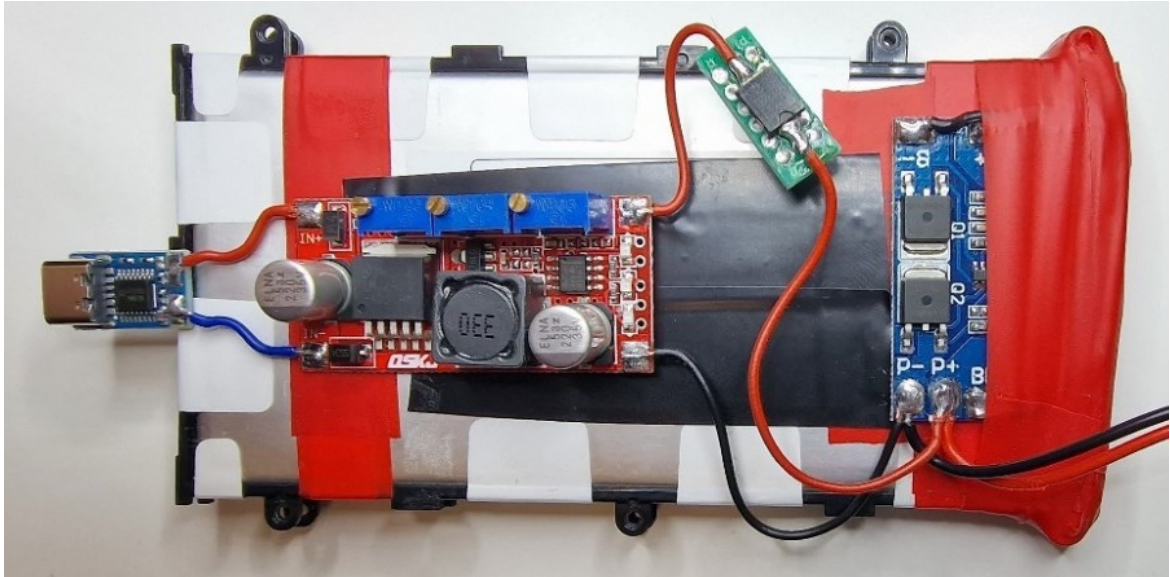
Za výstupem DC-DC měniče je umístěna Schottkyho dioda B34 (Obrázek 12), která má méně než poloviční úbytek napětí (změřeno 0,29 V) v porovnání s běžnými Si nebo Ge diodami. Tato dioda zajišťuje to, aby proud tekla pouze ze zdroje do baterie a ne naopak. Bez ní by stavové diody na DC-DC měničích stále svítily, dokud se baterie zcela nevybila.



Obrázek 12 Schottkyho dioda B34 – 3A max

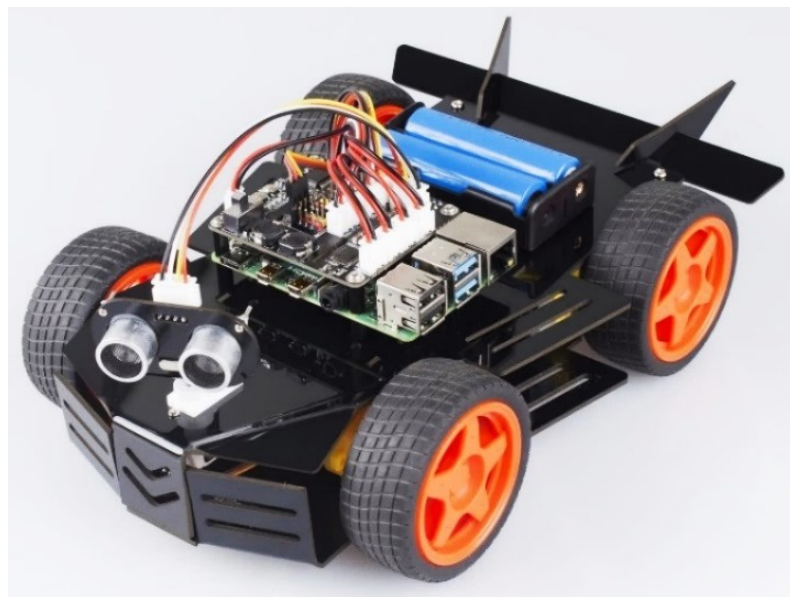
Výstup z DC-DC měniče je napojen na vstup/výstup 2S BMS (2 in series battery management system), který slouží k vyrovnání (tzv. balancování) napětí v každém článku v sérii. BMS také sleduje minimální a maximální napětí v každém článku, aby se zabránilo podbití nebo přebití baterií. To je zásadní pro prodloužení životnosti baterií, protože opakované podbití nebo přebití může vést k jejich poškození nebo dokonce zničení. BMS také monitoruje proud procházející baterií. Pokud je proud příliš vysoký, může dojít k přehřátí baterií, což by mohlo způsobit poškození nebo dokonce výbuch. Proto BMS často obsahuje ochranu proti přetížení, která omezí proud, když dosáhne určitého limitu. Další výhodou BMS je monitorování teploty článků baterie. Li-Pol baterie mohou být citlivé na extrémní teploty, takže jejich monitorování a regulace může předcházet problémům. Nakonec, BMS také často obsahuje funkci pro detekci zkratu. Pokud dojde ke zkratu, BMS okamžitě odpojí baterii, aby zabránilo dalšímu poškození.

BMS (obrázek 13) napravo částečně pod izolační páskou je důležitým prvkem jakéhokoli systému s Li-Pol bateriemi, protože poskytuje řadu funkcí pro ochranu a správu baterií, které zvyšují jejich životnost a především bezpečnost.



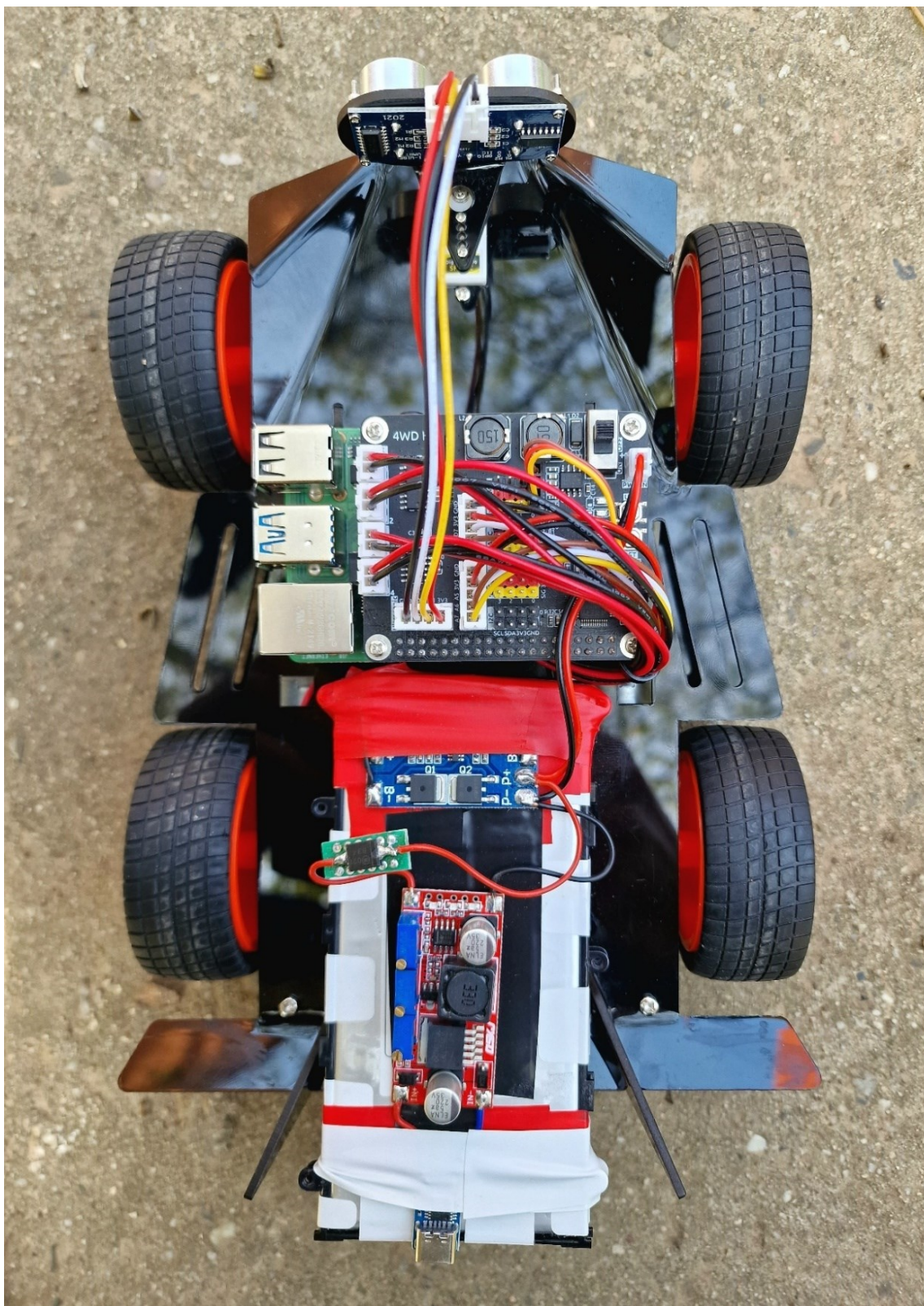
Obrázek 13 Kompletní napájecí soustava

BMS je připojeno k 2S baterii na anodě i katodě, a to i v bodě, kde je série vytvořena. Toto zapojení umožňuje měření napětí každého jednotlivého článku v sérii. Vstupní/výstupní porty BMS jsou pak napojeny na RPi HAT 4WD. Zde (obrázek 13) je napájení rozděleno a rozváděno do H-můstků, senzorů a samotného RPi. Toto řešení zajišťuje efektivní a bezpečné napájení celého systému.



Obrázek 14 RpiCar-4WD originální stav bez úprav [19]

Porovnání vzhledu Rpi-Car 4WD před a po úpravě napájení je na obrázcích 14 a 15. Zvětšený obrázek 14 je příloze P5 [19].



Obrázek 15 PiCar-4WD s novou baterií a nabíjením

6 NÁVRH APLIKACE

Další fází této práce je návrh vlastní webové aplikace. V této kapitole se zaměříme na klíčové aspekty našeho přístupu k vývoji této nové webové aplikace a poté se budeme zabývat požadavky na projekt.

Při vývoji nové webové aplikace jsme se soustředili na několik klíčových vlastností, které jsme identifikovali jako nezbytné pro úspěch našeho projektu:

1. **Rozšiřitelnost:** Naším cílem bylo vytvořit aplikaci, která je schopna růst a rozvíjet se společně s potřebami uživatelů a změnami v technologickém prostředí. To zahrnuje využití modulárního designu a principů čistého kódu pro snadnou integraci nových funkcí a úprav stávajících komponent.
2. **Udržitelnost:** Důraz byl kladen na vytvoření kódu, který je snadno čitelný, dobře dokumentovaný a připravený pro budoucí údržbu. Tím jsme se snažili zajistit, že jakékoli budoucí změny budou moci být implementovány efektivně a bez zbytečných komplikací.
3. **Uživatelská přívětivost:** Konečným cílem našeho projektu je vytvoření produktu, který je intuitivní a snadno použitelný pro uživatele. To zahrnuje pečlivý návrh uživatelského rozhraní a zkušeností, s důrazem na jednoduchost a uživatelský komfort.

Tyto klíčové vlastnosti odrážejí naši filozofii vývoje a pomohly nám při tvorbě rozhodnutí v průběhu celého procesu.

6.1 Požadavky na projekt

Před vývojem této nové webové aplikace byly pečlivě analyzovány a definovány požadavky tohoto projektu, které byly v průběhu diskutovány s firmou ZF Engineering Plzeň s.r.o. Toto je klíčový krok, který napomohl jasně definovat očekávané výsledky a funkcionality, které tato aplikace musí poskytovat. Požadavky jsou rozděleny do dvou hlavních kategorií: funkční a nefunkční požadavky.

