VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

SYSTÉM PRO AUTONOMNÍ ŘÍZENÍ MODELU AUTÍČKA NA ZÁVODNÍ DRÁZE

SYSTEM FOR AUTONOMOUS NAVIGATION OF TOY CAR ON A RACE TRACK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JAROSLAV KATRUŠÁK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VÁCLAV ŠIMEK**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2021/2022

Zadání diplomové práce

Student:	**Katrušák Jaroslav, Bc.**
Program:	Informační technologie a umělá inteligence
Specializace:	Vývoj aplikací
Název:	**Systém pro autonomní řízení modelu autíčka na závodní dráze** **System for Autonomous Navigation of Toy Car on a Race Track**
Kategorie:	Vestavěné systémy

Zadání:

1. Nastudujte typické způsoby vytyčení závodní dráhy pro samostatný pohyb vozidla (modelářského autíčka) bez zásahu uživatele.
2. Seznamte se s principy detekce závodní dráhy dle bodu 1) zadání. Připravte krátkou studii, v níž přehledně shrnete získané poznatky.
3. Navrhněte koncepci systému autonomního řízení, který bude schopen regulovat rychlost pohybu vozidla s ohledem na rozpoznání klíčových úseků závodní dráhy (např. rovinku/zatáčku/křížení cest).
4. Zvolte vhodnou implementační platformu. Na obvodové úrovni k ní navrhněte schéma zapojení podpůrného systému pro detekci překážek v dráze modelářského autíčka. Tento detekční systém realizujte v podobě desky plošných spojů a proveďte jeho oživení.
5. Implementujte obslužný firmware, který s využitím zvolených detekčních mechanismů zajistí autonomní navigaci vozidla po vytyčené závodní dráze.
6. Ověřte funkčnost vytvořeného řešení v reálných podmínkách.
7. Zhodnoťte dosažené výsledky a navrhněte případná rozšíření systému autonomního řízení.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Šimek Václav, Ing.**
Vedoucí ústavu:	Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání:	1. listopadu 2021
Datum odevzdání:	18. května 2022
Datum schválení:	10. května 2022

Abstrakt

Cílem této práce je realizace systému pro autonomní řízení modelu autíčka na závodní dráze. První částí práce je studie typických způsobů vytyčení závodní dráhy pro pohyb modelářského autíčka a na ni navazující studie možných způsobů realizace autonomního pohybu modelu po závodní dráze. Další část je věnována hardwarové a softwarové stránce realizace navrženého systému. Předposlední část představuje navržený systém pro detekci překážek, jeho realizaci a výsledné vlastnosti. Poslední část práce prezentuje výslednou realizaci autonomního modelu schopného samostatné jízdy po vytyčené závodní dráze, čtení značek a reakce na překážku umístěnou na trati.

Abstract

The goal of this project is the implementation of a system for autonomous control of a toy car model on a racetrack. First part of this project is a study of typical ways of laying out a racetrack for the movement of a model car and a follow-up study of possible ways of implementing the autonomous movement of the model on the raceteack. The next part is devoted to the hardware and software side of the implementation of the proposed system. The penultimate part presents the proposed system for obstacle detection, it's implementation and resulting properties. The last part of the project presents the final implementation of an autonomous model capable of independent driving on a set racetrack, reading signs and reaction to obstacle located on the track.

Klíčová slova

Autonomní, autíčko, senzor, řídící jednotka, NXP Cup, závodní dráha, obrazový snímač, zpracování obrazu, Python, LIDAR

Keywords

Autonomous, toy car, sensor, control unit, NXP Cup, racetrack, image sensor, image processing, Python, LIDAR

Citace

KATRUŠÁK, Jaroslav. *Systém pro autonomní řízení modelu autíčka na závodní dráze.* Brno, 2022. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Václav Šimek

Systém pro autonomní řízení modelu autíčka na závodní dráze

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením pana Ing. Václava Šimka. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jaroslav Katrušák
16. května 2022

Poděkování

Rád bych poděkoval panu Ing. Václavu Šimkovi za veškeré rady, připomínky, nápady a konzultace při psaní této práce a za zapůjčení mnoha součástek potřebných pro vývoj. Dále bych chtěl poděkovat za podporu a pomoc členům svojí rodiny, konkrétně mámě za to, že zvládala žít s občasně nepřístupnou kuchyní, ve které probíhalo testování autíčka. Mému tátovi za pomoc s výrobou závodní dráhy a mojí sestře za to, že byla mojí debugovací kačenkou. Poděkování patří také Anně Šiškové rovněž za roli debugovací kačenky a všem zmíněným společně s mými prarodiči také náleží velké poděkování za mnohaletou podporu v průběhu celého studia.

Obsah

Kapitola 1

Úvod

Snahy o vývoj autonomních vozidel a dopravních prostředků jsou těsně spjaty s vývojem a inovacemi v oblasti výpočetní techniky. Přestože je možné některé pokusy, o jistou úroveň autonomního řízení, datovat již do první poloviny 20. století, tak většího rozvoje se tato oblast dočkala až s příchodem výpočetní techniky ve druhé polovině 20. století. Počáteční nadšení v této oblasti bylo jistým způsobem utlumeno právě nedostatky tehdejší výpočetní techniky a senzorového vybavení, které je, společně s komplexností prostředí, ve kterém se pozemní vozidla pohybují, hlavním limitem pro autonomní řízení.

Z tohoto důvodu bylo hlavních pokroků dosaženo až v průběhu 21. století. Přestože skutečně autonomní řízení dopravních prostředků, ve smyslu naprosté nepřítomnosti lidské kontroly, nebylo dosud plošně nasazeno, tak v mnoha oblastech lidského života jsou autonomní systémy jeho běžnou součástí. Zřejmě nejvýrazněji patrné systémy autonomního řízení lze nalézt v oblasti letectví, kde tyto systémy každým dnem zvyšují bezpečnost provozu, umožňují autonomní let včetně přistání nebo přesné lety v malé výšce nad terénem. Jisté autonomní systémy jsou dnes již nedílnou součástí některých automobilů přičemž dochází k jejich neustálému zlepšování a snižování potřeby lidské kontroly. Doposud dosažená úroveň autonomního řízení a rychlost jeho vývoje v různých oblastech použití koresponduje s komplexností prostředí, v němž se má daný autonomní prostředek pohybovat.

Cílem této práce je studie možných způsobů realizace autonomního pohybu modelu robotického vozítka po závodní dráze. Na základě získaných poznatků navrhnout a implementovat systém pro autonomní pohyb robotického vozítka po závodní dráze a následně navrhnout systém pro detekci překážek umístěných na závodní dráze a o tento systém rozšířit již implementovaný systém autonomního pohybu. Práce vychází ze soutěže NXP cup, ovšem v průběhu samotné práce bylo rozhodnuto nevázat se pouze na regule této soutěže, zejména co se použitého hardwaru týče, s cílem prozkoumat širší spektrum možností. Ze zmíněné soutěže však jsou, pro tuto práci a samotnou realizaci, využity specifikace závodní dráhy a základní platforma robotického autíčka.

Tato práce je dělena do 8 kapitol, včetně úvodu a závěru. Kapitola 2 seznamuje čtenáře se způsoby vytyčení závodní dráhy pro samostatný pohyb vozidla, 3. kapitola se zabývá principy detekce vytyčené závodní dráhy. V kapitole 4 je rozebrána navržená koncepce systému autonomního řízení. Kapitola 5 je věnována hardwarové a softwarové stránce realizace navrženého systému a poznatkům objevených při vývoji tohoto systému. Předposlední kapitola s číslem 6 čtenáře seznámí s navrženým systémem pro detekci překážek, realizací tohoto systému a jeho výslednými vlastnostmi. Poslední 7. kapitola představuje výslednou podobu autonomního robotického vozítka, konstrukci testovací dráhy a rozebírá dosažené výsledky při jízdě po několika typech závodní dráhy a v různých soutěžních disciplínách.

Kapitola 2

Jízdní dráha

Jízdní dráha je nezbytnou součástí této práce. Následující sekce jsou věnovány některým ze způsobům vytyčení jízdní dráhy. Na začátku je v krátkosti zmíněna jejich podobnost s běžnými pozemními komunikacemi a představeny potřebné vizuální vlastnosti prvků, vytyčujících jízdní dráhu, z pohledu její detekce obrazovými snímači. Jsou představeny některé z pořádaných soutěží autonomních robotických vozítek včetně soutěže NXP cup[1], jejíž způsob vytyčení testovací dráhy byl, přestože z důvodu použitého hardwaru se navrhované robotické vozítko nemůže oficiální soutěže účastnit, převzat pro účely této práce. Mimo samotné vytyčování jízdní dráhy v jednotlivých soutěžích je věnována pozornost i rozdílným pravidlům a povolenému hardwaru.

2.1 Způsoby vytyčení dráhy

Proto, aby se mohlo autonomní vozítko smysluplně pohybovat, potřebuje příslušnou dráhu. Na běžných pozemních komunikacích je tato dráha vytyčena zejména samotnou existencí povrchu určeného pro pohyb vozidel, který sám o sobě poskytuje jisté vytyčení dráhy. Kromě toho je tato dráha dále vytyčena vodorovným (obrázek 2.1) a svislým dopravním značením. Jelikož jsou robotické modely autíček vcelku výhodnou vývojovou platformou i pro některé budoucí systémy autonomních vozidel určených pro pozemní komunikace, zejména díky jejich nižší ceně dané jejich menším měřítkem a mnohem nižšímu nebezpečí v případě nějaké nehody, tak si dráhy, určené pro autonomní pohyb robotických autíček, často propůjčují některé z vodících prvků používaných na běžných pozemních komunikacích. Typicky se jedná o vodorovné dopravní značení, zejména krajní, případně středové vodící čáry (obrázek 2.2), které umožňují velmi jednoduše vytyčit dráhu ve vnitřním prostředí, ve kterém se často nachází jednolitý povrch.

Pro účely detekce jízdní dráhy prostřednictvím obrazových snímačů je potřebné aby vodící čáry, středové či okrajové, byly dostatečně odlišitelné od standardního povrchu. To lze zajistit dostatečným kontrastním poměrem mezi čarou a okolním povrchem. Obdobně jako na silničních komunikacích je zde tedy možné nalézt bílé, nebo žluté vodící čáry na tmavém podkladu, případně jejich inverzovanou variantu (například bílá dráha s černými vodícími čarami jako na obrázku 2.7), které jsou dostatečně odlišitelné od okolního povrchu.

[1] https://nxpcup.nxp.com/

Obrázek 2.1: Vytyčení jízdních pruhů na pozemní komunikaci (zdroj [27]).

2.2 Pořádané soutěže

Tato sekce se zaměřuje na soutěže autonomních robotických vozítek ve kterých se používá pro vytyčení jízdní dráhy vizuálních prvků nastíněných v sekci 2.1. Níže uvedené soutěže se, přes mnohé své podobnosti, liší jak v pojetí samotné soutěže, tak i ve způsobu vizuálního značení vytyčujícího samotnou závodní dráhu. Konkrétně jsou zde představeny 3 soutěže, kterými jsou NVIDIA's DIY Autonomous Car Race[2], AWS DeepRacer[3] a Formula Pi[4].

NVIDIA's DIY Autonomous Car Race

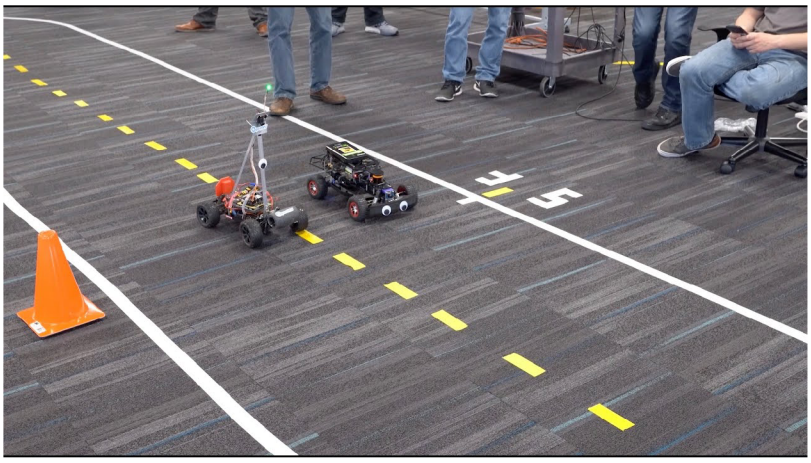
Jedná se o soutěž [19] pod záštitou společnosti nVidia[5], přičemž pravidla týkající se použitého hardwaru jsou nastavená relativně volně (při porovnání s některými dalšími soutěžemi). Je vyžadováno použití některého modulu z řady Jetson[6], ovšem co se týče dalšího hardwaru jako například obrazových snímačů, senzorů vzdálenosti, akcelerometrů či dalších senzorů je zde v podstatě ponechána volná ruka vývojářům. Z tohoto důvodu je soutěž rozdělena na dvě kategorie. Základní kategorie je určená pro vozítka s rozvorem kol maximálně 190 mm a nižší pořizovací cenou, zatímco nelimitovaná kategorie, kde vozítka mohou dosahovat znatelně větších rozměrů, nemá stanovenou horní hranici celkové ceny hardwaru.