Funkční požadavky se týkají konkrétních funkcí a operací, které aplikace musí poskytnout. Zahrnují například:

1. Implementace grafického programování (VPL)
2. Integrace s existujícím softwarovým systémem.
3. Poskytnutí interaktivního uživatelského rozhraní pro pohodlné ovládání.

4. Webová aplikace

Nefunkční požadavky se týkají operativních aspektů systému, jako je výkon, bezpečnost, udržitelnost, škálovatelnost a uživatelské prostředí. Některé z našich klíčových nefunkčních požadavků zahrnují:

1. Aplikace musí být schopna rychle a efektivně zpracovat požadavky uživatelů.
2. Kód musí být čitelný a snadno udržitelný pro budoucí rozšíření a údržbu.
3. Uživatelské rozhraní musí být intuitivní a snadno použitelné.

Tyto požadavky jsou vodící linií během celého procesu vývoje a pomohly vytvořit aplikaci, která splňuje potřeby našich uživatelů a zároveň je technicky robustní a udržitelná.

7 REALIZACE APLIKACE

Softwarová architektura tohoto projektu je složena ze dvou hlavních součástí: webové aplikace a souboru Python skriptů provozovaných na Raspberry Pi. Tyto skripty, vycházející z původního softwaru [16], jsou rozděleny do několika komponent, každá se specializací na určitou oblast funkcionality systému. Tyto skripty zajišťují provoz webového rozhraní a řízení hardwaru připojeného k Raspberry Pi.

Nová webová aplikace je vytvořena pomocí standardních webových technologií HTML, JavaScript a CSS. Přestože existují moderní frameworky, tyto nebyly využity vzhledem k požadavkům zmíněným v předešlé kapitole.

7.1 Analýza Python skriptů

Klíčové Python skripty jsou podrobně prozkoumány. Analýza je nezbytná, protože oficiální dokumentace k těmto skriptům neexistuje a komentáře v kódu jsou většinou buď zcela chybějící nebo jsou napsané v čínštině.

Skripty lze často spustit i přímo, což je běžné při prvotní inicializaci nebo pro testovací účely. Některé skripty, jejichž funkcionality není pro naše účely podstatná, nebudou dále podrobněji rozebírány.

Přehled všech Python skriptů:

- **Instalace**
 - setup.py
- **Web-server**
 - start.py
 - web_server.py
 - remote_control.py
- **Součásti knihovny picar_4wd**
 - __init__.py
 - adc.py
 - filedb.py
 - i2c.py
 - motor.py
 - pin.py
 - pwm.py
 - servo.py
 - speed.py
 - ultrasonic.py
 - utils.py

- **Autonomní jízda**
 - follow.py
 - obstacle_avoidance.py
 - track_line.py
- **Ostatní**
 - keyboard_control.py
 - move_forward.py

7.1.1 Podrobnější rozbor

Rozbor skriptů, který poskytne dostatečné pochopení celkové funkčnosti, s důrazem na ty skripty, které jsou pro fungování aplikace klíčové.

7.1.1.1 setup.py

Tento skript provádí několik úkonů potřebných pro správnou instalaci a konfiguraci picar-4wd, jako jsou:

- Funkce *setup* konfiguruje metadata balíčku (*package*) Pythonu. Toto zahrnuje název, verzi, autora, licence, závislosti a další detaily. Tato funkce také definuje vstupní body, což jsou skripty, které budou nainstalovány do systémové cesty a budou moci být spuštěny přímo z příkazové řádky.
- Funkce instalace je spuštěna při instalaci package. Provede aktualizaci systémového package manažeru (*apt-get*), instalaci potřebných závislostí (jako je *pip*, *sysstat* a *i2c-tools*) a také nastavení systémových konfiguračních souborů pro správnou funkci *picar-4wd*.
- Třídy pro práci s konfiguračními soubory jako *Modules* a *Config* jsou určené k manipulaci se systémovými konfiguračními soubory na systému Linux.
- Funkce pro práci s terminálem tedy *readchar* a *readkey* umožňují čtení vstupu od uživatele přímo z terminálu.

Spouštění příkazů v systému jako Funkce *run_command* a do umožňují spouštět systémové příkazy přímo z Pythonu a zachytit jejich výstupy a návratové kódy.

7.1.1.2 start.py

Tento skript je základem celé funkcionality, ten vyváří dva servery a to HTTP (*Hypertext Transfer Protocol*) server a websocket server, které poskytují webové rozhraní pro ovládání

modelu. HTTP server je určený pro provoz webových aplikací, kdežto server websocket zajišťuje interakci mezi webovou aplikací a vozidlem, jeho činnosti:

- Hlavní část skriptu se stará o spuštění serverů a zajišťuje jejich bezpečné ukončení v případě, že je skript přerušen. Zde se také získává IP adresa počítače.
- Obsahuje několik funkcí, které ovládají chod obou serverů. Funkce *start_http_server* a *start_websocket* spouštějí servery, zatímco funkce *close_http_server* a *close_websocket* je ukončují.
- Třída *restartServer* která je odvozena od třídy *BaseHTTPRequestHandler*, obsahuje přetíženou metodu *do_GET* pro zpracování HTTP GET požadavků. Pokud je požadavek na URL *'restart'*, server je restartován. Jinak je vrácena chybová stránka.

7.1.1.3 *web_server.py*

Tento skript řídí PiCar-4WD pomocí dálkového ovládání skrze websocket servery, které umožňují komunikaci v reálném čase mezi ovládacím rozhraním a modelem auta.

Používá se několik důležitých importů:

- *picar_4wd* je knihovna k ovládání auta PiCar-4WD. Více viz kapitola 7.1.1.4.
- *asyncio* a *websockets* jsou knihovny pro vytvoření asynchronních websocket serverů.
- *json* a *time* jsou standardní Python knihovny pro manipulaci s JSON daty a časem.

Systém spustí vlákno pro sledování rychlosti a inicializuje seznam pro sledování hodnot ze senzoru šedé škály (grayscale sensor). V rámci skriptu jsou vytvořeny dvě globální proměnné:

- *REC_DICT* udržuje aktuální stav všech důležitých parametrů vozidla, jako je rychlost, směr, status senzorů a další.
- *SEND_DICT* je použit pro odesílání informací zpět na server.

Dále jsou definovány tři hlavní asynchronní funkce:

1. *recv_server_func(websocket)*: Tato funkce přijímá data ze vzdáleného ovládání prostřednictvím websocketu, aktualizuje *recv_dict* slovník a zprostředkovává příslušné akce pro ovládání vozidla.
2. *send_server_func(websocket)*: Tato funkce shromažďuje data ze senzorů vozítka, aktualizuje *send_dict* slovník a odesílá tato data zpět do ovládacího rozhraní prostřednictvím websocketu.

3. *main_func()*: Tato funkce implementuje hlavní logiku pro ovládání vozítka na základě příchozích dat v *recv_dict*.

Dále jsou definovány dvě pomocné asynchronní funkce:

1. *main_logic_1(websocket,path)*
2. *main_logic_2(websocket,path)*

Ty slouží jako obálky (*wrppery*) pro přijímání a odesílání funkcí serveru.

V hlavní části skriptu je zahájen proces získávání IP adresy. Jakmile je IP adresa získána, jsou zahájeny dva webové servery na této IP adrese a na portech 8765 a 8766, každý pro jednu z funkcí přijímání a odesílání.

Tento skript také obsahuje blok *try/finally*, který zajišťuje, že v případě jakýchkoli chyb nebo ukončení programu je vozidlo zastaveno pomocí metody *STOP* knihovny *picar-4wd*.

7.1.1.4 Součásti knihovny *picar_4wd*

Tyto skripty jsou součástí knihovny pro řízení robota PiCar-4WD. Každý skript má specifickou funkci, popis hlavních funkčností skriptů:

1. **__init__.py**: Tento skript může být spuštěn jako hlavní program, načítá konfiguraci, inicializuje motory a serva, čte hodnoty čidel.
2. **adc.py**: Tento skript je zaměřen na práci s Analogově Digitálním Převodníkem (ADC) přes I²C komunikaci na konkrétním kanálu, který je určen při vytváření instance třídy. ADC je používán pro konverzi analogových signálů, jako jsou například ty, které pochází z různých senzorů, na digitální hodnoty. Hlavní funkce *read* v třídě *ADC* je používána k získání aktuální hodnoty z ADC převodníku. Proces zahrnuje zaslání požadavku na čtení z I²C adresy ADC a následné přečtení dvou bajtů, které představují hodnotu přečtenou z ADC.
3. **filedb.py**: Tento skript implementuje třídu *FileDB*, která umožňuje jednoduché ukládání a načítání dat do a ze souboru na disku. Metoda *get* načítá hodnoty z databáze podle zadaného klíče. Pokud klíč neexistuje, vrátí se předem definovaná výchozí hodnota. Metoda *set* nastavuje hodnotu pro daný klíč, nebo pokud klíč neexistuje, vytvoří nový záznam.
4. **i2c.py**: Tento skript obsahuje kód pro řízení komunikace pomocí I²C protokolu. Je to sériový komunikační protokol, který se často používá pro komunikaci mezi

mikrokontrolerem a různými periferními zařízeními, jako jsou senzory nebo jiné moduly. Využívá knihovny *smbus*.

5. **motor.py:** Tento kód definuje třídu *Motor*, která ovládá DC motory pomocí PWM signálu a směrových pinů. Metoda *set_power* nastavuje výkon a směr otáčení motoru podle hodnoty vstupu. Kód také obsahuje zakomentované sekce, které naznačují možnost postupného zvyšování výkonu motoru.
6. **pin.py:** Tento Python skript definuje třídu *Pin*, která je určena pro manipulaci s GPIO piny na Raspberry Pi. Třída obsahuje metody pro nastavení módu pinu (vstupní/výstupní), čtení a nastavování hodnoty pinu, aktivaci a deaktivaci pinu, a také pro nastavení přerušování. Součástí třídy je také vnitřní třída *cpu*, která představuje specifické GPIO piny Raspberry Pi.
7. **pwm.py:** Tento Python skript definuje třídu *PWM*, která rozšiřuje třídu *I2C*. Třída PWM umožňuje ovládání PWM (*Pulse Width Modulation*) signálů na Raspberry Pi. Třída obsahuje metody pro nastavení frekvence, šířky pulsu, prescaleru a period. Metoda *pulse_width_percent* umožňuje nastavit šířku pulsu jako procento periody.
8. **servo.py:** Třída *Servo* v tomto kódu umožňuje ovládat servo motor připojený k Raspberry Pi. Metoda *__init__* inicializuje třídu s daným pinem pro servo a případným offsetem. Tato metoda také nastavuje periodu a *prescaler* pro PWM signál na pinu. Metoda *set_angle* umožňuje ovládat polohu serva nastavením úhlu mezi -90 a 90 stupňů. Tato metoda převede úhel na dobu vysoké úrovně signálu (*High_level_time*), která je poté převedena na hodnotu pro nastavení šířky pulsu PWM signálu.
9. **speed.py:** Tento skript obsahuje třídu *Speed*, která slouží k měření rychlosti kol pomocí GPIO pinů Raspberry Pi. Třída obsahuje metody pro inicializaci (s nastavením GPIO režimu a pinu jako vstupu), spuštění měření (spuštěním vlákna), zastavení měření. Třída *Speed* používá separátní vlákno pro měření rychlosti v pravidelných intervalech. Měření je provedeno počítáním stoupajících a klesajících hran ze signálu získaném na pinu. Počet těchto hran je poté převeden na otáčky za sekundu (*rps*) a poté na rychlost v mm/s.
10. **ultrasonic.py:** Tento kód definuje třídu *Ultrasonic*, která se používá pro ovládání ultrazvukového senzoru v robotických aplikacích. Třída obsahuje metodu *get_distance*, která měří vzdálenost mezi senzorem a objektem pomocí echolokace – vysílá ultrazvukový signál a měří dobu, kterou signál potřebuje k návratu zpět po odrazu od objektu. Třída také obsahuje inicializační metodu, která nastavuje

počáteční parametry senzoru, jako je úhel skenování a krokování. Přerušené části kódu naznačují další plánované funkce pro skenování v určitém úhlu natočení.

11. **utils.py:** Tento skript obsahuje různé pomocné funkce, které jsou používány v ostatních skriptech. jako jsou reset, čtení úrovně napětí baterie, testy a systémové informace.

7.2 Webová aplikace

Webová aplikace je částečně generována, což se týká jednotlivých stránek, kaskádových stylů (CSS) a některých JavaScriptových (JS) souborů, které se starají o animace, události a další funkce. Vzhledem k tomu, že tyto generované soubory nejsou snadno upravitelné z hlediska jejich umístění nebo přejmenování, projekt jako celek může působit neúplně. Manuálně vytvořené CSS a JS soubory umístěny samostatně ve složkách "css" a "js".

Důležité je zmínit, že všechny ikony použité v aplikaci pochází z webu Flaticon [23].

V této webové aplikaci je implementováno dynamické načítání, alespoň prozatím, některých JS knihoven. To znamená, že tyto knihovny nejsou součástí počátečního balíčku stahovaného ze serveru, ale jsou požadovány a načítány za běhu aplikace. Příkladem těchto dynamicky načítaných knihoven jsou skripty pro Blockly, což je knihovna pro tvorbu vizuálních editorů kódu. Tyto skripty jsou načítány z externích zdrojů na internetu pomocí následujících řádků v našem HTML kódu:

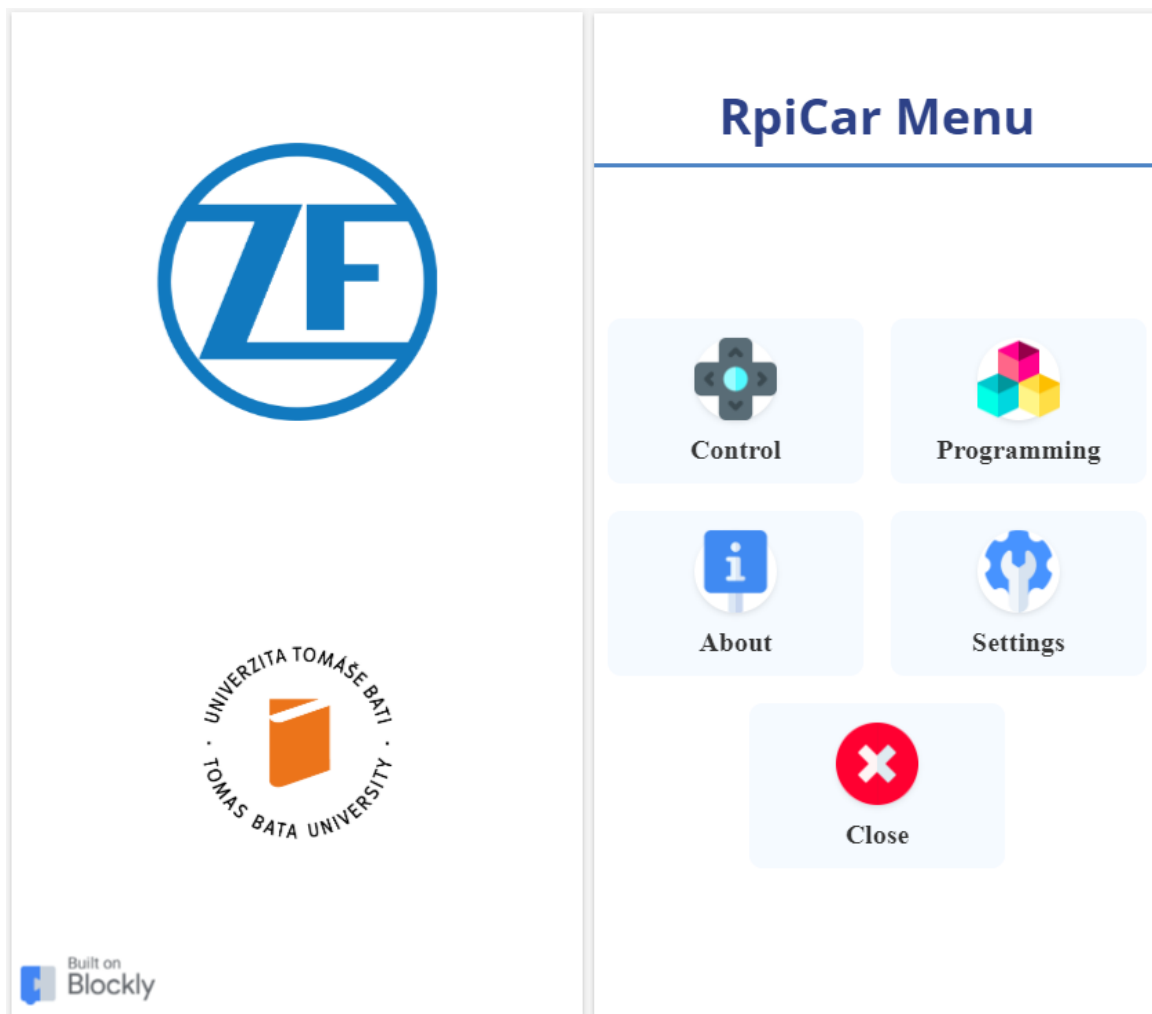
```
<script src="https://unpkg.com/blockly/blockly.min.js"></script>  
<script src="https://unpkg.com/blockly/javascript_compressed.js"></script>
```

Když webový prohlížeč zpracovává tyto řádky, automaticky stahuje a spouští odpovídající JS soubory. Tento proces probíhá dynamicky během načítání webové stránky, což umožňuje efektivnější využití zdrojů a rychlejší počáteční načítání stránky. Všechno je to za cenu nutnosti internetového připojení.

7.2.1 Představení vizuální stránky aplikace

Webová aplikace disponuje animovanou uvítací obrazovkou, kde je zmíněno logo Univerzity Tomáše Bati ve Zlíně a logo firmy ZF Engineering Plzeň spolu s logem knihovny Blockly.

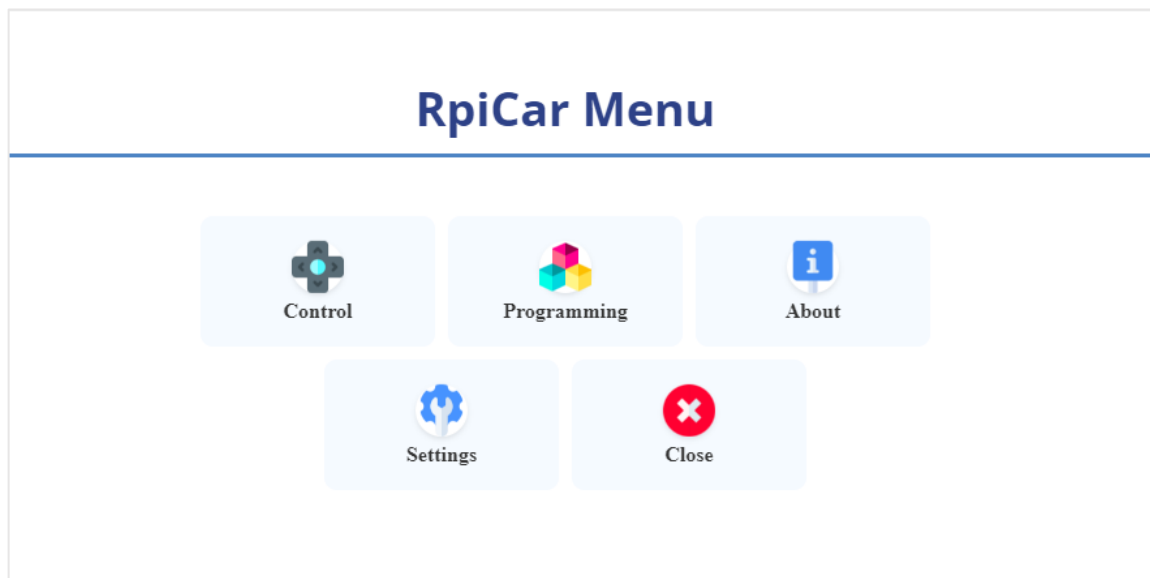
Ukázka některých stránek webové aplikace:



Obrázek 16 App – uvítací obrazovka, Obrázek 17 App – hlavní menu

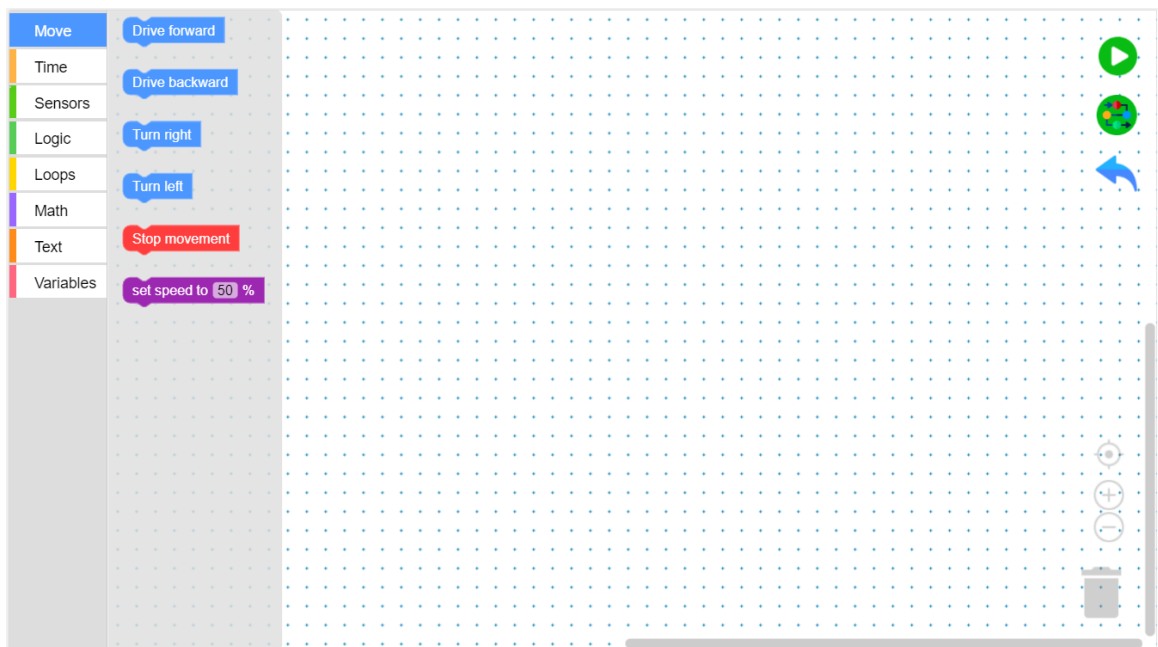
Dále se zobrazí hlavní menu (obrázek 17, 18 pro PC) celé aplikace, které se skládá z několika dlaždic, to jsou:

- **Control:** Slouží pro vstup na stránku pro manuální ovládání.
- **Programming:** Slouží pro vstup na stránku s grafickým programováním.
- **About:** Slouží pro vstup na stránku, která obsahuje krátký popis a zmínku o autorských právech.
- **Settings:** Slouží pro nastavení úhlu natočení ultrazvukového čidla.
- **Close:** Je prozatím nachystán pro budoucí využití.