Co se týče závodní dráhy, tak tato je celkem značně inspirována silničním vodorovným dopravním značením (obrázek 2.1). Dráha je vyznačena dvěma okrajovými plnými bílými čarami a ve středu dráhy se nachází přerušovaná žlutá čára (obrázek 2.2). Pro vyznačení dráhy jsou použity barevné lepící pásky, přičemž minimální šířka dráhy musí být v kterémkoliv bodě tratě 1 m (obrázek 2.3). Zajímavým aspektem této soutěže je využití oranžových dopravních kuželů. V soutěži může vozítko vyjet z dráhy dané bílými vodícími čarami, avšak pokud tak učiní v zatáčce, tak nesmí objet kužel umístěný na vnitřní straně zatáčky vnitřkem okruhu, vedlo by to k okamžité diskvalifikaci, jelikož kužely slouží ke stanovení minimální dráhy, kterou musí vozítko projet. Případné dotknutí se kuželu, při kterém nedojde k jeho objetí vnitřkem okruhu, je časově penalizováno.

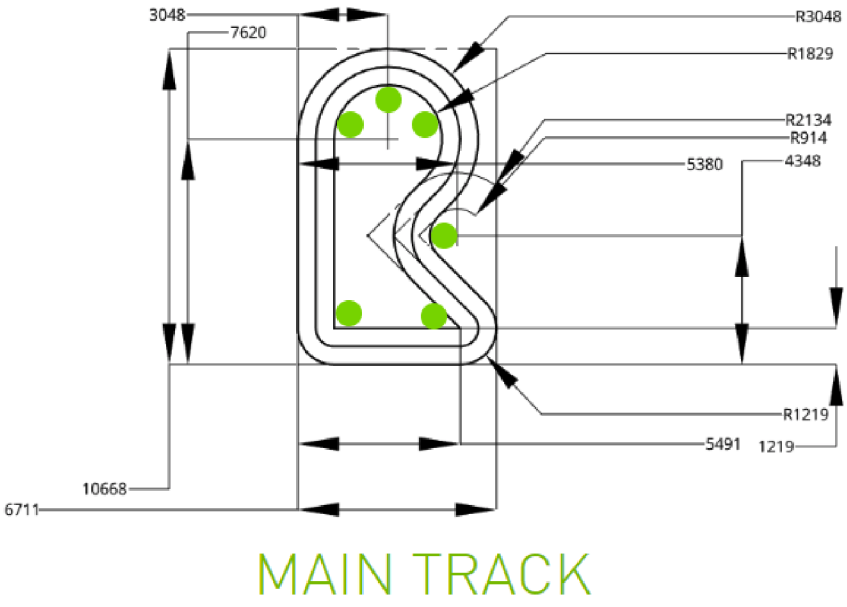
Při samotném závodu se v běžném kole na trati nacházejí dvě vozítka, která startují současně. První kolo se odehrává v náhodném pořadí, v následujících kolech pak vedle sebe stojí vozidla s co nejpodobnějšími časy. Pro určení toho kdo postupuje do semifinále se však počítá pouze první kolo. Semifinále se účastní šest nejrychlejších vozítek, která

[2] https://developer.nvidia.com/embedded/diy-ai-race
[3] https://aws.amazon.com/deepracer/
[4] https://www.formulapi.com/
[5] https://www.nvidia.com/cs-cz/
[6] https://www.nvidia.com/cs-cz/autonomous-machines/jetson-store/

jedou současně, zatímco dvě nejrychlejší z nich se budou účastnit finálního závodu. Ve všech závodech se také na trati nachází překážka, kterou musejí vozítka detekovat a objet, přičemž po každém kole je umístění této překážky náhodně změněno.



Obrázek 2.2: Vytyčení jízdní dráhy v soutěži NVIDIA's DIY Autonomous Car Race (zdroj [34]).

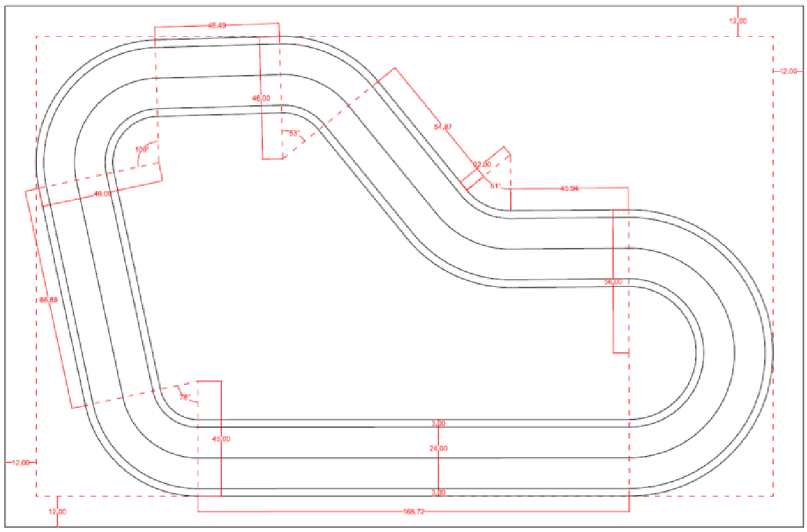


Obrázek 2.3: Specifikace závodní dráhy (rozměry v milimetrech) pro soutěž NVIDIA's DIY Autonomous Car Race - zelené objekty představují umístění kuželů podél tratě (převzato z [19]).

AWS DeepRacer

Soutěž AWS DeepRacer [33], pořádaná společností Amazon[7] má přesně stanovenou specifikaci hardwaru. Pro soutěž je nutné mít model vozítka navržený a vyrobený společností Amazon. Všichni účastníci tedy mají stejný hardware a ovlivňují pouze softwarovou stránku vozítka. Soutěž se přímo zaměřuje na využití strojového učení a pro trénování vlastního modelu jsou za poplatek poskytovány softwarové nástroje. Trénování a testování modelu je možné realizovat ve virtuálním prostředí simulátoru a až poté testovat navržený model na fyzickém vozítku a trati. Samotné závody mohou mít jak plně virtuální podobu, konanou v prostředí simulátoru, tak i fyzickou, která se odehrává fyzické závodní dráze.

Obrázek 2.4: Vzhled fyzické závodní dráhy soutěže AWS DeepRacer (převzato z [2]).



Obrázek 2.5: Specifikace fyzické závodní dráhy (rozměry v palcích) pro soutěž AWS DeepRacer z roku 2018 (převzato z [1]).

Fyzická závodní trať (obrázek 2.4) je vizuálně podobná trati ze soutěže NVIDIA's DIY Autonomous Car Race (viz 2.2). Specifikace dráhy [6] přímo hovoří o černém podkladu

[7] https://aws.amazon.com/

trati, který imituje povrch podobný skutečným silničním komunikacím, s bílými okrajovými čarami vytyčujícími samotnou dráhu a oranžovou středovou přerušovanou čarou, přičemž povrch dráhy i okrajové čáry by měly být z materiálu, který hůře odráží dopadající světlo. Dráha by s tolerancí přibližně 7,5 cm (oficiální rozměry jsou udávány v palcích) měla být široká alespoň 60 cm (obrázek 2.5).

Formula Pi

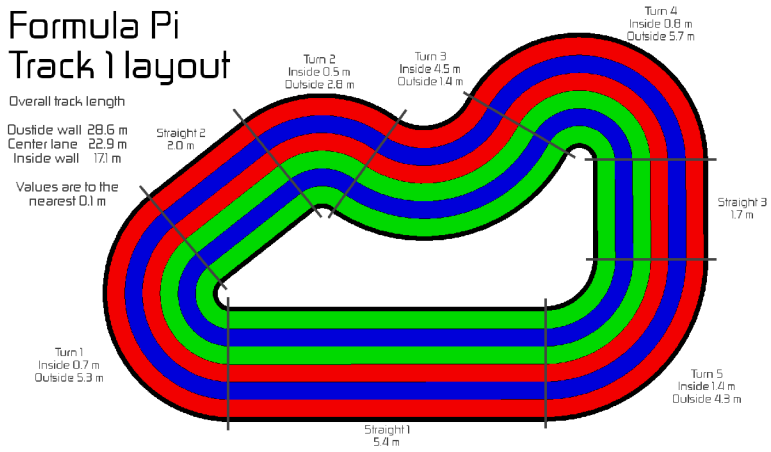
Formula Pi [11] je soutěž založená na používání vývojové desky Raspberry Pi[8]. Obdobně jako v případě soutěže AWS DeepRacer (viz 2.2) zde soutěžící týmy mají k dispozici stejný hardware. Pro ladění a tvorbu vlastního řídícího programu je k dispozici simulátor, ve kterém probíhá veškerý vývoj. Vyvinutý kód poté tým zašle organizátorům do soutěže, která se koná vzdáleně, kde organizátoři pro každý tým zajistí vozítko s kompletní kontrolou a servisem včetně nahrání softwaru.

Protože týmy nemají přístup k samotnému vozítku, tak jsou poskytovány dobrovolná testovací kola, ve kterých vozítko daného týmu projede trať. Je možné zajet více kol, ale čas na trati je limitován na 2 minuty, poté je testovací jízda ukončena a výsledky, včetně logovacích souborů, jsou odeslány zpět danému týmu, který tak má možnost následně provést úpravy svého kódu na základě získané telemetrie. Samotné bodované závody se v první fázi konají mezi 2 soutěžícími vozítky, kdy je cílem ujet co nejvíce kol, při vyhýbání se druhému vozítku, za 3 minuty. V závodní fázi závodí na trati současně 5 vozítek po dobu 10 minut a ve finále závodí 10 dosud nejlepších vozítek ve dvou 5 členných skupinkách. Veškerá kola, ať již testovací nebo hodnocená, jsou živě vysílána a každý tým tak má možnost sledovat jak se jejich vozidlu, v průběhu soutěže, daří.

Nejzajímavější částí soutěže je však, oproti doposud představeným soutěžím, podoba závodní dráhy. Její povrch je tvořen celkem šesti barevnými pruhy, kdy roboti jedoucí po dráze ve směru hodinových ručiček vidí na levé polovině dráhy červený, modrý a červený pruh, zatímco na pravé polovině dráhy vidí zelený, modrý a zelený pruh. Dráha je po celé své délce ohraničena mantinely, tudíž žádné vozítko nemůže v průběhu závodu vyjet z dráhy, ovšem pokud narazí do překážky a přestane se následně pohybovat, nebo dokonce vyrazí v opačném směru vytvoří tak překážku pro ostatní vozítka, které se jí musejí vyhýbat.

Oproti ostatním, dosud představeným soutěžím, se zde tedy vozítka nesetkávají pouze se staticky umístěnou překážkou, ale musejí se vyhýbat i pohybujícím se překážkám, jež jsou tvořeny ostatními vozítky, které se dokonce mohou pohybovat v protisměru. Srážky nejsou přímo penalizovány, ovšem srážka s jiným vozíkem může způsobit, že vozítko již závod nedokončí a nebo se zhorší jeho umístění, a v nejhorším případě jízdy v protisměru se každé takto ujeté kolo odečítá od celkového počtu ujetých kol. Jelikož zde soutěží velké množství vozítek současně, tak dráha dosahuje, ve srovnání s jinými soutěžemi, velké šířky a to přibližně 1,83 m, přičemž na každý jeden barevný pruh připadá circa 305 mm (obrázek 2.6).

[8] https://www.raspberrypi.org/



Obrázek 2.6: Specifikace fyzické závodní dráhy soutěže Formula Pi (převzato z [36]).

2.3 Soutěž NXP Cup

Jedná se o celosvětovou soutěž autonomních robotických modelů autíček, jejíž historie se datuje až do roku 2003, kdy se odehrála první soutěž v Jižní Korei. V následujících letech došlo k rozšíření této soutěže do zbytku Asie a následně do celého světa. Soutěž je pořádána pod záštitou společnosti NXP[9]. Text této sekce čerpá informace z pravidel soutěže a specifikace závodní dráhy [20, 23].

Pravidla soutěže - hardware

Obdobně jako v případě dalších soutěží, uvedených v této práci, dává záštita dané společnosti tušit jakým hardwarem mohou být modely, v rámci dané soutěže, vybaveny. V soutěži se využívá podvozek DFRobot Car Kit[10] a pro ročník 2022 byla povolena nová platforma v podobě podvozku WLToys 12409 chassis. Podvozek může být v provedení s kartáčovými nebo bezkartáčovými elektromotory, kdy pro kartáčové elektromotory je doporučována deska MikroE[11], zatímco pro bezkartáčové provedení elektromotorů je doporučována řídící jednotka RDDRONE-FMUK66[12] (na obrázku 3.1). Na bázi jednotky RDDRONE-FMUK66 je možné stavět jak autonomní robotická vozítka, tak i nejrůznější drony. V současné podobě soutěže jsou tyto desky pouze doporučovány, na rozdíl od minulých ročníků kdy bylo použití konkrétní řídící jednotky povinné, a je možné využít i jiných desek, případně mikrokontrolerů nebo mikroprocesorů, ovšem musejí být od společnosti NXP. Obdobná podmínka platí i pro obrazové snímače, kdy je sice možné použít i kameru od jiného výrobce, ovšem ta posléze musí být připojena přímo k hlavní řídící desce, na níž bude probíhat zpracování obrazu. V případě kamer od NXP je možné je připojit i k vedlejšímu mikrokontroleru (rovněž od NXP) pro případné předzpracování obrazových dat. Doporučovaným obrazovým snímačem je kamera Pixy2[13]. Mimo doporučený hardware je

[9] https://www.nxp.com/

[10] https://cz.mouser.com/new/dfrobot/dfrobot-rob0170-racing-car/

[11] https://www.nxp.com/design/development-boards/freedom-development-boards/mcu-boards/mikroe-nxp-cup-mainboard:RDDRONE-CUPK64

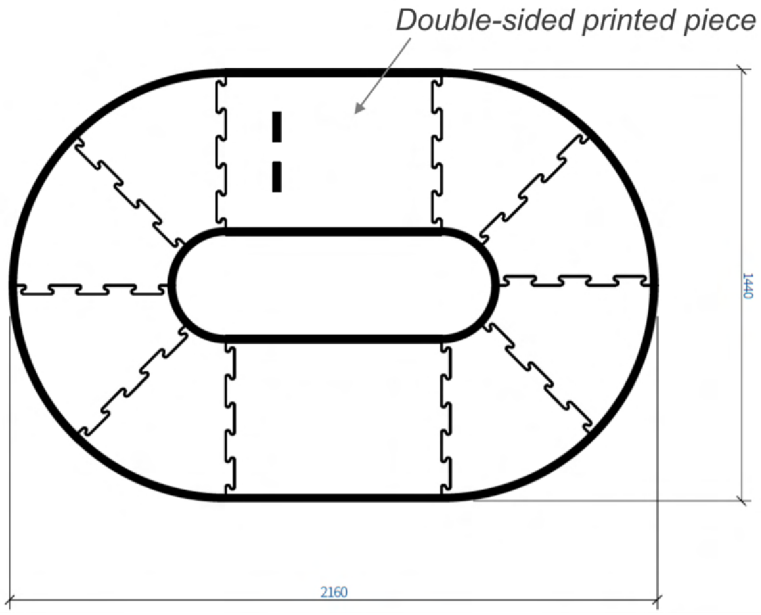
[12] https://www.nxp.com/design/designs/px4-robotic-drone-vehicle-flight-management-unit-vmu-fmu-rddrone-fmuk66:RDDRONE-FMUK66

[13] https://pixycam.com/pixy2/

možné přidávat i další senzory, které mohou například sloužit pro detekci překážek, což je jedna z disciplín v této soutěži.

Závodní dráha

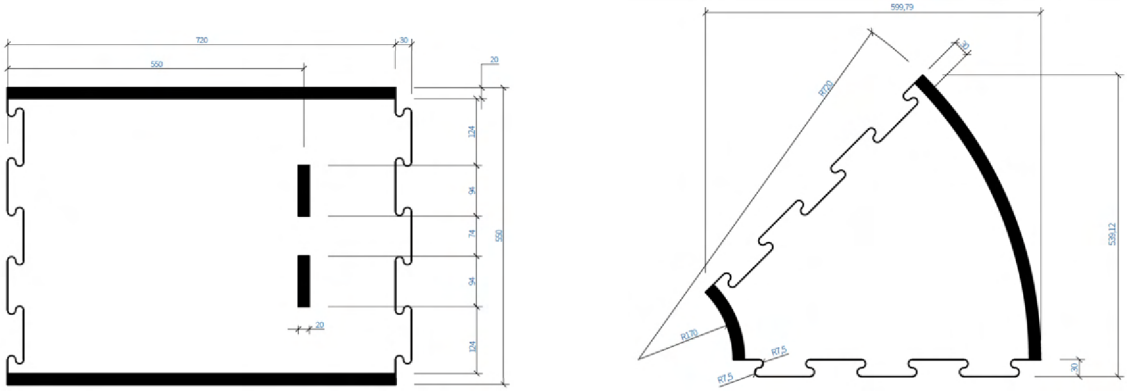
Obdobně jako povolený hardware tak i závodní dráha v průběhu let prošla mnohými změnami. V současné době má dráha podobu skládačky (obrázek 2.7), kdy jednotlivé díly tvořící závodní dráhu jsou libovolně vzájemně propojitelné s cílem umožnit vytvoření naprosto libovolného okruhu včetně křížení. Podoba jednotlivých dílů je přesně specifikována pravidly soutěže (obrázek 2.8). Závodní dráha má oproti jiným, zde uvedeným soutěžím, invertované barvy a je tvořena bílým podkladem s černými okrajovými čárami. Šířka dráhy je 550 mm, přičemž černá okrajová čára na každé straně je široká 20 mm. Kromě dílů s hraničními čarami v podobě rovinek a zatáček jsou součástí tratě ještě díly, na kterých se nacházejí další značky, které představují start a cíl, nebo které znamenají, že vozítko má zpomalit na poloviční rychlost, případně opět zrychlit na původní rychlost. Ke zpomalení slouží rovný díl, na kterém se kromě okrajových čar nacházejí i 4 rovnoběžné pruhy, zatímco v případě 3 pruhů má vozítko naopak zrychlit na svou standartní rychlost. Posledním prvkem dráhy je polystyrenová krychle o rozměru $20 \times 20 \times 20$ cm, kterou musí vozítko detekovat a objet aniž by se jí dotklo, nebo vyjelo mimo hranice dráhy.



Obrázek 2.7: Ukázka závodní dráhy soutěže NXP Cup (převzato z [23]).

Průběh soutěže

Celá soutěž je složena z více bodovaných disciplín (povinných a volitelných), přičemž celkový počet získaných bodů za jednotlivé disciplíny určuje konečné pořadí účastníků. Oproti jiným, již představeným, soutěžím je na trati vždy přítomné maximálně jedno vozítko, tudíž na výsledek nemá vliv nečekaná srážka s jiným vozítkem. Hlavní disciplína je představována závodem na dráze předem neznámého tvaru, kde je úkolem ji projet v co nejkratším čase,



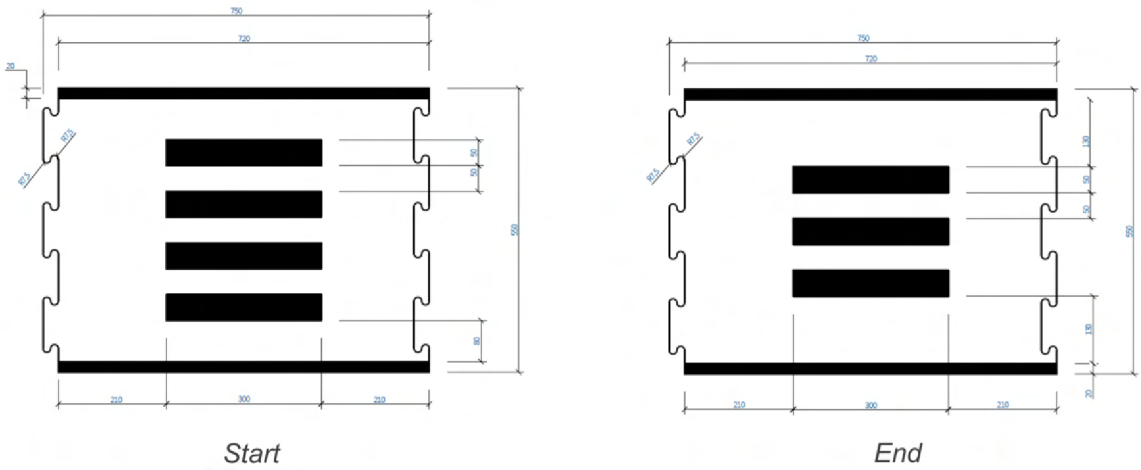
Obrázek 2.8: Ukázka specifikace dílů dráhy - vlevo specifikace startovní a cílové rovinky, vpravo specifikace dílu představujícího zatáčku (převzato z [23]).

přičemž na úspěšné zvládnutí této disciplíny jsou 3 pokusy kdy po každém neúspěšném pokusu má jeden člen týmu, ve 2 minutovém časovém okně, možnost provézt úpravy vozítka. Jednotlivé týmy jsou na základě svých časů seřazeny, týmu s nejrychlejším časem je uděleno 650 bodů a každý další tým, až do 10. místa, získává o 50 bodů méně než tým předchozí.

Další disciplínou (volitelnou) je trať ve tvaru osmičky, ve které má vozítko každého týmu za úkol ve vyhrazeném čase, typicky 60 sekund až několika minut, ujet co největší množství kol, přičemž žádná část vozítka nesmí vyjet z dráhy.

Detekce překážky je složena ze dvou disciplín - cílem první z nich je překážku detekovat a objet bez dotknutí se jí a vyjetí z dráhy, druhá disciplína spočívá v tom, že po ujetí několika kol rozhodčí umístí na náhodné místo překážku a úkolem vozítka je po jejím detekovaní zastavit.

Disciplína zaměřená na zrychlení a zpomalování se skládá z trati, na které se nachází díl se čtyřmi černými rovnoběžnými pruhy (obrázek 2.9), po jejichž detekci musí vozítko zpomalit na polovinu svojí běžné rychlosti a po detekování tří pruhů musí opět zrychlit na svou standardní rychlost. Tato disciplína, stejně jako obě disciplíny s detekcí překážky, je ohodnocena 150 bonusovými body.



Obrázek 2.9: Vlevo začátek pomalého úseku, vpravo konec pomalého úseku (převzato z [23]).

Kapitola 3

Principy detekce dráhy

Následující kapitola je založena na poznatcích uvedených v kapitole 2 pojednávající o způsobech vytyčení závodních drah pro autonomní robotická vozítka. Následující podkapitoly jsou věnovány shrnutí získaných poznatků na jejichž základě jsou stanoveny požadavky na detekci dráhy (podkapitola 3.1). Následně jsou v podkapitole 3.2 rozebrány principy použitelné pro samotnou detekci na základě těchto požadavků. Rovněž je věnována pozornost možným technickým řešením takové detekce a jejich vlastnostem v podkapitole 3.3.

3.1 Požadavky na detekci

V kapitole 2 byly uvedeny příklady několika soutěží autonomních vozítek. Mezi těmito soutěžemi je, přes jejich různé odlišnosti, možné najít mnohé společné rysy. V každé z uvedených soutěží je nějakým způsobem nutné detekovat samotnou závodní dráhu, která přestože může být vytyčena více způsoby, tak se typicky jedná o plochu ohraničenou krajními čarami, případně o plochu se středovou vodící čarou. Z této skutečnosti vyplývá, že k udržení vozítka na dráze je nutné zajistit detekci těchto čar, udávajících tvar samotné dráhy, a jejich rozlišení od povrchu dráhy určeného pro pohyb vozítka a také okolního povrchu.

Dalším, v mnoha soutěžích přítomným elementem, bylo detekování fyzické překážky (případně více překážek) umístěné na trati. Nejčastěji používaným typem překážek ve zmíněných soutěžích byla kostka umístěná na libovolné místo na trati. Ovšem například v soutěži Formula Pi (viz 2.2) nebo NVIDIA's DIY Autonomous Car Race (viz 2.2) bylo možné se setkat i s pohybující se překážkou tvořenou dalšími vozítky na trati, což oproti statické překážce přináší určité výzvy, například v podobě mnohem kratšího času na zareagování pokud vozítka jedou přímo proti sobě.

Jelikož, jak již bylo zmíněno v úvodu, je tato práce založena na soutěži NXP Cup (viz 2.3), ze které přejímá pravidla závodu a specifikace závodní dráhy společně se základní platformou vozítka, tak primárně se následující text bude zabývat následujícími dvěma oblastmi:

- detekce závodní dráhy včetně značek na dráze
- detekce statické překážky

3.2 Detekce závodní dráhy

Jelikož hlavní vlastností odlišující od sebe povrch dráhy a čáry je jejich barva, což je vizuální vlastnost, tak bude nutné zajistit získávání obrazu dráhy a to je možné prostřednictvím nějakého vhodného obrazového snímače. Vzhledem k tomu, že vozítko se má po dráze neustále pohybovat, bude nutné provádět snímání dráhy s dostatečnou frekvencí a následně získaná obrazová data rychle zpracovávat tak, aby řídící jednotka vozítka mohla provádět potřebné korekce směru a rychlosti jízdy.