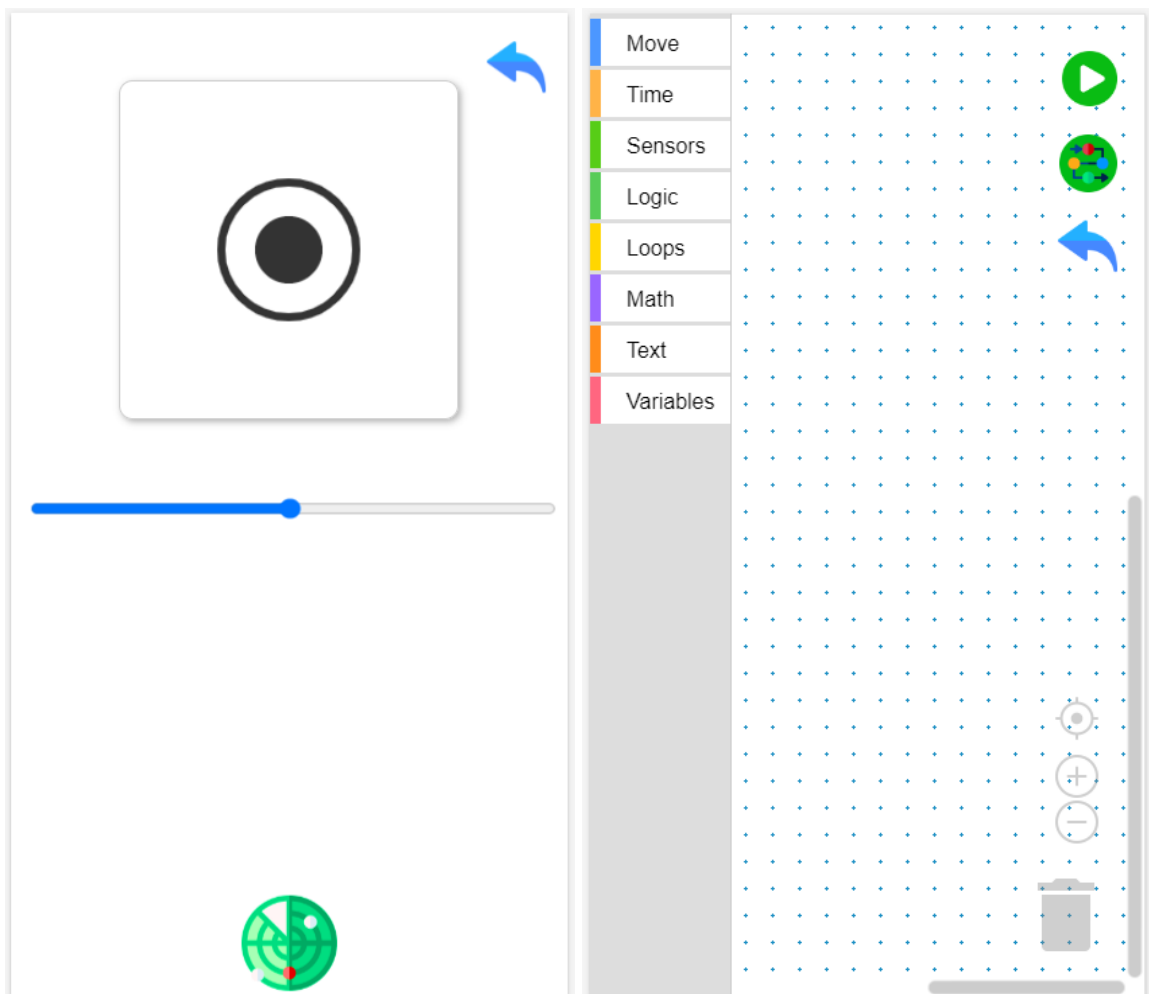


Obrázek 18 App – hlavní menu na PC

První odkaz **Control**, která slouží pro manuální ovládání. Je zde i jezdec (*slider*) pro nastavení rychlosti pohybu RpiCar-4WD. Čtverec je ohraničení pole joysticku pro ovládání směru jízdy. Tlačítko u spodní hrany obrazovky spouští skript pro ultrasonický senzor, který je zde umístěn pro test funkcionalit pro další rozšíření aplikace.



Obrázek 19 App – prac. plocha Blockly na PC



Obrázek 20 App – Manuální řízení, Obrázek 21 App – prac. plocha Blockly

Druhý odkaz **Programming** slouží pro grafické programování. Na této stránce (obrázek 19, 21) je většina prostoru vyhrazena pracovní ploše, pro umístění funkčních bloků, která se nachází uprostřed obrazovky. Na levé straně obrazovky se nachází panel nástrojů, který poskytuje různé nástroje pro grafické programování. V pravém horním rohu jsou umístěny ovládací prvky pro spuštění a opuštění stránky, ty se mění v závislosti na stavu programu a připojení k RpiCar-4WD. V pravém dolním rohu najdete symbol popelnice, který slouží k odstranění nechtěných bloků a prvky pro orientaci v pracovním prostoru.

7.2.2 Komunikace se serverem

Zde jsou popsány knihovny, které využívají při komunikaci se serverem – requireSocket.js, responsesocket.js a manual.js

7.2.2.1 *requireSocket.js*

Proměnná *requireWebsocket* je objekt, který slouží k připojení a komunikaci přes websocket. Jejím účelem je vytvořit spojení se serverem pomocí websocket protokolu.

Proměnná *connected* je logická proměnná, která označuje stav připojení. Na začátku je nastavena na hodnotu *false*, což značí, že připojení ještě nebylo navázáno.

- Funkce *requireWebsocket.connect* slouží k navázání spojení se serverem pomocí websocket. V této funkci je vytvořen nový *WebSocket* objekt a připojení se serverem je zahájeno. Po úspěšném navázání spojení je zobrazena zpráva "Require socket connect open..." a proměnná *connected* je nastavena na hodnotu *true*. Zároveň jsou zobrazeny tlačítka "Play" a "Step" pro ovládání přehrávání.
- Funkce *onmessage* je událostní funkce, která je volána při přijetí zprávy ze serveru. V této funkci je zalogována zpráva o stavu a proměnná *connected* je nastavena na hodnotu *true*.
- Funkce *onerror* je událostní funkce, která je volána při chybě ve spojení. V této funkci je zalogována chyba, proměnná *connected* je nastavena na hodnotu *false* a zobrazuje se okno s informací o odpojení. Tlačítka "Stop", "Play" a "Step" jsou skryta a zobrazuje se tlačítko pro opětovné připojení.
- Funkce *onclose* je událostní funkce, která je volána při uzavření spojení. V této funkci je zalogováno uzavření spojení, proměnná *connected* je nastavena na hodnotu *false* a zobrazuje se okno s informací o odpojení. Tlačítka "Stop", "Play" a "Step" jsou skryta a zobrazuje se tlačítko pro opětovné připojení.

7.2.2.2 *responseSocket.js*

Proměnná *responseWebsocket* je objekt, který slouží k připojení a komunikaci přes websocket. Jeho účelem je navázat spojení se serverem a přijímat odpovědi.

- Funkce *responseWebsocket.connect* slouží k navázání spojení se serverem pomocí *WebSocket* protokolu. V této funkci je vytvořen nový websocket objekt a připojení se serverem je zahájeno. Po úspěšném navázání spojení je do konzole zapsáno "Response socket connect open...".
- Funkce *onmessage* je událostní funkce, která je volána při přijetí zprávy ze serveru. Zpráva je předána ve formátu JSON a obsahuje data odpovědi.

7.2.2.3 manual.js

Tento skript pracuje s webovými sokety a komunikaci s robotem. Jeho hlavní úkolem je zasílat konkrétní hodnoty (skrze *Manual.sendValue*) robotu prostřednictvím webového socketu. Provádí se v něm i například filtrace přijatých data ze senzorů např z ultrazvukového senzoru vzdálenosti.

Pokud se tato funkce *sendValue* zavolá s hodnotami, které mají být odeslány, *sendValue* poté odesílá tyto hodnoty pomocí webového socketu. K tomu se používá metoda *send* na objektu *requireWebsocket.reqWs*. Tato metoda odesílá data pomocí webového socketu na server. Data, která se odesílají, jsou převedena na JSON string pomocí funkce *JSON.stringify* (Kód 1).

```
Manual.upArrowEvent = function () {
  Manual.sendValue['RC'] = 'forward'
  requireWebsocket.reqWs.send(JSON.stringify(Manual.sendValue))
}

Manual.downArrowEvent = function () {
  Manual.sendValue['RC'] = 'backward'
  requireWebsocket.reqWs.send(JSON.stringify(Manual.sendValue))
}

Manual.leftArrowEvent = function () {
  Manual.sendValue['RC'] = 'turn_left'
  requireWebsocket.reqWs.send(JSON.stringify(Manual.sendValue))
}

Manual.rightArrowEvent = function () {
  Manual.sendValue['RC'] = 'turn_right'
  requireWebsocket.reqWs.send(JSON.stringify(Manual.sendValue))
}

Manual.stop = function () {
  Manual.sendValue['RC'] = 'rest'
  requireWebsocket.reqWs.send(JSON.stringify(Manual.sendValue))
}
```

Kód 1 Funkce pro odeslání změny pohybu

7.2.3 Implementace grafického rozhraní pro programování

Pro základní funkci knihovny Blockly, se schopností dynamicky se přizpůsobit velikosti okna, je zapotřebí splnit tyto kroky:

1. Importovat knihovno do projektu, HTML souboru, existuje více způsobů, v tomto projektu se využívá unpkg.com, ze které se při spuštění stránky stahují potřebné komprimované skripty.
2. Definice *toolboxu*. Zde (Kód 2) je vidět samotná definice kategorie MOVE a dole pokračování definice další kategorie TIME.

```
var toolbox1 = {
  "kind": "categoryToolbox",
  "contents": [{
    "kind": "category",
    "name": "Move",
    "colour": "#4C97FF",
    "contents": [{
      "kind": "block",
      "type": "drive_forward"
    },
      {
        "kind": "block",
        "type": "drive_backward"
      },
      {
        "kind": "block",
        "type": "turn_right"
      },
      {
        "kind": "block",
        "type": "turn_left"
      },
      {
        "kind": "block",
        "type": "stop_movement"
      },
      {
        "kind": "block",
        "type": "change_speed"
      }
    ]
  },
  {
    "kind": "category",
    "name": "Time",
```

Kód 2 Ukázka definice toolboxu Blockly

3. Definice prostoru pro Blockly v HTML. Pro lepší představu více informací zde⁴. Je nutná definice těchto proměnných *blocklyArea* (Kód 3) a *blocklyDiv* (Kód 4):

⁴ <https://google.github.io/blockly-samples/examples/resizable-demo/index.html>

```
<body id="blocklyArea" class="document-body">  
  <script gwd-served-sizes="" type="applicatio
```

Kód 3 Ukázka definice prostoru blocklyArea

```
<div class="gwd-page-content gwd-div-1e1w">  
  <div id="blocklyDiv" style="position: absolute;"></div>  
  <script>
```

Kód 4 Ukázka definice prostoru blocklyDiv

4. Dalším krokem je umístění prvku *blocklyDiv* nad prvek *blocklyArea*. Za tímto účelem je nutné, aby *blocklyDiv* (Kód 5) byl nastaven na absolutní pozicování:

```
<div id="blocklyDiv" style="position: absolute;"></div>
```

Kód 5 Ukázka nastavení absolutního pozicování

5. Injektování (Kód 6) pracovního prostoru (workspace) s nepovinnými nastaveními zoomu, mřížky a dalších.

```
const outputArea = document.getElementById('blocklyArea');  
var blocklyDiv = document.getElementById('blocklyDiv');  
var demoWorkspace = Blockly.inject('blocklyDiv', {  
  toolbox: toolbox1,  
  zoom: {  
    controls: true,  
    wheel: true,  
    startScale: 1.0,  
    maxScale: 3,  
    minScale: 0.3,  
    scaleSpeed: 1.2,  
    pinch: true  
  },  
  trashcan: true,  
  grid: {  
    spacing: 20,  
    length: 3,  
    colour: '#1b8fc4',  
    snap: true  
  },  
  renderer: 'thrasos',  
  trashcan: true  
});
```

Kód 6 Injektování Blockly a další nastavení

Po těchto pěti základních krocích se začne pracovní plocha (*workspace*) Blockly zobrazovat na webové stránce.