K detekci čar se však dá použít i jiného přístupu a to využití více infračervených senzorů umístěných vedle sebe, kde jednotlivé senzory jsou umístěně na stejné přímce a tvoří na ní body. Toto je využitelné zejména pokud je dráha vytyčena například plnou černou čarou uprostřed bílé vozovky, kdy vozítko vlastně sleduje tuto čáru (starší ročníky soutěže NXP Cup). V případě, že se vozítko musí udržet mezi dvěma čarami, tak jako v případě většiny v této práci představených soutěží, by toto nebyl úplně elegantní způsob řešení. Bylo by nutné vytvořit dvě hlavní senzorová pole (případně jedno pokrývající celou šířku dráhy), každé umístěné na jedné straně vozítka tak, aby senzory mohly detekovat krajní vytyčující čáry a následně se podle toho, zda na jedné straně některý ze senzorů umístěný blíže k autíčku detekuje černou čáru, upraví trajektorie. Například když senzory na pravé straně blíže u vozítka detekují čáru, zatímco podle senzorů na levé straně je čára dále od vozítka, je nutné provézt korekci směru jízdy více doleva aby vozítko nevyjelo z dráhy. Navržené řešení v podobě dvou nepropojených senzorových polí by, oproti jednomu poli pokrývajícímu celou šířku dráhy, přineslo úsporu několika senzorů (v závislosti na šířce dráhy), Jelikož není nutné mít senzory mezi koly vozítka a sledovat po většinu času jen střed vozovky, kde se nenachází žádná čára. Senzory zde umístěné by byly využitelné pouze v situaci, kdy by vozítko zorným polem senzorového pole příliš vyjelo z dráhy. V takovém případě by pouze při detekci čar na stranách vozidla byl velmi obtížný návrat zpět na trať. Ve chvíli, kdy by se levá okrajová čára nacházela mimo levé senzorové pole a pravá čára se nacházela mimo pravé senzorové pole vozítka, tak by zde jednak došlo ke ztrátě informace o aktuální pozici vozítka mezi čarami, ovšem také by došlo ke ztrátě informace o tom, kterým směrem se vozítko pohybuje, pokud by senzorů bylo málo. Při dostatečném množství senzorů by bylo možné, vzhledem k tomu, jak by postupně danou čáru detekovaly vedle sebe umístěné senzory, odhadnout, kterým směrem vozítko vyjíždí z dráhy a provézt korekci na opačnou stranu a vrátit vozítko zpět na trať.

Ovšem toto řešení by vzhledem k tomu, že by IR senzory směřovaly přímo na dráhu a měly tak omezený dohled, přinášelo problém, jelikož by v určitých situacích, jako například prudkých, nebo opakovaných zatáček, vozítko bylo značně omezeno ve své maximální rychlosti, protože by nebyla známa povaha trati před vozítkem s dostatečnou rezervou a byly by nutné prudké změny směru, které by mohly způsobit vyjetí vozítka z trati. Jelikož cílem závodu je jet nejen co nejpřesněji, ale typicky i co nejrychleji do cíle, tak by možné rychlostní omezení, plynoucí z tohoto přístupu, nemuselo být zrovna úspěšné v konkurenci ostatních autíček, které by pro detekci čar používaly přístup, který by poskytoval větší přehled o trati před vozítkem.

Použití obrazového snímače (kamery), který snímá trať s dostatečným dohledem před vozítko, přináší jednoznačnou výhodu v tom, že na pořizovaných snímcích je vidět větší část dráhy před vozítkem. Toto posléze umožňuje v takových snímcích hledat okrajové čáry dostatečně dopředu a na základě jejich směrování průběžně upravovat rychlost a směr vozítka tak, aby bylo dosaženo co nejrychlejšího času kola, bez vyjetí z dráhy.

Technické řešení - NXP Cup

V případě soutěže NXP cup (viz 2.3) je doporučovanou řídící jednotkou deska RDDRONE-FMUK66 (obrázek 3.1), která je přímo určená pro nejrůznější ať již pozemní, nebo letecké autonomní prostředky. Pro tyto účely disponuje celou řadou zabudovaných senzorů a možností nastavení. Použití pouze této řídící desky ovšem může být, do jisté míry, limitujícím faktorem, prakticky určujícím celou podobu autonomního vozítka. V tomto případě se jedná zejména o optimalizaci celé řídící jednotky s cílem dosáhnout rozumné energetické efektivity vzhledem k poskytovanému výkonu, s cílem snížit spotřebu a umožnit zejména dronům co nejdelší pobyt ve vzduchu. Je tedy zřejmé, že výpočetní výkon, pro případné zpracování obrazu, není prioritou. Parametry jednotky RDDRONE-FMUK66 jsou uvedeny v tabulce 3.1.

Obrázek 3.1: Kit RDRONE-FMUK66 (uprostřed) s veškerým dodávaným příslušenstvím (převzato z [21]).

Společně s řídící deskou je doporučen i obrazový snímač v podobě kamery Pixy2[1], která je připravena pro naučení se detekce nejrůznějších předmětů (například míčků) a rovněž také disponuje zabudovanými algoritmy pro detekci a sledování čar, což kameře propůjčuje, společně se snímáním při 60 snímcích za sekundu, dobré předpoklady pro použití v soutěži NXP Cup. Parametry kamery Pixy2 je možné nalézt v tabulce 3.2.

Soutěž NXP Cup ve stávající podobě umožňuje použití i jiné kamery, ovšem v případě, že bude kamera od jiného výrobce než NXP je nutné kameru připojit přímo k řídící jednotce od NXP, jejíž výkon může být v takovém případě určujícím limitem. V případě použití kamerového modulu vybaveného procesorem od společnosti NXP je však umožněno použít

[1] https://pixycam.com/pixy2/

Hlavní FMU procesor	Mikrokontroler Kinetis K66 MK66FN2MOVLQ18 s jádrem Arm Cortex-M4 běžícím na frekvenci 180 MHz
Paměť FLASH	2 MB
Paměť SRAM	256 KB
Konektivita	Ethernet 2× USB
Akcelerometr/Gyroskop	BMI088/ICM42688
Akcelerometr/Magnetometr	FXOS8700CQ
Gyroskop	FXAS21002CQ
Magnetometr	BMM150
Barometr	ML3115A2
Barometr	BMP280
GPS Přijímač	u-blox Neo-M8N GPS/GLONASS
GPS Akcelerometr/Magnetometr	FXOS8700CQ
GPS Magnetometr	IST8310

Tabulka 3.1: Parametry RDRONE-FMU66 (čerpáno z [21]).

Procesor	NXP LPC4330 se 2 jádry na frekvenci 204 MHz
Paměť RAM	264 KB
Paměť FLASH	2 MB
Obrazový snímač	Aptina MT9M114 s rozlišením 1296 × 976
Úhel záběru	60° horizontálně a 40°vertikálně
Napájení	USB (5 V), nebo neregulovaný vstup (6 až 10 V)
Datový výstup	sériový UART, SPI, I2C, USB

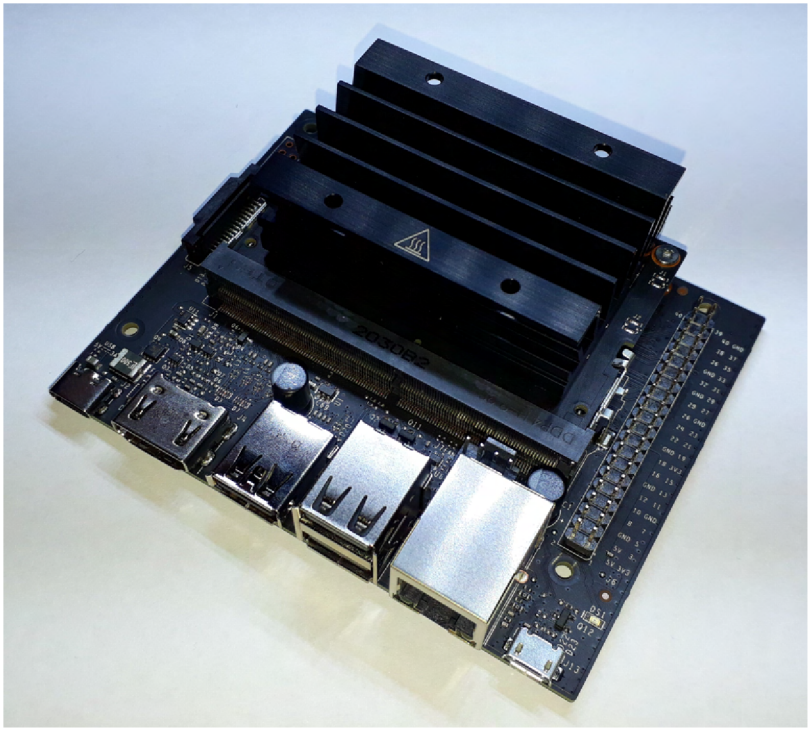
Tabulka 3.2: Parametry kamery Pixy2 (čerpáno z [22]).

pro analýzu obrazu nějaký pomocný procesor, musí však být rovněž od společnosti NXP. Tento přístup již poskytuje do jisté míry větší možnosti než základní platforma.

Technické řešení - NVIDIA's DIY Autonomous Car Race

Soutěž NVIDIA's DIY Autonomous Car Race (viz 2.2) vyžaduje použití některého modulu z řady Jetson - například Jetson Nano (obrázek 3.2). Jedná se o modul určený pro vývoj umělé inteligence a robotických zařízení, přičemž pro tyto oblasti nabízí, zejména ve srovnání s RDDRONE-FMUK66 (viz 3.2), velmi vysoký výkon poskytovaný kombinací procesoru a grafického procesoru (tabulka 3.3), který přímo umožňuje provádět analýzu obrazu z kamery [16].

Co se kamer týče, tak v soutěži není omezení na nějaký konkrétní typ a je možné použít jakýkoliv snímač, jenž je s Jetson Nano kompatibilní, přičemž seznam podporovaných snímačů je poměrně obsáhlý [15]. Obecně jsou mezi uživateli velmi oblíbené kamery založené na snímačích IMX219 a IMX477, jelikož jsou tyto snímače základem kamer Raspberry Pi



Obrázek 3.2: nVidia Jetson Nano (galerie autora).

Grafický procesor	128jádrový procesor NVIDIA Maxwell™
Procesor	Čtyřjádrový procesor ARM® A57, 1,43 GHz
Paměť RAM	2 GB, 64bitová, LPDDR4, 25,6 GB/s
Úložiště	microSD slot
Encoder videa	4Kp30 | 4× 1080p30 | 9× 720p30 (H.264/H.265)
Dekodér videa	4Kp60 | 2× 4Kp30 | 8× 1080p30 | 18× 720p30 (H.264/H.265)
Konektivita	Gigabitový Ethernet, bezdrátové připojení 802.11ac*
Kamera	1x konektor MIPI CSI-2
Monitor	HDMI
USB	1× USB 3.0 typu A, 2× USB 2.0 typu A, 1× USB 2.0 Micro-B
Ostatní	40 pinový konektor (GPIO, I2C, I2S, SPI, UART) 12 pinový konektor (napájení a související signály, UART) 4 pinový konektor ventilátoru*
Rozměry	100 × 80 × 29 mm

Tabulka 3.3: Parametry nVidia Jetson Nano.

Camera Module 2 a Raspberry Pi High Quality Camera určených pro Raspberry Pi[2], které se těší velké oblibě mezi uživateli pro svoje široké možnosti použití (obrázek 3.3).

Tyto kamery jsou, vzhledem ke svým parametrům (IMX219 viz 3.7 a IMX477 viz 3.4), vhodné pro použití v autonomním vozítku, jelikož disponují dostatečným rozlišením a snímkovací frekvencí, díky čemuž je možné získávat dobré informace o podobě závodní dráhy před vozítkem. Ve spolupráci s nVidia Jetson Nano, které disponuje vysokým výpočetním

[2]https://www.raspberrypi.org/

Obrázek 3.3: Na obrázku vlevo se nachází Raspberry Pi Camera V2 (převzato z [25]), zatímco na obrázku vpravo je kamera Raspberry Pi HQ Camera (převzato z [26]).

výkonem je dokonce možné použít například dvě kamery současně, jejichž obraz bude skládán do jednoho pro dosažení většího zorného pole. Případně teoreticky je možné používat druhou kameru pro pohled za vozítko za účelem detekce dalšího soupeřícího vozítka, kterému by, za pomoci detekce objektů ve snímaném obraze, bylo možné aktivně bránit v předjíždění.

Typ snímače	CMOS
Snímač	SONY IMX477
Fyzické rozlišení snímače	4056 × 3040 pixelů
Rozlišení videa na RPi	1080p30 | 720p60 | 640 × 480p60/90
Výstup	sériový CSI2 (4/2 linkový)
Rozměry snímače	6,287 mm × 4,712 mm

Tabulka 3.4: Parametry kamery Raspberry Pi HQ Camera (čerpáno z [14]).

Technické řešení - Formula Pi

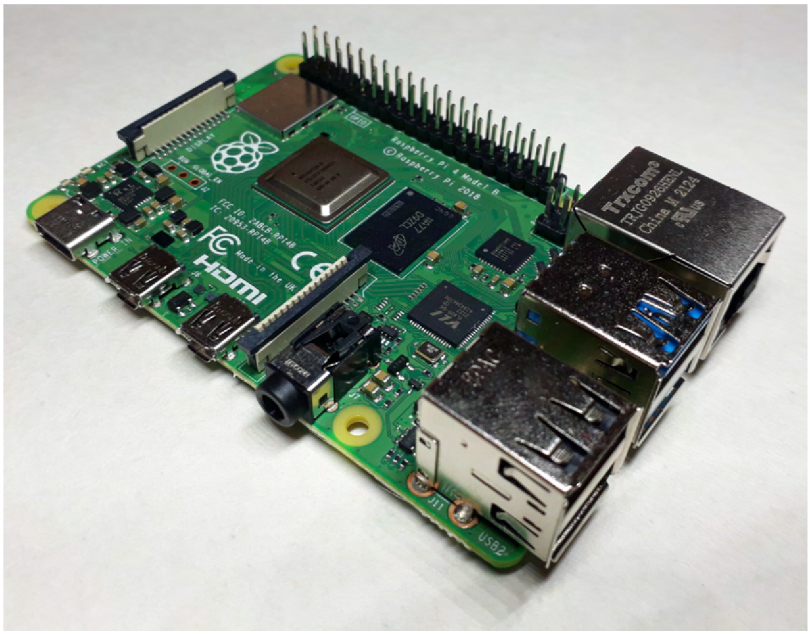
V soutěži Formula Pi 2.2 je konkrétní podoba hardwarového vybavení autíčka určena pravidly. Všichni účastníci mají vozítka v naprosto stejné konfiguraci a tedy jediným faktorem rozhodujícím o umístění autíčka je samotný řídící program. Používanou řídící deskou je Raspberry Pi 3B (tabulka 3.5), ovšem dá se předpokládat, že pokud se budou uskutečňovat další ročníky, tak dojde k přechodu na novější verzi v podobě Raspberry Pi 4B (tabulka 3.6, obrázek 3.4).

Procesor	Čtyřjádrový 64 bitový Broadcom BCM2837, 1,2 GHz
Paměť RAM	1 GB
Úložiště	microSD slot
Konektivita	100 Mbit Ethernet BCM43438 bezdrátová LAN Bluetooth Low Energy
Monitor	HDMI
USB	4× USB 2.0 typu A, 1× USB 2.0 Micro-B (napájení)
Ostatní	40-pinů GPIO rozhraní CSI, DSI
Rozměry	85 × 56 × 16 mm

Tabulka 3.5: Parametry Raspberry Pi 3 Model B (čerpáno z [24]).

Procesor	Čtyřjádrový 64 bitový ARM-Cortex A72, 1,5 GHz
Paměť RAM	1, 2, 4 nebo 8 GB LPDDR4
Úložiště	microSD slot
Dekodér videa	do 4Kp60 (H.265) | do 1080p60 (H.264)
Konektivita	1 Gbit Ethernet (s podporou PoE) B802.11 b/g/n/ac bezdrátová LAN Bluetooth 5.0 (BLE)
Monitor	2× micro-HDMI (až 2× 4k60p)
USB	2× USB 3.0 typu A 2× USB 2.0 typu A, 1× USB-C (napájení)
Ostatní	28-pinů GPIO rozhraní CSI, DSI
Rozměry	85 × 56 × 16 mm

Tabulka 3.6: Parametry Raspberry Pi 4 Model B (čerpáno z [7]).



Obrázek 3.4: Raspberry Pi 4B 8 GB (galerie autora).

Z uvedených parametrů modelů Pi 3B a Pi 4B, je zejména patrný výkonový nárůst co se síťové konektivity a zpracování videa týče. Pro použití v autonomním vozítku je ale nejpodstatnější změnou nárůst procesorového výkonu, který bude využitelný pro analýzu obrazu z kamery, ovšem v pravidlech soutěže Formula Pi je ke dni 16.1.2022 zatím stále uveden model 3B, který ve spolupráci s kamerou Raspberry Pi Camera V2 (obrázek 3.3) představuje dostatečné řešení pro navigaci autonomního vozítka. Tato kamera je založena na snímači SONY IMX219 [14], jehož parametry je možné nalézt v tabulce 3.7.

Typ snímače	CMOS
Snímač	SONY IMX219
Fyzické rozlišení snímače	3280 × 2464 pixelů
Rozlišení videa na RPi	1080p30 | 720p60 | 640×480p60/90
Výstup	sériový CSI2 (4/2 linkový)
Rozměry snímače	3,68 mm × 2,76 mm

Tabulka 3.7: Parametry Raspberry Pi Camera V2 (čerpáno z [30, 14]).

3.3 Detekce překážek

Na tuto oblast je možné se podívat více způsoby. Jednou z cest jak detekovat fyzickou překážku je využití již přítomného obrazového snímače, pokud byl zvolen pro detekci vodících čar. Jelikož detekci překážky na trati je možné s pomocí algoritmů strojového učení realizovat pouze na bázi kamerových snímků trati, kdy při vytrénování neuronové sítě na pravidly přesně definované překážce by bylo možné dosahovat velmi spolehlivé detekce překážky. Problémem by v takovém případě bylo získávání informace o vzdálenosti od takovéto překážky, což je technicky proveditelné, ovšem pro detekci vzdálenosti od překážky se jako vhodnější, než tuto vzdálenost počítat z kamerových snímků, jeví využití nějakého senzoru vzdálenosti - zejména z důvodu přesnosti a rychlosti měření. Co se senzorů vzdálenosti týče, tak zde se nabízí celkem široká řada možností, kterými se budou zabývat následující sekce.

Ultrazvukové senzory

Ultrazvukové senzory představují, v minulosti i v současnosti, na různých soutěžích, hojně používané řešení. Tyto senzory využívají odrazu vysílaného ultrazvukového signálu od překážky a na základě doby lety odraženého signálu je možné stanovit vzdálenost překážky. Při použití více senzorů, pro vytvoření senzorového pole pokrývajícího prostor před vozítkem, je možné dosáhnout přesného přehledu o tom, kde a v jaké vzdálenosti před vozítkem se překážka nachází a posléze určit, vzhledem k její pozici vůči okrajovým čarám, nejvhodnější způsob jejího objetí. Realizace a vlastnosti takto vytvořeného senzorového pole se odvíjejí od parametrů použitých senzorů, zejména závisí na jejich šířce záběru. V případě použití senzorů s širším úhlem záběru bude možné použít méně senzorů, ovšem nebude možné získat přesnou informaci o pozici překážky a spíše se bude jednat o informaci, že se před vozítkem nějaká překážka nachází, avšak nebude snadné na základě dostupných dat stanovit spolehlivý postup jejího objetí. Senzory s užším úhlem záběru umožňují vytvořit hustější senzorové pole a tím pádem získat přesnější informaci o tom, kde před vozítkem se překážka

nachází. Na úhel záběru je při tvorbě senzorového pole nutné brát zřetel nejen z důvodu přesnosti určení pozice překážky, ale i z toho důvodu, aby se jednotlivé senzory ve vytvořeném senzorovém poli nepřekrývaly. V takovém případě by mohlo docházet k situacím, kdy by vyslaný signál z jednoho senzoru byl po svém odražení přijat i sousedním senzorem a tím pádem by skutečná pozice překážky byla zkreslována. Příkladem senzoru použitelného pro autonomní vozítko je například senzor HC-SR04 (tabulka 3.8, obrázek 3.5), jehož úhel záběru je 12°.

Napájení	5 V DC (2 mA)
Úhel záběru	12°
Rozsah měřených vzdáleností	2 cm až 400 cm
Rozlišení	3 mm
Frekvence signálu	40 kHz
Rozměry	45 mm × 20 mm x 15 mm

Tabulka 3.8: Parametry ultrazvukového senzoru HC-SR04 (čerpáno z [10]).

Obrázek 3.5: Ultrazvukový senzor HC-SR04 (převzato z [32]).

ToF (*Time-of-Flight*) senzory

Jako alternativu ultrazvukových senzorů je možné použít ToF senzory, jejichž princip funkce je velmi podobný ultrazvukovým senzorům, pouze namísto vysílání ultrazvukových vln vysílají světelné paprsky, nejčastěji v podobě infračerveného světla. Jejich hlavní výhodou oproti ultrazvukovým senzorům je rychlost letu světla, která umožňuje rychleji a častěji získat informaci o vzdálenosti objektu, a menší rozměry. Nevýhodou těchto senzorů je jejich mnohdy vyšší pořizovací cena a z toho plynoucí riziko pro soutěže robotických autíček, kde hrozí jejich poškození v případě kolize. Kromě ceny také tyto senzory mají oproti ultrazvukovým užší zorný úhel a tedy je jich na pokrytí stejné oblasti potřeba více. Na druhou stranu se toto však může jevit i jako výhoda, jelikož použitím více senzorů s užším zorným polem je možné, díky rozdělení oblasti před autíčkem na menší sektory, dosáhnout přesnější informace o tom, kde před vozítkem se daná překážka nachází. Nedá se tedy obecně přímo říct, který přístup je lepší, poněvadž toto závisí na konkrétních požadavcích na detekci a tedy pro určitou soutěž mohou přinášet výhodu ultrazvukové senzory, zatímco v jiném případě může být výhodnější použít ToF senzory.

Příkladem takového senzoru je senzor TFMINI-S (na obrázeku 3.6), jehož parametry jsou uvedeny v tabulce 3.9. Oproti ultrazvukovému senzoru HC-SR04 (viz tabulka 3.8) má senzor TFMINI-S 3× větší dosah, avšak minimální detekční vzdálenost je 5× vyšší a to sice 10 cm. Rozlišení 10 mm je u tohoto senzoru nižší v porovnání s 3 mm rozlišením u ultrazvukového senzoru. Tyto rozdíly jsou dány zejména tím, že na vzdálenost v řádech desítek až stovek centimetrů je doba letu světelného paprsku natolik krátká, že chyba měření této doby je větší než v případě ultrazvukového senzoru a na vzdálenost menší než 10 cm je velmi obtížné změřit dobu letu. Proto je minimální dosah senzoru TFMINI-S větší než v případě senzoru HC-SR04. Na druhou stranu má tento senzor velmi úzký úhel záběru, 2°, oproti 12° u ultrazvukového senzoru, což představuje v podstatě bodové měření vzdálenosti. Toto může být jak výhoda, v případě použití mnoha senzorů pro vytvoření přesného senzorového pole, tak i nevýhoda v podobě nutnosti použití většího množství dražších senzorů na pokrytí přední polosféry vozítka, ve srovnání s ultrazvukovými senzory s širším záběrem a nižší jednotkovou cenou.

Napájení	5 V DC (140 mA)
Úhel záběru	2°
Rozsah měřených vzdáleností	10 cm až 12 m
Rozlišení	1 cm
Vlnová délka	850 nm
Frekvence snímání	1-1000 Hz UART | 1-100 Hz I2C
Výstup	UART | I2C
Rozměry	42 mm × 15 mm x 16 mm

Tabulka 3.9: Parametry senzoru TFMINI-S (čerpáno z [31]).

Obrázek 3.6: ToF senzor TFMINI-S. Obdobně jako v případě ultrazvukového senzoru, který má zvlášť přijímač a vysílač signálu, tak i zde je jednou čočkou světelný signál vysílán a druhou přijímán (galerie autora).

Mikrovlnné senzory

Obdobně jako již zmíněné ultrazvukové a ToF senzory i mikrovlnné senzory vysílají signál a prostřednictvím měření doby jeho letu k objektu a zpět určují vzdálenost detekovaného objektu. Příkladem takového senzoru je DFROBOT SEN0192 [18], který pracuje na principu Dopplerovského radaru a využívá tedy tzv. Dopplerův jev, s pomocí kterého detekuje pohybující se objekty. Obrázek 3.7 zobrazuje samotný senzor zatímco jeho detekční zóna je zobrazena na obrázku 3.8.