6. Pro funkci automatického přizpůsobení velikosti pracovního prostředí knihovny Blockly je nutná implementace tohoto kódu (Kód 7), který počítá přesné rozměry pro *blocklyArea*.

```
var onresize = function(e) {  
  // Compute the absolute coordinates and dimensions of blocklyArea.  
  var element = blocklyArea;  
  var x = 0;  
  var y = 0;  
  do {  
    x += element.offsetLeft;  
    y += element.offsetTop;  
    element = element.offsetParent;  
  } while (element);  
  // Position blocklyDiv over blocklyArea.  
  blocklyDiv.style.left = x + 'px';  
  blocklyDiv.style.top = y + 'px';  
  blocklyDiv.style.width = blocklyArea.offsetWidth + 'px';  
  blocklyDiv.style.height = blocklyArea.offsetHeight + 'px';  
  Blockly.svgResize(demoWorkspace);  
  
  console.log('resize');  
};  
window.addEventListener('resize', onresize, false);  
onresize();
```

Kód 7 Skript pro automatickou změnu velikosti Blockly [24]

7.2.3.1 Definice bloků

Definice bloků v Blockly jsou klíčovým konceptem pro vytváření vlastních bloků, které mohou být použity v rámci vizuálního editoru. Každý blok má definici, která specifikuje jeho chování a vzhled.

Definice bloku je strukturovaný soubor informací, který obsahuje následující prvky:

- **Typ bloku:** Jedinečný identifikátor pro blok. Tento identifikátor je poté použit při vytváření instance bloku.
- **Název bloku:** Text, který je zobrazen na bloku. Tento text může být statický nebo může obsahovat speciální políčka pro vstupní hodnoty nebo výběr z nabídky.
- **Vstupy a pole bloku:** Každý blok může mít definované různé vstupy a pole, které určují, jaké hodnoty může blok přijímat a jakými parametry lze manipulovat.
- **Barvu bloku:** Barva, kterou bude mít blok v editoru.

- **Funkci bloku:** Funkci, která je spuštěna, když je blok použit v programu.

Definice bloku jsou obvykle napsány v jazyce JS, ale mohou být i ve formátu JSON. Prvním případem jsou pak přidány do Blockly prostřednictvím metody `Blockly.Blocks['<type>'] = { init: function() {...} }`. V této funkci `init()` je definován vzhled a chování bloku. Pro více detailních informací je vhodné navštívit webovou stránku Google developers⁵.

Zde (Kód 8) je ukázka kompletní definice bloku a generátoru pro jízdu vpřed (`drive_forward`) v JS.

```
Blockly.Blocks['drive_forward'] = {
  init: function() {
    this.appendDummyInput()
      .appendField("Drive forward");
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour("#4C97FF");
    this.setTooltip('Sends a message to the robot to drive straight');
    this.setHelpUrl('');
  }
};

Blockly.JavaScript['drive_forward'] = function(block) {
  var code = "upArrowEvent();\n";
  return code;
};
```

Kód 8 Ukázka definice bloku v JS

Zde je důležité zdůraznit, že kód reprezentovaný tímto blokem neposílá příkaz pro pohyb vozidla přímo na server. Místo toho generuje tzv. zástupný kód - kód, který je spuštěn v js-interpreteru v bezpečném, tzv. sandboxovém režimu.

V tomto kontextu je `upArrowEvent()`; pouze symbolická náhrada, která sama o sobě nemá funkčnost. Skrze API je následně volána funkce, která na základě tohoto zástupného kódu spustí efektivní kód.

⁵ <https://developers.google.com/blockly/guides/create-custom-blocks/define-blocks>

Tímto způsobem je zajištěno, že vygenerovaný kód, i když je spuštěn v sandboxovém módu, dokáže provést požadovanou akci, zároveň však zůstává izolován od ostatních částí systému pro maximální bezpečnost.

Pro úplnost také definice ve formátu JSON (ten je obecně preferovaný). Tady je ukázka definice bloku čekání (Kód 9) a také definice generátoru (Kód 10), který je stejný pro oba případy.

```
Blockly.defineBlocksWithJsonArray([[{
  "type": "wait_seconds1",
  "message0": "wait %1 milliseconds",
  "args0": [{
    "type": "field_number",
    "name": "SECONDS",
    "min": 0,
    "max": 60000,
    "value": 1000
  }],
  "previousStatement": null,
  "nextStatement": null,
  "colour": "#5b80a5"
}]);
```

Kód 9 Ukázka definice bloku ve formátu JSON

```
Blockly.JavaScript['wait_seconds1'] = function(block) {
  const seconds = Number(block.getFieldValue('SECONDS'));
  const code = 'waitForMseconds(' + seconds + ');\\n';
  return code;
};
```

Kód 10 Ukázka definice generátoru pro JSON i JS

7.2.3.2 Generování a spuštění kódu

Díky flexibilní architektuře nabízí Blockly možnost generovat výsledný kód do několika různých programovacích jazyků, včetně JavaScriptu, Pythonu, Lua, Dart a PHP. V našem případě generujeme kód do JS, který se následně spouští v instanci js-interpreteru.

Aby bylo možné využít tuto funkci, je nutné importovat specifickou generátorovou knihovnu pro daný jazyk spolu s hlavní knihovnou Blockly. V našem případě, jsme importovali JavaScriptový generátor pomocí následujícího řádku:

```
<script src="https://unpkg.com/blockly/javascript_compressed.js"></script>
```

Tímto způsobem je Blockly schopno převést bloky, které uživatel vytvoří v grafickém editoru, do JS kódu, který lze poté spustit nebo použít v dalším vývoji.

Nyní je tedy zapotřebí vykonat některé instrukce venku, a ne v sandboxu vytvořeným js-interpretem, k tomu nám slouží API (Kód 11,12), kterým lze tohoto efektu docílit.

Funkce *initInterpreterWaitForSeconds* přidává asynchronní metodu *waitForMseconds* do globálního rozsahu interpreteru. Tato funkce se pak stává dostupnou uvnitř virtuálního prostředí poskytnutého pro Blockly kód. Tato metoda je zvláštní tím, že je asynchronní – to znamená, že její provedení je odloženo o určitý čas, který je určen argumentem *timeInSeconds*. Tento časový interval je vyjádřen v milisekundách.

V kontextu naší aplikace *waitForMseconds* poskytuje schopnost Blockly programů "čekat" určitou dobu před pokračováním v další činnosti. Toto je užitečné pro řízení tempa vykonávaných akcí. Například, může být použito pro zastavení pohybu robota na určitou dobu.

Funkce *initInterpreterWaitForSeconds* využívá API Blockly JavaScript, konkrétně metodu *addReservedWords*, aby se ujistila, že název funkce *waitForMseconds* nebude kolidovat s názvy proměnných použitých v Blockly kódu.

Nakonec je metoda *waitForMseconds* přidána do globálního rozsahu interpreteru prostřednictvím metody *setProperty*. Tím se stane dostupnou pro Blockly kód.

```
function initInterpreterWaitForSeconds(interpreter, globalObject) {
  // Ensure function name does not conflict with variable names.
  Blockly.JavaScript.addReservedWords('waitForMseconds');

  const wrapper = interpreter.createAsyncFunction(
    function(timeInSeconds, callback) {
      // Delay the call to the callback.
      setTimeout(callback, timeInSeconds);
    });
  interpreter.setProperty(globalObject, 'waitForMseconds', wrapper);
}
```

Kód 11 Definice API pro WaitForSeconds

```
function initApi(interpreter, globalObject) {
  // Add an API function for the alert() block, generated for "text_print" blocks.
  const wrapperAlert = function alert(text) {
    text = arguments.length ? text : '';
    showBlocklyTextWindow(text);
    console.log('\n' + text);
  };
  interpreter.setProperty(globalObject, 'alert',
    interpreter.createNativeFunction(wrapperAlert));

  // Add an API function for the prompt() block.
  const wrapperPrompt = function prompt(text) {
    return window.prompt(text);
  };
  interpreter.setProperty(globalObject, 'prompt',
    interpreter.createNativeFunction(wrapperPrompt));

  // Add an API function for highlighting blocks.
  const wrapper = function(id) {
    id = String(id || '');
    return highlightBlock(id);
  };
  interpreter.setProperty(globalObject, 'highlightBlock',
    interpreter.createNativeFunction(wrapper));

  // Add an API for the wait block.
  initInterpreterWaitForSeconds(interpreter, globalObject);
  // Add an other APIs
  initInterpreterMoveForward(interpreter, globalObject);
  initInterpreterMoveBackward(interpreter, globalObject);
  initInterpreterTurnRight(interpreter, globalObject);
  initInterpreterTurnLeft(interpreter, globalObject);
  initInterpreterStopMotion(interpreter, globalObject);
  initInterpreterSpeedChange(interpreter, globalObject);
  initInterpreterDistance(interpreter, globalObject);
}
```

Kód 12 Inicializace API

Spuštění generace a samotného kódu (Kód 13) probíhá funkci *startCode*, která řídí celý proces spuštění Blockly kódu. Tato funkce je klíčová pro spuštění a řízení interpretace Blockly kódu.

Pokud interpret není nastaven (tj. *myInterpreter* je null), funkce *resetStepUi(true)* se zavolá, aby se vyčistil výstup programu. Poté se z Blockly *workspace* vytvoří JavaScriptový kód pomocí *Blockly.JavaScript.workspaceToCode(demoWorkspace)*.

Následuje nastavení časovače, který umožňuje resetování hodnoty *outputArea.value* před zahájením interpretace. Po tomto resetu se vytvoří nová instance interpretu s nejnovějším kódem a API inicializovaným funkcí *initApi*.

Vnitřní funkce *runner* je následně spuštěna. Tato funkce zajišťuje opakované spouštění interpretu a sleduje, zda ještě existují další instrukce k vykonání. Pokud interpret narazí na asynchronní volání v kódu, znamená to, že musí čekat, dokud tato asynchronní operace bude dokončena, aby mohl pokračovat v provádění dalšího kódu. To je stav, kdy je interpret "zablokovaný" - ne proto, že by byl zaseknutý nebo nefunkční, ale proto, že čeká na dokončení. V takovém případě funkce *runner* nechá interpret, aby pokračoval v čekání, a nastaví si časovač pomocí *setTimeout*, aby se pokusila spustit interpret znovu po krátkém časovém intervalu. To je, co se myslí frází "Try again later" v kódu (Kód 12). Jedná se o způsob, jak se vyrovnat s asynchronní povahou některých operací v JavaScriptu. Pokud už není žádný další kód k vykonání, program je považován za dokončený a aktualizuje se UI.

```
function startCode() {
  if (!myInterpreter) {
    // Clear the program output.
    resetStepUi(true);
    const latestCode = Blockly.JavaScript.workspaceToCode(demoWorkspace);
    // In a timeout to allow the outputArea.value to reset first.
    setTimeout(function() {
      myInterpreter = new Interpreter(latestCode, initApi);

      function runner() {
        if (myInterpreter) {
          const hasMore = myInterpreter.run();
          if (hasMore) {
            // Execution is currently blocked by some async call.
            // Try again later.
            runnerPid = setTimeout(runner, 10);
          } else {
            // Program is complete.
            console.log('\n\n<< Program complete >>');
            $('#play-button').show();
            $('#stop-button').hide();
            $('#step-button').show();
            resetStepUi(false);
          }
        }
      }
      runner();
    }, 1);
  }
  return;
}
```

Kód 12 Ukázka celého procesu generování a spuštění

Aplikace obsahuje i funkci pro postupné vykonávání instrukcí, známou jako *code stepping*. Tato funkce umožňuje uživatelům detailně sledovat a ladit jednotlivé kroky programu. Tato možnost je užitečná zejména pro odhalování a řešení problémů v kódu.

Nicméně, v současnosti nejsou všechny bloky plně kompatibilní s tímto způsobem vykonávání. Problémy mohou nastat zejména u bloků, které se vykonávají asynchronně. Asynchronní bloky obecně zahrnují časově náročné operace, například zpracování bloků reprezentujících čtení informací z čidel. Proto je implementována funkce *stepCode()*, která je podobná funkci *startCode()*. Tato funkce je však inicializována a použita s modifikovanými API, což jí umožňuje lépe se vypořádat s asynchronními bloky. Práce na optimalizaci a zlepšení kompatibility krokovací funkce s asynchronními bloky není dokončená.