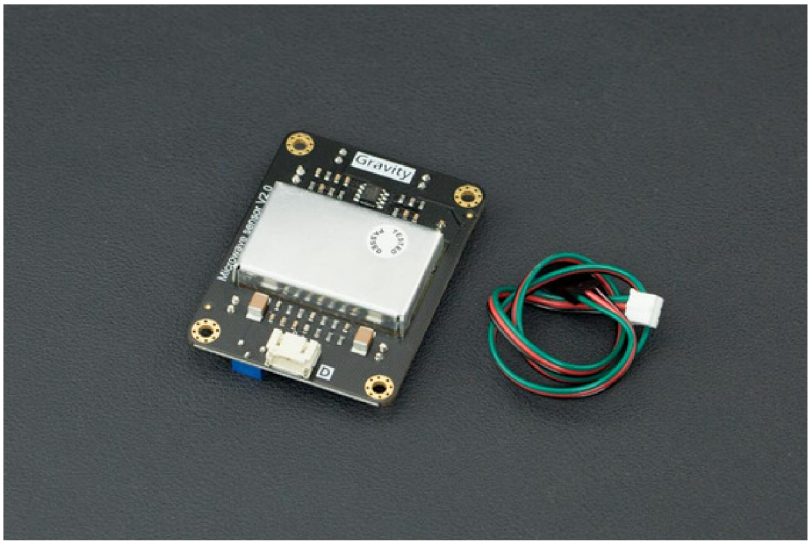
Dopplerův jev [35] značí změnu frekvence vysílané vlny na objektu představujícího překážku v závislosti na pohybu zdroje vysílaných vln. V případě, že se zdroj mikrovlnného vysílání (autíčko vybavené senzorem DFROBOT SEN0192) bude blížit k překážce, tak rozestup mezi jednotlivými mikrovlnnými vlnami bude tím kratší, čím vyšší bude rychlost autíčka, jelikož každá nově vyslaná vlna bude vyslána o kousek blíž cíle, než vlna předcházející. Pokud by se zdroj signálu vůči cíli nepohyboval, tak by rozestupy mezi jednotlivými vlnami odpovídali vysílací frekvenci. V případě, že by autíčko couvalo a vzdalovalo se tak od cíle, tak by naopak rozestupy mezi jednotlivými vyslanými vlnami byly větší úměrně k rychlosti jeho vzdalování se od cíle.

Při pohledu na parametry senzoru (tabulka 3.10), je patrné, že senzor DFROBOT disponuje v porovnání s senzorem TFMINI (tabulka 3.9) a ultrazvukovým senzorem HC-SR04 (tabulka 3.8) mnohonásobně vyšší minimální detekční vzdáleností a to 2 m oproti 10 cm, respektive 2 cm v případě senzoru HC-SR04. Takto velká detekční vzdálenost u tohoto konkrétního senzoru do značné míry limituje možnosti jeho použití pro autonomní robotická vozítka, jelikož dráha může mít takový tvar, že překážka nacházející se těsně za zatáčkou bude skryta mimo zorný úhel senzoru až do doby, kdy se překážka bude nacházet již příliš blízko na to, aby ji senzor detekoval. Tím pádem může dojít k tomu, že vozítko překážku jednoduše neuvidí a narazí do ní. Vzhledem k principu fungování tohoto senzoru je překážkou myšlena překážka pohybující se vůči statickému senzoru, případně senzor pohybující se na pohyblivé platformě vůči statické překážce, aby mohla být překážka prostřednictvím Dopplerova jevu detekována. Statická překážka nebude staticky umístěným senzorem detekována.

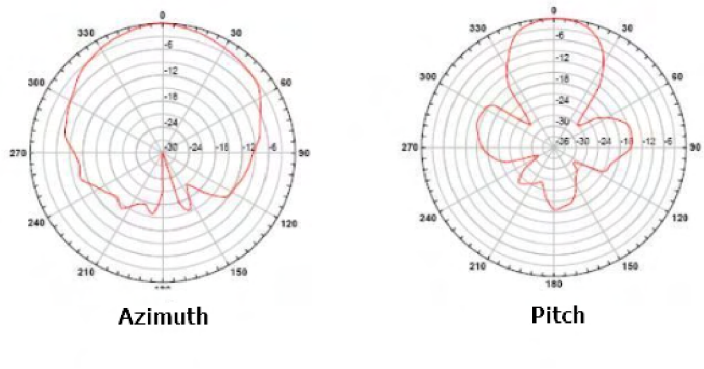
V porovnání s předchozími senzory je tedy tento senzor nejméně všestranný, jelikož se zaměřuje na pohybující se objekty, a jeho minimální detekční vzdálenost není příliš vhodná pro použití v autonomním vozítku minimálně na pozici hlavního senzoru pro detekci překážek. Využití tohoto senzoru je možné, avšak spíše jako doplňkového senzoru, kdy by pro primární detekci jak statických, tak i pohybujících se překážek byl využit jiný senzor s lepšími vlastnostmi a senzor DFROBOT by sloužil pouze jako doplňkový senzor s dlouhým dosahem pro detekci ostatních vozítek na závodní dráze v případě soutěží, ve kterých se na dráze pohybuje více autíček současně.

Napájení	5 V DC (max 60 mA)
Úhel záběru	2°
Rozsah měřených vzdáleností	2 m až 16 m
Rozlišení	-
Vysílací frekvence	10,525 GHz ()
Rozměry	48,5 mm × 63 mm

Tabulka 3.10: Parametry senzoru DFROBOT SEN0192 (čerpáno z [18]).



Obrázek 3.7: Senzor DFROBOT SEN0192 (převzato z [18]).



Obrázek 3.8: Detekční zóna senzoru DFROBOT SEN0192, vlevo se nachází horizontální oblast detekce, jejíž úhel záběru je 72° a vpravo je znázorněna vertikální detekční oblast, ve které je úhel záběru 36° (převzato z [18]).

LIDAR (*Light Detection And Ranging*)

LIDAR představuje se všemi dosud uvedenými možnostmi zřejmě nejpokročilejší řešení. Jedná se o zařízení určené pro detekci objektů v rozsahu až 360°. Princip fungování LIDARU je v základu stejný jako v případě ToF senzoru, je zde rovněž využíváno detekce odraženého světelného paprsku a na základě doby letu probíhá výpočet vzdálenosti objektu, od kterého se paprsek odrazil. Ovšem oproti statickému ToF senzoru LIDAR rotuje a tím pádem umožňuje opakované měření stále stejným senzorem, případně skupinou senzorů a tím získává mnohem více informací, za použití menšího množství senzorů, než v případě, že by 360° skenování okolí vozítka probíhalo za pomoci senzorového pole tvořeného statickými ToF senzory, jichž by bylo nutné použít velmi velké množství, řádově desítky vzhledem k úhlu jejich záběru, pro dosažení obdobného výsledku, jakého umožňuje dosáhnout LIDAR.

Oproti předchozím řešením také LIDAR, vzhledem k jeho principu fungování, umožňuje vytvářet mapu okolního prostředí a v případě, kdy skenování neprobíhá pouze v horizontální

rovině, ale senzor disponuje i vertikálním rozsahem, tak je možné vytvářet velmi podrobnou 3D mapu okolí jako na obrázku 3.9.

Obrázek 3.9: 3D mapa místnosti složená ze 4,7 milionů bodů získaných LIDARem (převzato z [39]).

LIDAR, vzhledem ke svým parametrům, představuje pochopitelně poměrně nákladné zařízení, ovšem cenově poměrně dostupné a zajímavé řešení představuje senzor RPLIDAR A1 [28]. Jedná se o 2D laserový scanner s dosahem až 6 metrů, který za jednu otáčku pořídí 360 vzorků při rozsahu 360° - tedy jeden vzorek na jeden stupeň, přičemž frekvence snímání je nastavitelná v rozmezí 1-10 Hz (vybrané parametry jsou obsaženy v tabulce 3.11).

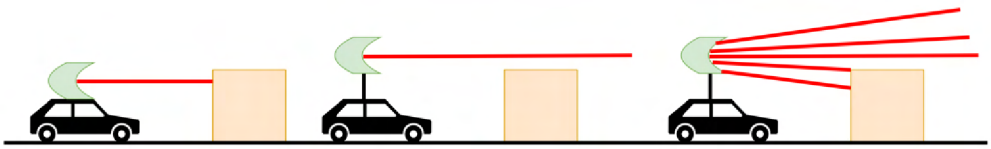
Napájení	5 V DC (max 600 mA)
Úhel záběru	360° (při kroku 1°)
Rozsah měřených vzdáleností	0,15 m až 6 m
Rozlišení	$<$ 0,5 mm do vzdálenosti 1,5 m $<$ 1 % jinak
Frekvence snímání	1-10 Hz (typicky 5,5 Hz)
Vlnová délka	775-795 nm (typicky 785 nm)
Rozměry	98,5 mm $\times$ 70 mm x 60 mm
Váha	190 g

Tabulka 3.11: Parametry senzoru RPLIDAR A1 (čerpáno z [28]).

Jelikož je RPLIDAR 2D senzorem, tak snímání vzdáleností probíhá pouze v horizontální rovině a to v takové výšce od země, ve které se senzor nachází. Pro použití v autonomním vozítku, je tedy důležité umístit senzor v takové výšce, aby překážku mohl zachytit. Pokud by například výška překážky byla 10 cm a paprsek senzoru by prováděl skenování ve výšce 15 cm od země, tak by se překážka skryla mimo zorné pole skeneru a nebyla by detekována. Při použití RPLIDARu je tedy nutné brát zřetel na velikost překážek, které mají být detekovány (jiná vozítka, nebo statické překážky), a podřídit jim umístění senzoru tak, aby tyto překážky zachytil. Toto skenování v jedné rovině a nikoliv i v nějakém rozsahu osy z představuje pro použití v soutěži autonomních vozítek drobnou komplikaci, ovšem při správném umístění senzoru je tato nevýhoda prakticky eliminována. Tato vlastnost je dána

určením senzoru RPLIDAR, který je doporučený pro použití v robotických vysavačích, kterým prostřednictvím svého skenování poskytuje informace o vzdálenosti stěn a předmětů v místnosti. Tyto překážky prakticky vždy začínají téměř, nebo zcela u země a dosahují výšky desítek až stovek centimetrů a představují tak pro robotický vysavač překážku, které se musí vyhnout a proto je skenování pouze v jedné rovině naprosto dostačující, jelikož tyto překážky jsou vždy zachyceny. Schopnost detekce v závislosti na umístění 2D a 3D LIDARu znázorňuje obrázek 3.11.

Obrázek 3.10: Senzor RPLIDAR A1 (převzato z [29]).



Obrázek 3.11: Autíčko vlevo znázorňuje situaci kdy je 2D LIDAR (zelený blok) umístěn dostatečně nízko nad zemí aby jeho paprsek (červená čára) narazil na překážku představovanou oranžovou kostkou. Prostřední autíčko znázorňuje nevhodné umístění 2D LIDARu ve velké výšce, kdy překážka v dráze autíčka zůstane nedetekována. Na autíčku vpravo je LIDAR umístěn rovněž vysoko nad zemí, ovšem jedná se o 3D LIDAR, který vysílá paprsky i ve směru osy z a tím pádem na rozdíl od 2D LIDARu, umístěného ve stejné výšce, dokáže překážku zachytit.

Kapitola 4

Návrh autonomního řízení pohybu po dráze

Tato kapitola se zabývá základní platformou, která bude použita pro realizaci autonomního vozítka a výběrem vhodné řídící desky, včetně způsobu detekce závodní dráhy, pro zamýšlené použití. Kapitola vychází z poznatků uvedených v kapitole 2 o jednotlivých soutěžích a zejména z kapitoly 3, jež se zabývala způsoby detekce dráhy a možnými technickými řešeními tohoto problému, z nichž bude navržené řešení vycházet.

4.1 Základní platforma pro realizaci

Poněvadž je tato práce v určitém smyslu pokračovatelem diplomové práce Ing. Viktora Steingarta [40] tak, co se základní platformy autíčka týče, neprobíhal žádný výběr, jelikož byla k dispozici celá podvozková platforma tak bylo rozhodnuto ji využít pro další realizaci, přičemž je plánováno využití podvozku s elektromotory a akumulátorem.

Na počátku této práce nebylo jasné kdy a zda vůbec se uskuteční další ročník soutěže NXP Cup. Právě z tohoto důvodu bylo rozhodnuto se nevázat přímo na pravidla této soutěže a využít této příležitosti k vyzkoušení jiných komponentů a možností realizace autonomního vozítka. Práce si však z této soutěže propůjčuje pravidla a disciplíny, pro které je vozítko navrhováno, tak aby bylo možné stanovit metriky, podle kterých bude úspěšnost samotné realizace hodnocena. Neúčastí v soutěži získaná volnost, co se výběru komponentů a technických řešení týče, se může v dalších fázích vývoje ukázat být jak pozitivním aspektem, v podobě získání poznatků a výsledků, které by při omezování se na pravidla nebylo možné získat, tak i negativním aspektem, kdy snaha vyzkoušet některý netradiční přístup povede do slepé uličky a zdržením vývoje dojde k tomu, že výsledná realizace nebude natolik úspěšná, jako by byla realizace od začátku tvořená v souladu s pravidly soutěže. Zajímavým zjištěním na konci vývoje bude, zda například použití výkonnějších komponentů umožní dosáhnout lepších výsledků ve srovnání s vozítkem, které se drželo pravidel soutěže.

Podvozková platforma

Podvozek autíčka bude sestavený z kitu DFROBOT ROB0165 [8] o rozměrech $30 \times 16 \times 7$ cm (výška 30 cm při použití stožáru pro umístění kamery). Podvozek se skládá ze 2 hlavních hliníkových dílů, které jsou spolu spojeny prostřednictvím distančních sloupků, kdy ve vzniklém meziprostoru se nachází místo pro uložení akumulátoru a 2 stejnosměrných bezkartáčových elektromotorů A2212/15T (tabulka 4.1) určených pro pohon zadní nápravy.

Pro řízení těchto elektromotorů jsou využity regulátory BLS30A. O natáčení přední nápravy se stará servomotor MG996R (tabulka 4.2).

Napájení	7-12 V DC (max 12 A po dobu 60 s)
Příkon	135 W
Elektrická účinnost	80 %
Doporučená hmotnost modelu	300-700 g
Rozměry	27,7 mm × 26,3 mm
Váha	47 g

Tabulka 4.1: Parametry elektromotoru A2212/15T (čerpáno z [4]).

Napájení	4,8-7,2 V DC (500-900 mA při 6V)
Kroutící moment	9,4 kg/cm (4,8 V) nebo 11 kg/cm (6 V)
Rozměry	40,7 mm × 19,7 mm × 42,8 mm
Váha	55 g

Tabulka 4.2: Parametry servomotoru MG996R (čerpáno z [17]).

Na obrázku 4.1 se nachází sestavené autíčko v podobě, ve které se může účastnit soutěže NXP Cup, jak jej navrhnul Ing. Viktor Steingart. Z tohoto původního autíčka si tato práce propůjčuje výše zmíněné komponenty, zatímco pro detekci závodní dráhy a řízení autíčka budou použity jiné součástky. Jedním z cílů další práce je dosáhnout nižší výšky stožáru, na kterém je umístěna kamera. Její umístění je zvoleno z důvodu dostatečného rozhledu před autíčko. Mohlo by se zdát, že takto vysoko umístěná součástka by mohla ovlivňovat těžiště a tím pádem i stabilitu autíčka při zatáčení, avšak vzhledem k umístění akumulátorů a motorů blízko u země je těžiště stále velmi nízko a na jízdní vlastnosti bude mít snížení výšky stožáru poměrně malý, možná rovnou zanedbatelný vliv. Tímto cílem je tedy hlavně sledována estetičnost výsledného řešení, než jízdní vlastnosti. V dalším vývoji bude zajímavé sledovat, jakým způsobem výška umístění kamery ovlivňuje schopnost autíčka držet se na dráze a její výsledné umístění tedy bude dáno až samotným testováním při dalším vývoji a nelze nyní odhadnout konečnou výšku, ve které bude nakonec kamera umístěna.

Řídící jednotka a obrazový snímač

Z možných řešení detekce dráhy a řídících platforem, jenž byly rozebrány v kapitole 3, bude pro další vývoj použita detekce závodní dráhy prostřednictvím obrazového snímače, který, z nastíněných možností, představuje nejvhodnější řešení. Z hlediska vyzkoušení jiného přístupu se jeví jako nejlepší řešení použití platformy nVidia Jetson Nano (viz 3.2), která disponuje vysokým výpočetním výkonem, umožňujícím provádět zpracování obrazu z kamery v takové míře, v jaké to například na platformě RDDRONE-FMUK66 (viz 3.2 není možné. Z pohledu zpracování obrazu z kamery je možné využít i platformu Raspberry Pi (viz 2.2), primárně se však další vývoj bude odehrávat na platformě nVidia Jetson Nano. K vyzkoušení obou platforem nahrává i možnost použití stejných kamer, založených na snímači IMX219 (viz 3.7) nebo IMX477 (viz 3.4). Přestože snímač IMX477 nabízí kromě vyššího rozlišení i vyšší snímkovací frekvence (v odkazovaných tabulkách uvedené snímkovací frekvence se týkají Raspberry Pi, na nVidia Jetson Nano bylo v praktických zkouškách dosaženo vyšších hodnot), tak další vývoj bude probíhat zejména, nikoliv však výhradně, s

Obrázek 4.1: Vozítko pro soutěž NXP Cup dle Ing. Viktora Steingarta, na vrcholu stožáru se nachází kamera Pixi2 a hlavní řídící jednotkou vozítka je RDDRONE-FMUK66 (galerie autora).

kamerou vybavenou snímačem MX219, jejíž rozlišení 1280×720 při 60 snímcích za sekundu dokáže poskytovat dostatek obrazových informací ke zpracování. Navíc také použití tohoto rozlišení přináší teoretickou možnost, využití i druhé kamery, jelikož by pro ni ještě stále byl k dispozici výpočetní výkon, což by nemuselo platit v případě použití snímače o vyšším rozlišení s vysokou snímkovou frekvencí.

Kamerou získaná obrazová data budou analyzována řídící jednotkou, která s využitím vhodných algoritmů nalezne ve snímcích hraniční čáry závodní dráhy, určí pozici autíčka vůči těmto čarám a následně provede korekci směru a rychlosti jízdy takovým způsobem, aby autíčko nevyjelo z dráhy. Za tímto účelem budou hnací motory ovládány prostřednictvím přítomných regulátorů.

4.2 Detekce čar ve snímcích z obrazového snímače

Jedním z možných, výkonově efektivních, řešení takové detekce je Houghova transformace [41], jež je využitelná pro hledání takřka libovolných parametrizovatelných tvarů. Mějme přímku v rovině (x, y) určenou prostřednictvím rovnice 4.1 převoditelnou do tvaru 4.2.

$$y = ax + b \tag{4.1}$$

$$b = -ax + y \tag{4.2}$$

Nyní v rovině (a, b) vynášíme na vodorovnou osu parametr a, zatímco na svislou osu vynášíme parametr b, přičemž přímce 4.1 tak bude díky transformaci odpovídat jeden bod v této rovině. Při zvolení parametru a v rovnici 4.2 lze pro bod o souřadnicích (x_1, y_1) vypočítat zbývající parametr b prostřednictvím rovnice 4.3. Pokud v rovině (a, b) změníme parametr a dostaneme jako výsledek přímku.

$$b = -ax_1 + y_1 \tag{4.3}$$

V případě, že se v rovině (x, y) tvořenou obrazem nacházejí body se souřadnicemi (x_1, y_1), (x_2, y_2) a (x_3, y_3) tak lze, dosazením do rovnice 4.3 a následném zanesení výsledků do roviny (a, b), zjistit, že v bodě (a_x, b_x) se nachází průsečík tří přímek a tím pádem na přímce $y = a_x x + b_x$ leží tři body.

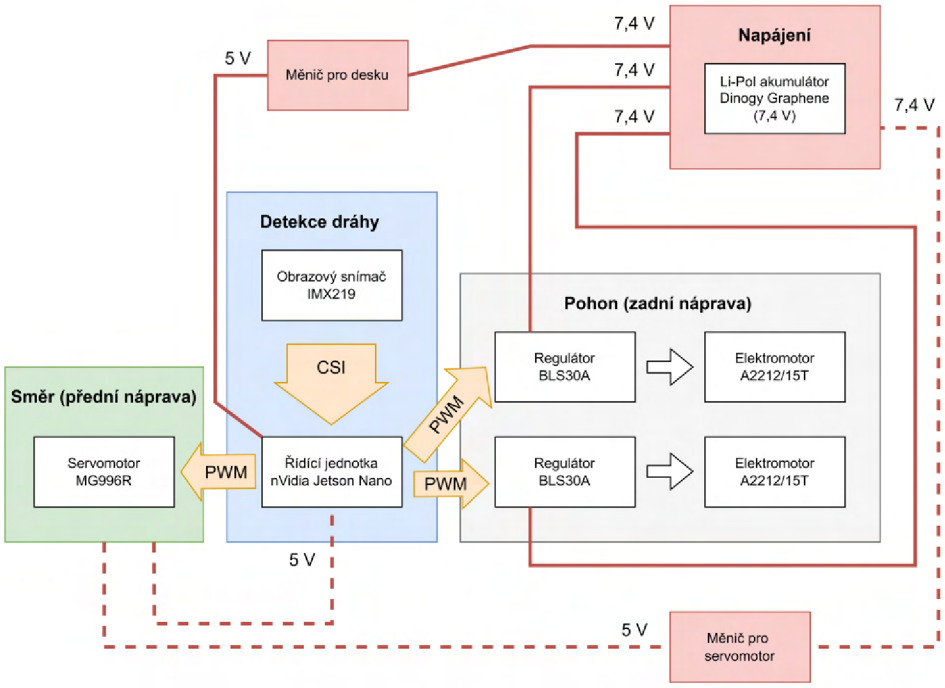
Pro výpočet v obrazu je snímek (rovina (a, b)), převedený do černobílé podoby, rozdělen na části, ve kterých je pro každý parametr a dopočítán pro každý bod parametr b. Nalezené body jsou postupně zaznamenávány, přičemž v bodě, ve kterém je zaznamenáno průsečíků nejvíce, odpovídají hodnoty parametrů $a = a_x$ a $b = b_x$ přímce $y = a_x x + b_x$.

4.3 Navržený řídící systém

Na obrázku 4.2 je navržený řídící systém robotického autíčka znázorněn prostřednictvím blokového schématu. Řídící jednotka představovaná vývojovou deskou Jetson Nano (parametry v tabulce 3.3) bude zodpovědná za veškeré řízení motorů na základě zpracovávaného vstupního obrazu z obrazového snímače, kterým je senzor IMX219 (s parametry 3.7). Snímky pořizované tímto snímačem budou do řídící jednotky přenášeny prostřednictvím CSI rozhraní. Získávané snímky projdou v řídící jednotce zpracováním jehož výstupem budou detekované hraniční čáry závodního okruhu. Tyto čáry budou následně vstupem pro samotný řídící algoritmus. Ten na jejich základě stanoví potřebné natočení přední nápravy, jež bude realizované servomotorem MG996R (viz 4.2), který bude řízen prostřednictvím PWM signálu (*Pulse-width modulation* neboli pulsně šířková modulace) o frekvenci 50 Hz z řídící jednotky. Rovněž budou prostřednictvím PWM signálu o frekvenci 50 Hz předávány pokyny regulátorům BLS30A, které následně na základě obdrženého signálu budou řídit rychlost dvou zadních elektromotorů A2212/15T, zajišťujících dopředný pohyb autíčka.

Návrh počítá s napájením celého systému prostřednictvím, přítomného Li-Pol akumulátoru Dinogy Graphene s kapacitou 1500 mAh a napětím 7,4 V (tabulka 5.1), z nějž budou přímo napájeny regulátory zadních elektromotorů. Pro napájení řídící jednotky bude nutné využít měnič na 5 V/3 A aby bylo zajištěno dostatečné napájení dle specifikace platformy Jetson Nano. Napájení předního servomotoru, který vyžaduje napájecí napětí v rozsahu 5-6 V, je na obrázku znázorněno přerušovanou červenou čarou dvěma možnostmi. První možností je napájení z akumulátoru přes měnič na 5 V, druhou možnost představuje napájení prostřednictvím 5 V pinu řídící jednotky Jetson Nano, jelikož ta na tomto pinu umožňuje i vyšší odběr než obvyklých 500 mAh, jako v případě mnohých dalších vývojových desek a nabízí se zde tedy možnost využít tohoto pinu pro napájení servomotoru bez nutnosti použít další měnič pro snižování napětí poskytovaného akumulátorem.

Navržený systém bude v průběhu samotné implementace řídícího algoritmu a jeho ladění upravován podle nově zjištěných poznatků, které mohou vézt k různým změnám a vylepšením oproti stávajícímu návrhu, jako například změna použitých senzorů, nebo komunikace. Vzhledem k poměrně dostupnému prostoru v útrobách vozítka se zde nabízí prostor i pro případná další potřebná hardwarová zařízení, která mohou být vyžadována pro zpracování vstupních dat, případně pro řízení vozítka.



Obrázek 4.2: Blokové schéma systému detekujícího dráhu a řídícího pohyb autíčka.

Kapitola 5

Implementace řídícího systému

Kapitola pojednává o hardwarové a softwarové implementaci řídícího systému a detekce dráhy, přičemž vychází z návrhu představeného v kapitole 4. V průběhu této fáze práce se vyskytlo mnoho technických problémů, z nichž některé byly velmi překvapivé. Tyto komplikace měly za následek četné změny a přehodnocení původního návrhu, který značně ovlivnily a proto budou v této kapitole podrobně rozebrány. Veškeré tyto problémy se vzájemně prolínaly a byly řešeny souběžně, avšak pro zjednodušení struktury textu bude tato kapitola členěna do čtyř podkapitol zabývajících se napájením celého systému, řízením elektromotorů, detekcí dráhy a následně samotným algoritmem pro řízení směru a rychlosti vozítka. Jelikož na sebe jednotlivé problémy navazovaly, tak nelze tyto podkapitoly od sebe zcela oddělit a pro přiblížení celkové situace se tyto jednotlivé podkapitoly někdy dotýkají i problémů spadajících do jiných podkapitol.