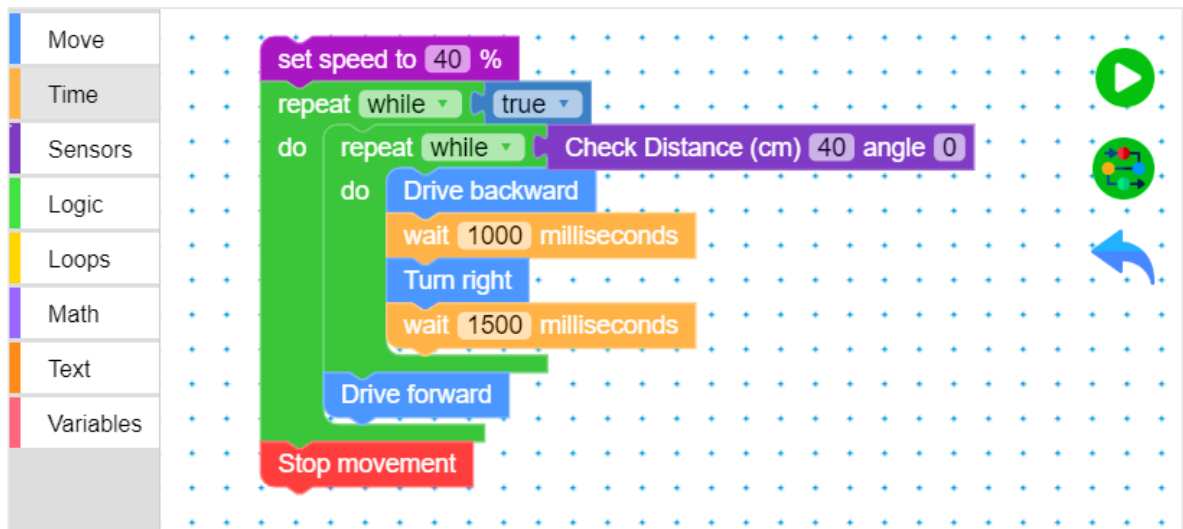
8 SOUČASNÝ STAV, VYUŽITÍ A PLÁNY DO BUDOUCNA

8.1 Současný stav

Aplikace je funkční a připravena pro použití. Následuje příkladná ukázka použití, ukázka je zaměřena pouze na grafické programování jakožto hlavní předmět tohoto vypracování.

8.1.1 Ukázka programu

Na obrázku 22 je poskládaný následující program:



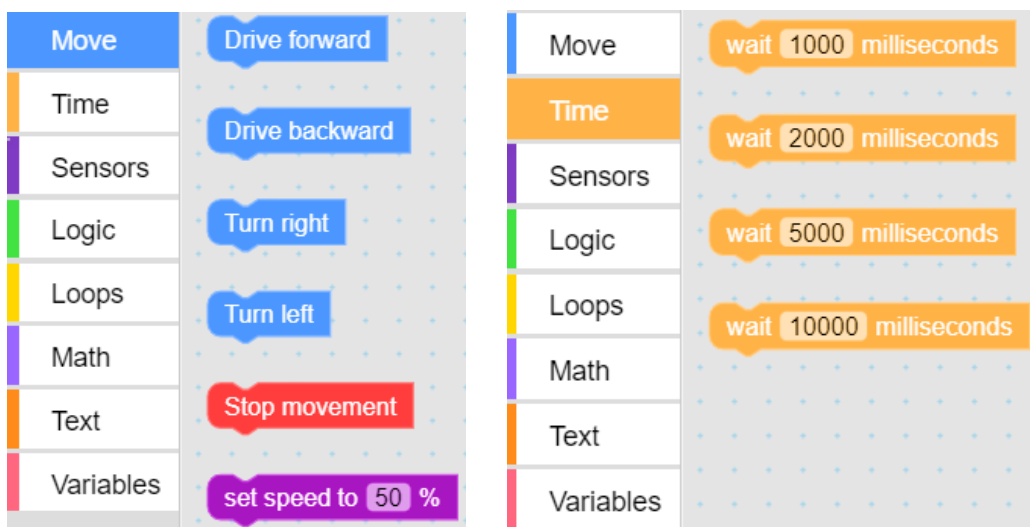
Obrázek 22 Ukázka jednoduchého programu

Nyní po stlačení zeleného tlačítka play, v pravém horním rohu, se začne s vykonáváním programu, a to v následujících krocích:

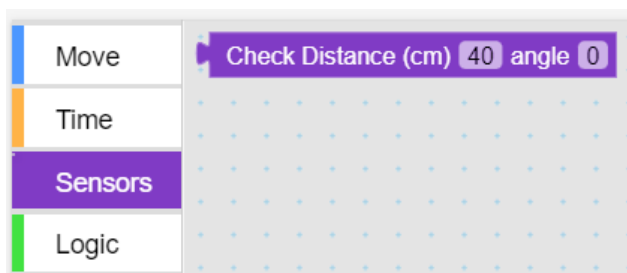
1. Rychlost pohybu se nastaví na 40 %.
2. Vstup do nekonečné smyčky **WHILE**(true).
3. Vstup do smyčky **WHILE** s podmínkou **CHECK DISTANCE**, ta přečte aktuální vzdálenost od překážky při natočení ultrazvukového čidla pod úhlem 0°.
 - a. Pokud je překážka vzdálená méně než 40 cm (true), provádí se vnořený cyklus **WHILE**. Auto začne po dobu 1 vteřiny couvat příkazem **DRIVE BACKWARD**, poté 1,5 vteřiny zatačet doprava **TURN RIGHT**.
 - b. Pokud není překážka ve vzdálenosti menší než 40 cm (false), přeskočí se vnořená smyčka **WHILE**.
4. Auto pokračuje v jízdě dopředu blokem **DRIVE FORWARD**.
5. Opakování nekonečné smyčky **WHILE**(true), blok **STOP MOVEMENT** se tedy nikdy nevykoná.

8.1.2 Toolbox

Ukázky bloků v toolboxu Blockly je představeny níže. Očekává se, že v budoucnu přibudou další bloky, které rozšíří možnosti grafického programování. Na obrázcích 23, 24 a 25 jsou uvedeny pouze klíčové stavební bloky pro porozumění – ostatní bloky jsou uvedeny v příloze P6.



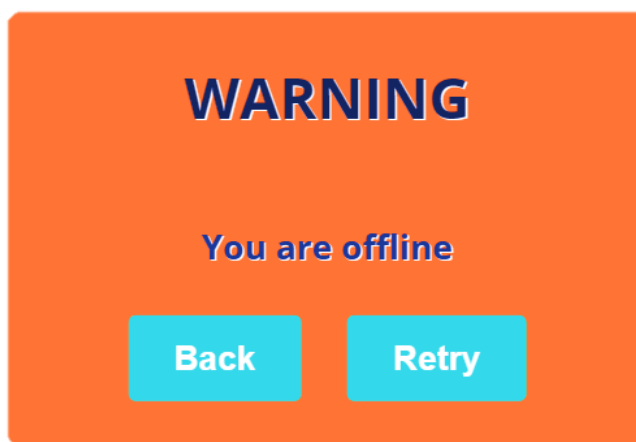
Obrázek 23 Kategorie Move, Obrázek 24 Kategorie Time



Obrázek 25 kategorie Sensors

8.1.3 Hlídání spojení a asistence při skladbě bloků

Při výpadku spojení s modelem auta se zobrazí následující dialogové okno (obrázek 26). Toto okno nabízí možnosti pro obnovení spojení (Retry) nebo ignorování tohoto problému (Back). Pokud uživatel zvolí možnost ignorování, stále je možné se pokusit o obnovení spojení. Toto lze provést na stránce Programming, kde se v pravém horním rohu zobrazí tlačítko (Obrázek 27) pro obnovení spojení.



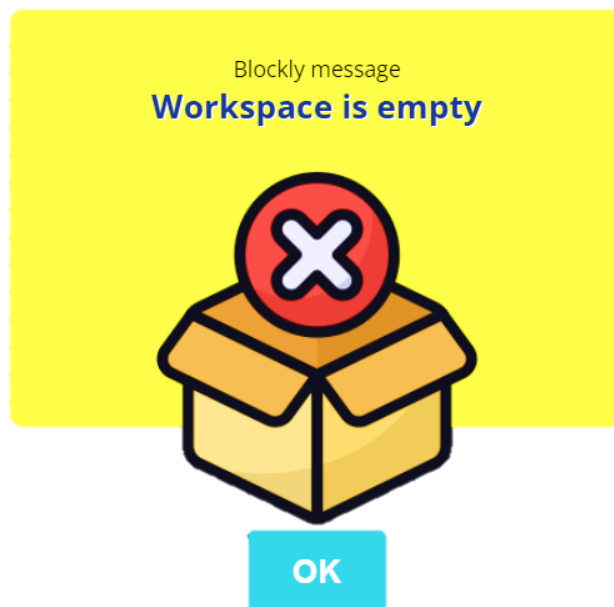
Obrázek 26 Okno výpadku spojení



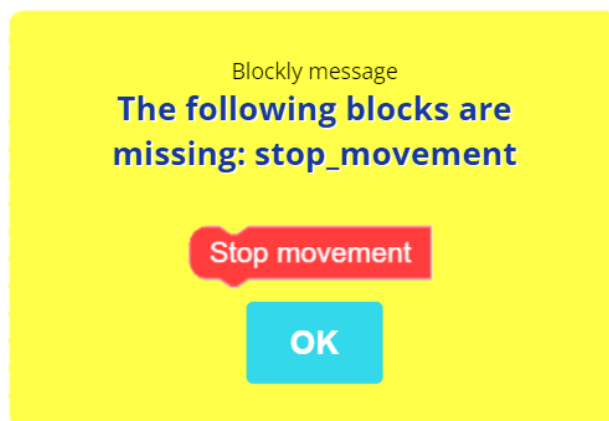
Obrázek 27 Tlačítko pro navázání spojení

Při spuštění programu sestaveného pomocí bloků jsou provedeny dvě kontroly. Nejprve je zkontrolováno, zda pracovní plocha není prázdná, tedy bez jakéhokoliv bloku. Pokud je pracovní plocha prázdná, zobrazí se upozornění (Obrázek 28). Dále je kontrolováno, zda kód

obsahuje blok Stop Movement. Pokud tento blok není obsažen, zobrazí se dialogové okno upozorňující na tuto skutečnost (viz Obrázek 29).



Obrázek 28 Dialogové okno při prázdné pracovní ploše



Obrázek 29 Dialogové okno pro chybějící blok Stop movement

8.2 Využití aplikace v propagaci

Aplikace již prošla zkouškou na několika akcích, včetně Veletrhu pracovních příležitostí iKariéra IAESTE UTB Zlín 2023 pořádané přímo na Fakultě aplikované informatiky Univerzity Tomáše Bati ve Zlíně. Na této akci se podařilo prokázat, že tento typ aplikace má reálný potenciál a je schopen vzbudit zájem u potenciálních zaměstnanců.

8.3 Návrhy na vylepšení

V kontextu této práce a vývoje naší nové webové aplikace je důležité zmínit, že tento projekt není konečným cílem, ale spíše výchozím bodem pro další rozvoj a inovace. I přes pečlivé plánování je nevyhnutelné, že v průběhu času budou vyvstávat nové požadavky a příležitosti pro zlepšení a rozšíření naší aplikace.

Následující oblasti by mohly být součástí budoucího pokračování prací na tomto projektu:

1. Rozšiřování funkcí: Na základě zpětné vazby uživatelů a nových požadavků na funkčnost můžeme průběžně rozšiřovat a upravovat funkce naší aplikace.
2. Integrace s dalšími systémy: Jak se vyvíjí technologické prostředí, může být užitečné integrovat naši aplikaci s dalšími systémy nebo platformami pro poskytnutí lepších služeb našim uživatelům.
3. Zlepšení bezpečnosti: Bezpečnost je oblast, která vyžaduje neustálou pozornost a údržbu. Během dalšího vývoje budeme neustále sledovat a vylepšovat tyto aspekty naší aplikace.
4. Údržba a podpora: Jakmile je aplikace spuštěna a používána, je důležité zajistit průběžnou podporu a údržbu, aby zůstala relevantní, funkční a bezpečná.
5. Zlepšení prostorové orientace: To pomocí gyroskopu. S využitím pokročilejších algoritmů pro kalibraci gyroskopu by mohla být zlepšena přesnost a spolehlivost orientace v prostoru.
6. Raspberry Pi kamera: Použití kamery rozšíří možnosti projektu tím, že umožní vizuální zpracování obrazu a lepší interakci s okolím. S využitím pokročilých algoritmů pro zpracování obrazu, jako je rozpoznávání obličejů, sledování pohybu nebo detekce objektů, by bylo možno získat zajímavé pokročilejší funkce.
7. Kontrola kódu: Sofistikovanější kontrola kódu po spuštění, například kontrola kódu proveditelnosti (Upozornění na příkazy, které se nikdy neprovedou).

ZÁVĚR

Cílem této práce bylo navrhnout a realizovat aplikaci pro řízení robotického auta. Jako robotické auto bylo použito PiCar-4WD, složené ze stavebnice, která byla zakoupena a poskytnuta firmou ZF Engineering Plzeň s.r.o.

Pro přímé řízení tohoto auta se využívá Raspberry Pi, na kterém běží skripty, která bezprostředně snímají signály z jednotlivých snímačů a vykonávají příkazy, které ovlivňují aktuální chování tohoto auta. Vedle toho byla vytvořena uživatelská aplikace, která umožňuje uživateli prostřednictvím vizuálního programování vytvářet vlastní programové kódy. Tyto kódy komunikují se zmíněnými skripty na Raspberry Pi. S pomocí této aplikace lze vytvářet vlastní programy, kterými lze předem definovat chování robotického auta v různých situacích. Koncepce této aplikace je navržena tak, že se při její tvorbě uživatel může učit logickému programování.

Teoretická část práce popisuje konstrukci robotického auta, jeho snímače a možnosti jeho řízení. Další kapitola se věnuje architektuře mikropočítače Raspberry Pi se zaměřením na verzi 4B, která byla použita pro řízení tohoto auta. Velká pozornost je také věnována možnosti komunikace. Poslední kapitola teoretické části se věnuje softwarovým nástrojům pro tvorbu mobilních aplikací. Zaměřena pozornost byla zejména na ty nástroje, které byly použity v praktické části práce.

Na začátku praktické části práce je popsáno sestavení robotického auta ze součástek dodanými ve stavebnici včetně vlastního řešení napájení. Dále byly analyzovány požadavky na aplikaci a v souvislostmi s možnostmi tohoto auta byly realizovány skripty, které běží na Raspberry Pi. Následně byla vytvořena webová aplikace optimalizována pro mobilní telefony s intuitivním uživatelským rozhraním, ve které si uživatel může vytvářet vlastní řídicí aplikace.

Aplikace je plně funkční a byla vyzkoušena studenty na Veletrhu pracovních příležitostí v dubnu 2023 na FAI UTB. Zároveň však nabízí velký potenciál pro další práci. Do budoucna je tedy plánováno její další vylepšování a rozšiřování.

SEZNAM POUŽITÉ LITERATURY

- [1] Sunfounder: piCar-4wd Car Kit for Raspberry Pi [online]. Shenzhen: © SunFounder, 2022 [cit. 2022-11-24]. Dostupné z: <https://docs.sunfounder.com/projects/picar-4wd/en/latest/>
- [2] Raspberry Pi Smart Car Kit (Picar-4WD) for Beginner. In: Sunfounder [online]. [cit. 2023-04-22]. Dostupné z: https://cdn.shopify.com/s/files/1/0474/7729/3217/products/CN0220D-01_600x.jpg?v=1599212913
- [3] Package List. In: Sunfounder [online]. [cit. 2023-04-22]. Dostupné z: https://cdn.shopify.com/s/files/1/0474/7729/3217/products/CN0220D-08_600x.jpg?v=1607589300
- [4] Raspberry Pi Smart Car Kit (Picar-4WD) for Beginner. In: Sunfounder [online]. [cit. 2023-04-22]. Dostupné z: https://cdn.shopify.com/s/files/1/0474/7729/3217/products/CN0220D-08_600x.jpg?v=1607589300
- [5] Component List [online]. In: . SunFounder Revision 3b30fb76. ©2021, s. 6 [cit. 2023-05-04]. Dostupné z: https://github.com/sunfounder/sf-pdf/raw/master/components_list/picar-4wd-component-list.pdf
- [6] UPTON, Eben a Gareth HALFACREE. Raspberry Pi: uživatelská příručka. Brno: Computer Press, 2013. ISBN 978-80-251-4116-8
- [7] Raspberry Pi 4 announced with quad-core. In: Neowin [online]. 2019, 24 June 2019 [cit. 2023-05-22]. Dostupné z: https://cdn.neowin.com/news/images/uploaded/2019/06/1561340114_annotated_pi_story.jpg
- [8] BCM2711 ARM Peripherals [online]. In: . 2022, s. 166 [cit. 2023-05-07]. Dostupné z: <https://datasheets.raspberrypi.com/bcm2711/bcm2711-peripherals.pdf>
- [9] Raspberry Pi hardware: GPIO and the 40-pin Header. In: Raspberrypi [online]. 2023 [cit. 2023-05-08]. Dostupné z: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html>
- [10] 23.2 Sériová sběrnice I2C. <https://maly.gitbook.io> [online]. 2020 [cit. 2023-05-08]. Dostupné z: https://maly.gitbook.io/hradla-volty-jednocipy/23_seriova_komunikace/232_seriova_sbernice_i2c

- [11] Raspberrypi: raspberry Pi OS: datasheet: configuration: the config.txt file : documentation [online]. Cambridge: ©Raspberry Pi, 2022 [cit. 2022-11-21]. Dostupné z: <https://www.raspberrypi.com>
- [12] Blockly: samples: references: guides [online]. Mountain View: Mountain View, 2022 [cit. 2022-11-24]. Dostupné z: <https://developers.google.com/blockly>
- [13] HORTON, John. Android Programming for Beginners. 3rd edition. Birmingham: ©Packt Publishing, 2021. ISBN 9781800563438.
- [14] Android Developers: develop android games: android studio: docs [online]. Mountain View: Mountain View, 2022 [cit. 2022-11-24]. Dostupné z: <https://developer.android.com>
- [15] JS-Interpreter Documentation. Neil's Open-Source Software [online]. 2021 [cit. 2023-05-10]. Dostupné z: <https://neil.fraser.name/software/JS-Interpreter/docs.html>
- [16] Sunfounder: picar-4wd. Github [online]. 2022 [cit. 2023-05-11]. Dostupné z: <https://github.com/sunfounder/picar-4wd.git>
- [17] Úvod do programování: 3.13.4 Knihovny [online]. [cit. 2023-05-15]. Dostupné z: <https://mrlvsb.github.io/upr-skripta/c/modularizace/knihovny.html>
- [18] ABOUT SUNFOUNDER. Sunfounder [online]. Shenzhen: © SunFounder [cit. 2023-05-15]. Dostupné z: <https://www.sunfounder.com/pages/about-us>
- [19] Hard13.jpeg. In: Sunfounder [online]. © Copyright 2021 [cit. 2023-05-16]. Dostupné z: https://docs.sunfounder.com/projects/picar-4wd/en/latest/_images/hard13.jpeg
- [20] Hardware Assembling. <https://docs.sunfounder.com>: Docs [online]. 2021 [cit. 2023-05-16]. Dostupné z: https://docs.sunfounder.com/projects/picar-4wd/en/latest/hardware_assembling.html#
- [21] Generating and Running JavaScript: JS-Interpreter [online]. [cit. 2023-05-16]. Dostupné z: <https://developers.google.com/blockly/guides/app-integration/running-javascript>
- [22] Raspberry Pi mění svět: Seznamte se s nejzajímavějším počítačem dneška. Hospodářské noviny: ihned [online]. 2016, 2016, 1 [cit. 2023-04-24]. ISSN 1213-7693. Dostupné z: <http://tech.ihned.cz/geekosfera/c1-65195330-raspberry-pi-meni-svet-seznamte-se-s-nejzajimavejsim-pocitacem-dneska>

- [23] Flaticon: Icons [online]. Freepik Company, c2010-2023 [cit. 2023-05-21]. Dostupné z: <https://www.flaticon.com>
- [24] Google developers: Position the workspace [online]. 2022 [cit. 2023-05-21]. Dostupné z: <https://developers.google.com/blockly/guides/configure/web/resizable>
- [25] Vývoj: Vývoj bez kompromisu s procesory ARM. DPS [online]. 6/2011 [cit. 2023-05-22]. Dostupné z: <https://www.dps-az.cz/vyvoj/id:11452/vyvoj-bez-kompromisu-s-procesory-arm>
- [26] Blog: Everything you need to know about Raspberry Pi HATs & pHATs. Okdo [online]. [cit. 2023-05-23]. Dostupné z: <https://www.okdo.com/blog/your-guide-to-hats-and-phats/>
- [27] SMBus Protocol: SMBus Protocol Introduction: [online]. 13 April 2022 [cit. 2023-05-23]. Dostupné z: <https://prodigytechno.com/smbus-protocol/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

IT	Informační Technologie.
IoT	Internet of Things.
VPLs	Visual Programming Languages
DIY	Do It Yourself
STEM	Science Technology Engineering Mathematics
SD	Secure Digital
4WD	Four Wheel Drive
HAT	Hardware Attached on Top
RPi	Raspberry Pi
Wi-Fi	Wireless Fidelity
iOS	iPhone Operating System
USB	Universal Serial Bus
IR	Infrared
SBC	Single Board Computer
LAN	Local Area Network
HDMI	High Definition Multimedia Interface
ARM	Advanced RISC Machines
RISC	Reduced Instruction Set Computing
MP3	MPEG-1 Audio Layer 3
OS	Operating System
LPDDR	Low Power Double Data Rate
SDRAM	Synchronous Dynamic Random-Access Memory
IEEE	Institute of Electrical and Electronics Engineers
BLE	Bluetooth Low Energy

GPIO	General Purpose Input/Output
PoE	Power over Ethernet
PCM	Phase Change Memory
I ² C	Inter-Integrated Circuit
DMA	Direct Memory Access
PWM	Pulse Width Modulation
UART	Universal Asynchronous Receiver-Transmitter
A2DP	Advanced Audio Distribution Profile
AVRCP	Audio/Video Remote Control Profile
SPI	Serial Peripheral Interface
CEC	Consumer Electronics Control
HDCP	High-bandwidth Digital Content Protection
EEPROM	Electrically Erasable Programmable Read-Only Memory
GSM	Global System for Mobile Communications
GPS	Global Positioning System
TV	Television
LoRa-WAN	Long Range Wide Area Network
LED	Light Emitting Diode
TWI	Two Wire Interface
SDA	Serial Data Line
SCL	Serial Clock Line
ACK	Acknowledgement
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
μC	Micro Controller

SMBus	System Management Bus
HTML	Hyper Text Markup Language
JS	JavaScript
PHP	Hypertext Preprocessor
Li-Ion	Lithium-Iont
Li-Pol	Lithium Polymer
BMS	Battery Management Systém
PD	Power Delivery
CC-CV	Constant Current-Constant Voltage
DC-DC	Direct Current to Direct Current
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation

SEZNAM OBRÁZKŮ

Obrázek 1 Raspberry Pi Smart Car Kit (PiCar-4WD) [2]	10
Obrázek 2 Kompletní obsah balení stavebnice [4]	12
Obrázek 3 Hlavní stránka originální aplikace SunFounder [16]	13
Obrázek 4 Rpi4 B [7].....	17
Obrázek 5 RPi a 4WD HAT složený stav.....	20
Obrázek 6 Vizualizace spojení Rpi a 4WD HAT[1]	21
Obrázek 7 Přehled 40-pinového headeru [9]	21
Obrázek 8 Blokové schéma sběrnice I ² C [10]	23
Obrázek 9 Krabice stavebnice PiCar-4WD	29
Obrázek 10 USB-C, IC IP2721 pro fyzickou vrstvu USB	31
Obrázek 11 DC-DC step down zdroj s CC-CV	31
Obrázek 12 Shottkyho dioda B34 – 3A max	32
Obrázek 13 Kompletní napájecí soustava.....	33
Obrázek 14 RpiCar-4WD originální stav bez úprav [19].....	33
Obrázek 15 PiCar-4WD s novou baterií a nabíjením	34
Obrázek 16 App – uvítací obrazovka, Obrázek 17 App – hlavní menu	43
Obrázek 18 App – hlavní menu na PC	44
Obrázek 19 App – prac. plocha Blockly na PC	44
Obrázek 20 App – Manuální řízení, Obrázek 21 App – prac. plocha Blockly	45
Obrázek 22 Ukázka jednoduchého programu.....	57
Obrázek 23 Kategorie Move, Obrázek 24 Kategorie Time	58
Obrázek 25 kategorie Sensors.....	58
Obrázek 26 Okno výpadku spojení.....	59
Obrázek 27 Tlačítko pro navázání spojení	59
Obrázek 28 Dialogové okno při prázdné pracovní ploše.....	60

SEZNAM KÓDŮ

Kód 1 Funkce pro odeslání změny pohybu	47
Kód 2 Ukázka definice toolboxu Blockly.....	48
Kód 3 Ukázka definice prostoru BlocklyArea	49
Kód 4 Ukázka definice prostoru BlocklyDiv	49
Kód 5 Ukázka nastavení absolutního pozicování	49
Kód 6 Injektování Blockly a další nastavení	49
Kód 7 Skript pro automatickou změnu velikosti Blockly [24].....	50
Kód 8 Ukázka definice bloku v JS.....	51
Kód 9 Ukázka definice bloku ve formátu JSON	52
Kód 10 Ukázka definice generátoru pro JSON i JS.....	52
Kód 11 Definice API pro WaitForSeconds	53
Kód 12 Ukázka celého procesu generování a spuštění.....	55

SEZNAM TABULEK

Tabulka 1 Přehled a porovnání SPI a I ² C [10].....	23
--	----

SEZNAM PŘÍLOH

- P1 Obsah příloženého CD
- P2 Detail součástí stavebnice picar-4wd 1 [5]
- P3 Detail součástí stavebnice picar-4wd 2 [5]
- P4 Detail součástí stavebnice picar-4wd 3 [5]
- P5 Zvětšený obrázek 14 [19]
- P6 Ukázka ostatních kategorií toolboxu

PŘÍLOHA P 1: OBSAH PŘILOŽENÉHO CD

CD je uloženo v kapse na vnitřní straně zadní desky.

Jeho obsah:

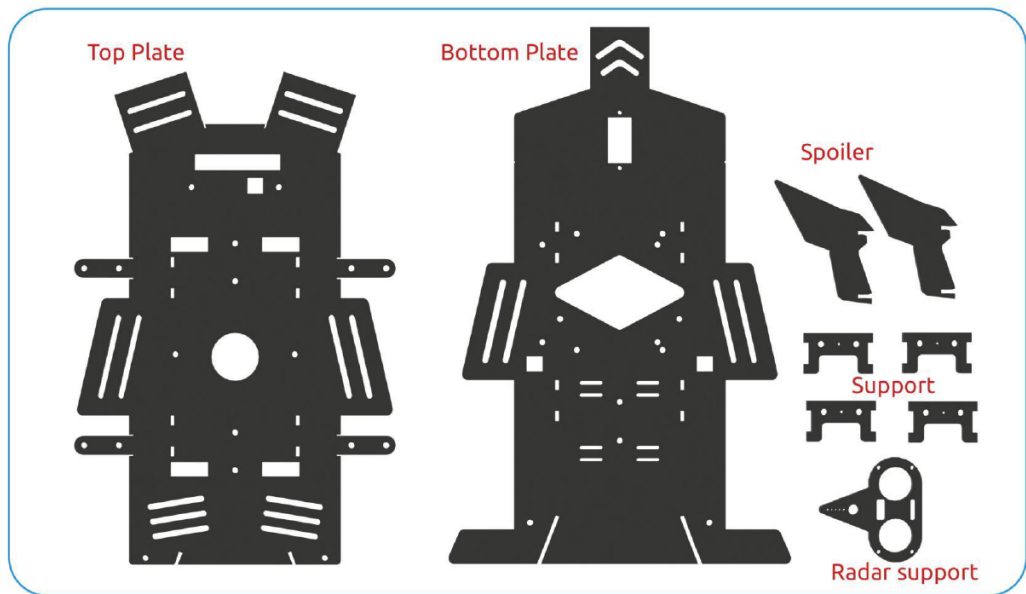
V rootu se nachází:

- Soubor **fulltext.pdf** (tento dokument)
- Složka **sources** se zdrojovými kódy aplikace a skripty pro RPi.

PŘÍLOHA P 2: DETAIL SOUČÁSTÍ STAVEBNICE PICAR-4WD 1

Component List

Structural Plates

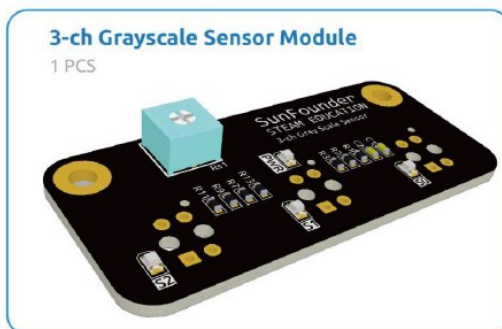
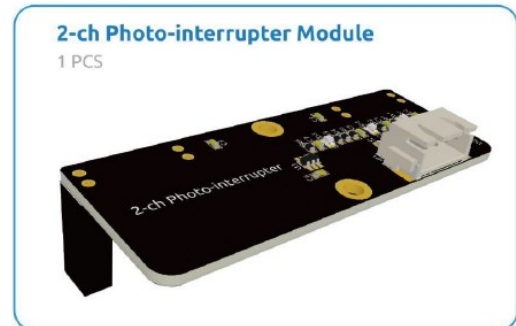
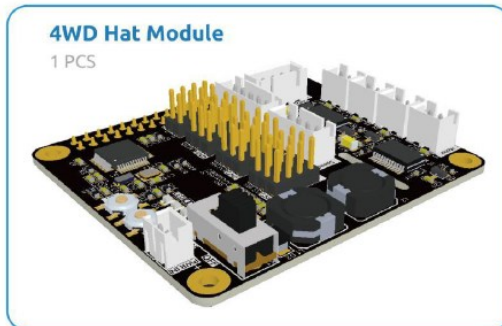


Mechanical Parts



PŘÍLOHA P 3: DETAIL SOUČÁSTÍ STAVEBNICE PICAR-4WD 2

Electronic Parts



Other Parts



PŘÍLOHA P 4: DETAIL SOUČÁSTÍ STAVEBNICE PICAR-4WD 3

Cable Tie

4 PCS



Screwdriver

1 PCS



Wheels

4 PCS



Battery Holder

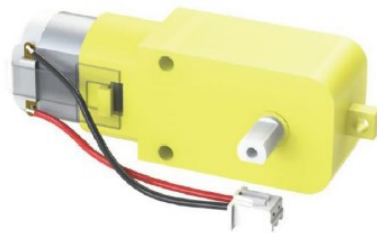
1 PCS



Driving Parts

Motor

4 PCS



Servo

1 PCS



Self- prepared Part

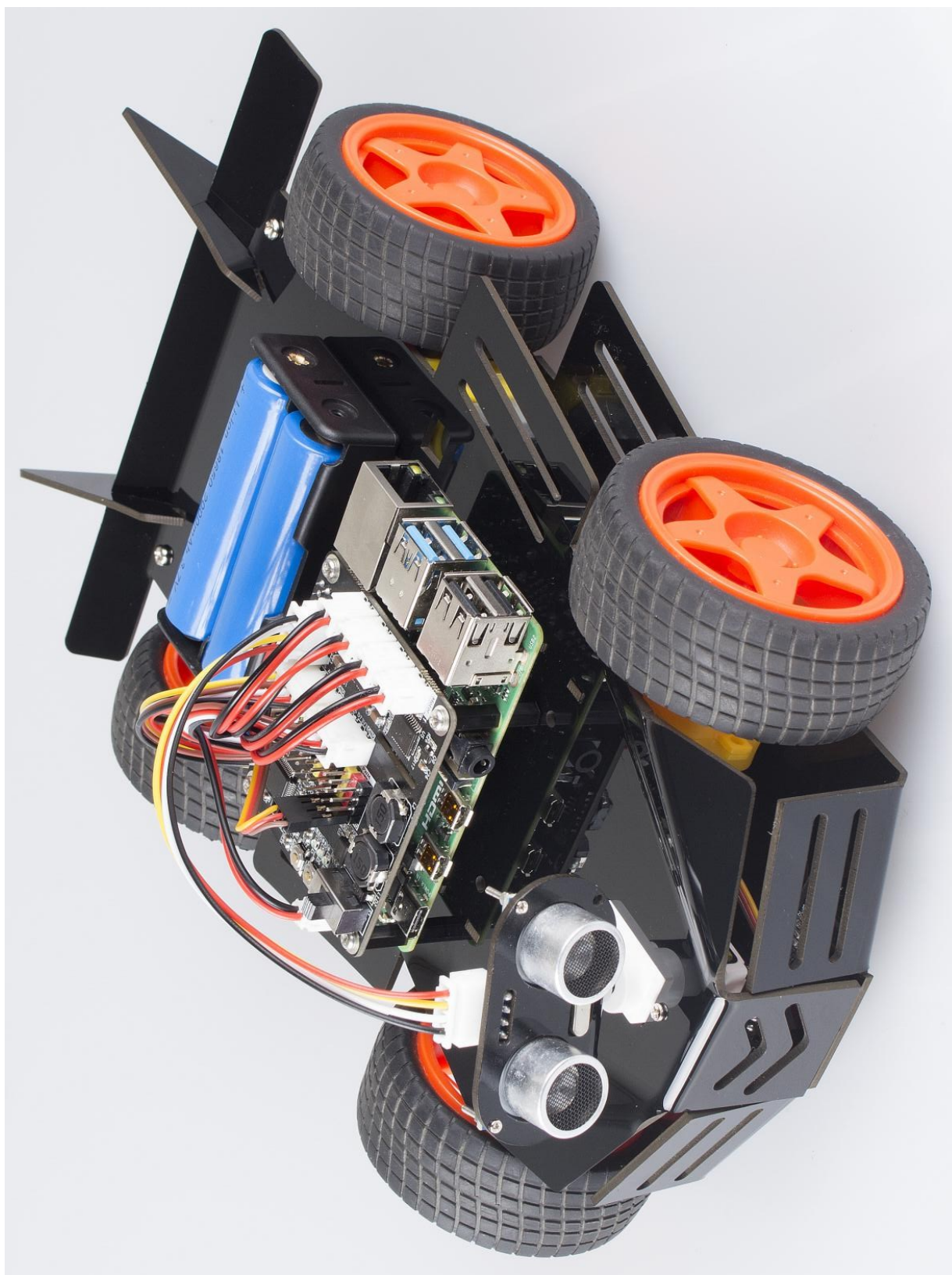
18650 Batteries

2 PCS



Note: After opening the package, please check whether the quantity of components is compliance with product description and whether all components are in good condition.

PŘÍLOHA P5 : ZVĚTŠENÝ OBRÁZEK 14



PŘÍLOHA P6: UKÁZKA OSTATNÍCH KATEGORIÍ TOOLBOXU