5.1 Napájení systému

Návrh řídícího systému (viz 4.3) představený v kapitole 4 počítal s napájením celého sytému prostřednictvím jednoho Li-Pol akumulátoru Dinogy Graphene 1500 mAh 2S 65C (tabulka 5.1). Tento akumulátor má kapacitu 1500 mAh a skládá se ze dvou sériově zapojených článků, které poskytují výstupní napětí 7,4 V. Značení 65C udává doporučené trvalé proudové zatížení tohoto akumulátoru, které je tedy rovno 65 násobku kapacity, což je 97,5 A. Takový výstupní proud by však akumulátor byl schopen dodávat pouze po dobu asi 73 sekund. Typický odběr elektromotorů A2212/15T je mnohem nižší a tedy i výdrž akumulátoru je v řádech desítek minut. V případě, kdy se akumulátorem bude napájet i přední servomotor a řídící jednotka se dá odhadovat, že se proudový odběr, v závislosti zejména na rychlosti zadních motorů, bude pohybovat zhruba v rozmezí 5-15 A, což by poskytovalo teoretickou výdrž v rozmezí 24-8 minut. Praktický proudový odběr z akumulátoru však bude nižší, jelikož zadní motory jsou při jízdě používány téměř na minimální rychlost a tedy maximální výdrž akumulátoru bude vyšší než odhadovaných 24 minut.

Oproti návrhu však v průběhu vývoje došlo k četným změnám a úpravám a jedna z těchto úprav se týkala také napájecího systému. Vzhledem k velmi špatné dostupnosti potřebných součástek v době vývoje, nebylo možné sehnat potřebný dodatečný stepdown měnič (případně součástky na jeho vyrobení) pro nápajení řídící desky Jetson Nano, jenž by byl schopný dodávat přes USB rozhraní proud 3 A, jinde než v zahraničí s dlouhými dodacími lhůtami. Právě pro urychlení vývoje bylo rozhodnuto použít pro napájení řídící desky jednoduše dostupnou dodatečnou powerbanku s 3 A USB výstupem. Použitá powerbanka

Kapacita	1500 mAh
Konfigurace	2S
Napětí	7,4 V
Trvalé proudové zatížení	65C
Max. proudové zatížení	130C
Rozměry	86 mm × 35 mm x 14 mm
Váha	80 g
Maximální nabíjecí poměr	3C/4,5 A
Servisní konektor	JST-XH
Silový konektor	XT60

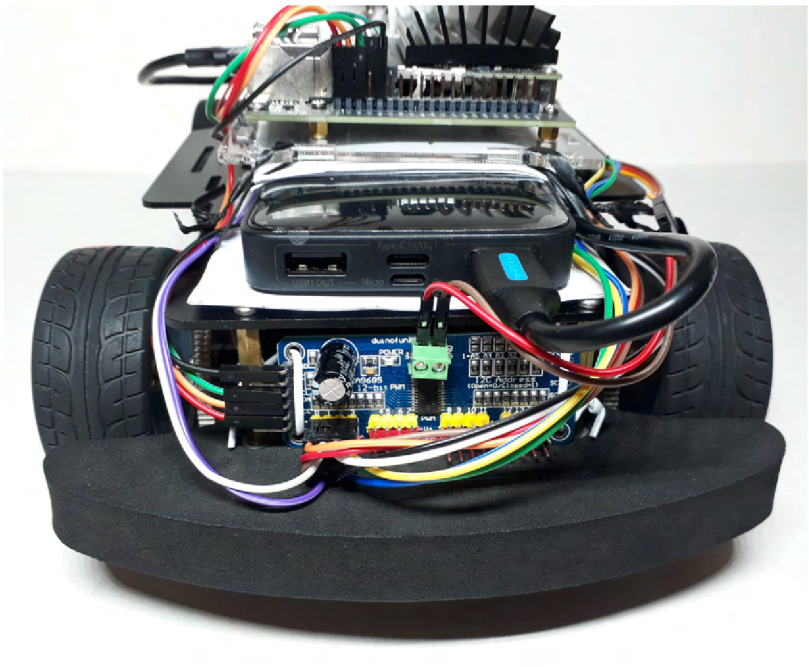
Tabulka 5.1: Parametry akumulátoru Dinogy Graphene 1500 mAh 2S-65C (čerpáno z [9]).

Baseus Bipow (tabulka 5.2) s kapacitou 10 000 mAh disponuje 3 A výstupem a dokáže tedy napájet Jetson Nano v požadovaném scénáři bez toho, aby docházelo ke zpomalování frekvence CPU z důvodu nedostatečného napájení. K tomuto jevu docházelo v počátečních pokusech s platformou Jetson Nano při používání nedostatečného napájecího adaptéru s výstupním proudem 2 A a také v situaci, kdy regulátorem Foxy UBEC, poskytujícím právě 3 A výstup, byl kromě řídící desky Jetson Nano napájen i servomotor MG996R.

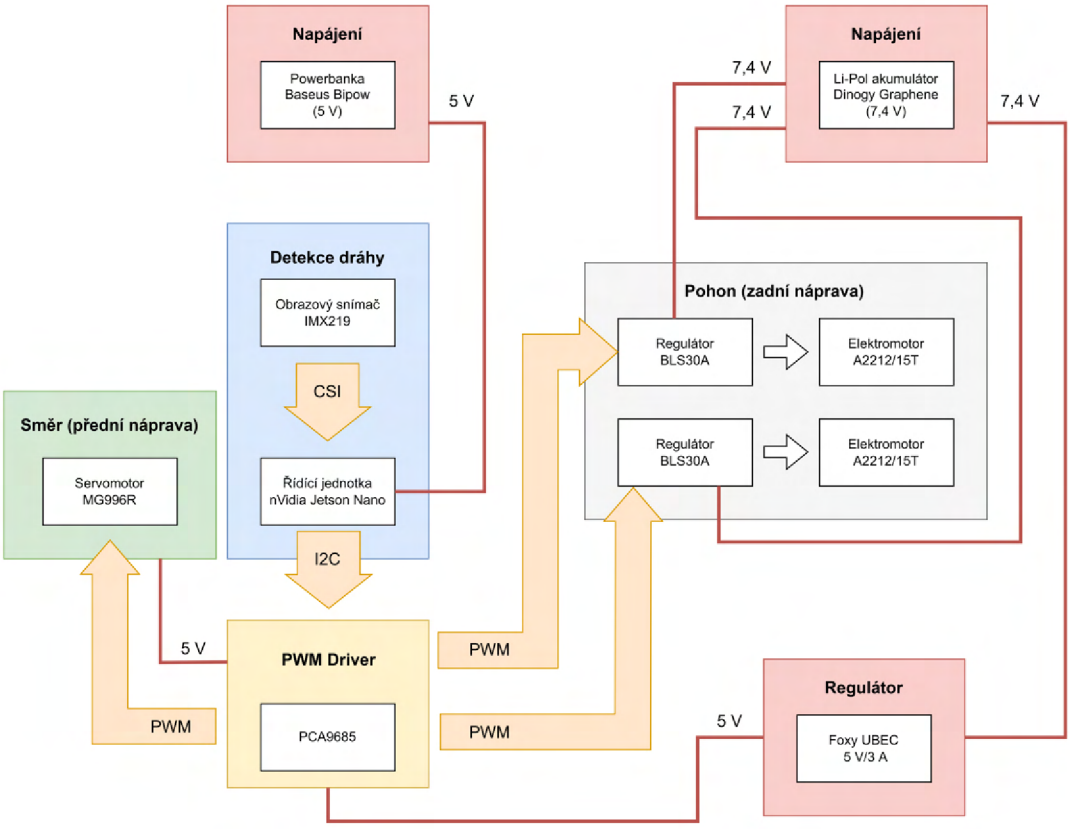
Kapacita	10 000 mAh
Počet výstupů	3 (2 USB-A, 1 USB-C)
Počet vstupů 2	(1 USB-C, 1 Micro USB)
Výstupní napětí a proud	5 V/3 A
Vstupní napětí a proud	5 V/3 A
Rozměry	151 mm × 69 mm x 17 mm

Tabulka 5.2: Parametry powerbanky Baseus Bipow Digital Display Power bank 10000mAh 15W Black (čerpáno z [3]).

Použití powerbanky umožňuje v konečné podobě autíčka použít na autíčku původně přítomný regulátor napětí Foxy UBEC, který byl v jeho původní podobě použit pro napájení řídící desky RDDRONE-FMUKK66, pouze k napájení servomotoru. V případě kdy byla z tohoto regulátoru napájena i řídící deska docházelo ve chvílích vysoké zátěže procesoru ke kolísání napájecího napětí, jež se projevovalo již zmíněným snižováním frekvence procesoru a v krajních případech dokonce i vypnutím řídící desky. Regulátor Foxy UBEC sice krátkodobě poskytuje možnost dodávat až 5 A, avšak s rostoucí proudovou zátěží v podobě dalších součástek při postupujícím vývoji již nedostačoval. Tento regulátor reguluje výstup akumulátoru Dinogy Graphene o napětí 7,4 V na výstup 5 V/3 A, který je využit pro napájení servomotoru a pomocného obvodu PCA9685, který, jak je rozebráno dále v části 5.2, se ukázal být nezbytným pro řízení veškerých elektromotorů autíčka. Obrázek 5.1 zachycuje zapojení obvodu PCA9685 včetně jeho napájení připojeného na zelené svorky a také USB výstup powerbanky, z něhož je napájena řídící deska Jetson Nano.



Obrázek 5.1: Detail zapojení napájení obvodu PCA9685 prostřednictvím výstupu regulátoru Foxy UBEC a powerbanky napájející platformu Jetson Nano (galerie autora).



Obrázek 5.2: Blokové schéma řídícího systému a jeho napájení.

Autíčko má tedy ve svojí výsledné podobě, určené pro detekci závodní dráhy a jízdy po ní, 2 napájecí okruhy. První 7,4 V okruh představovaný Li-Pol akumulátorem zajišťuje napájení dvou zadních elektromotorů A2212/15T prostřednictvím regulátorů BLS30A a prostřednictvím regulátoru Foxy UBEC, z něhož je napájen obvod PCA9685, je zajišťováno napájení předního servomotoru MG996R. Druhý 5 V napájecí okruh je tvořen powerbankou Baseus Bipow a zajišťuje napájení řídící jednotky Jetson Nano a k ní připojených periférií. První napájecí okruh tedy napájí veškeré pohyblivé komponenty, zatímco druhým napájecím okruhem jsou napájeny komponenty řídícího systému. Celkové schéma napájecího systému je znázorněno na obrázku 5.2.

5.2 Řízení elektromotorů

Pro účely jízdy po dráze je, kromě detekce dráhy, nutné zajistit řízení motorů vzhledem ke zjištěné pozici autíčka na závodní dráze. Jak již bylo zmíněno v části rozebírající navržený řídící systém (viz 4.3) tak řízení jak předního servomotoru, tak i zadních bezkartáčových elektromotorů je prováděno skrze PWM signál o frekvenci 50 Hz. Tato sekce se ve své první části (5.2) věnuje teorii týkající se řízení použitých bezkartáčových elektromotorů a servomotoru. V druhé části (5.2) se zabývá praktickými poznatky a objevenými problémy týkajících se jejich řízení souvisejícími s použitou platformou Jetson Nano a poslední část (5.2) se věnuje jejich řešení.

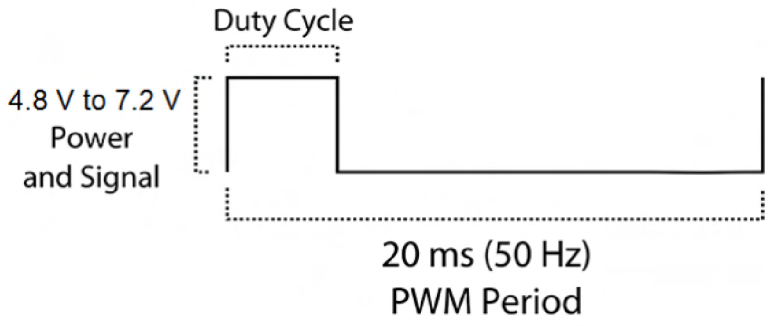
Teorie řízení elektromotorů

V případě předního servomotoru MG996R (viz 4.2), jenž ovládá natočení přední nápravy, se jedná o přímé řízení, kdy PWM signál přímo řídí daný servomotor. V případě zadních bezkartáčových elektromotorů A2212/15T je PWM signál zpracováván regulátorem BLS30A, který následně postupně spíná cívky motoru v takovém pořadí a rychlosti, aby se motor točil požadovaným směrem a rychlostí.

Řízení úhlu natočení servomotoru je realizováno prostřednictvím změny střídy PWM signálu. Při frekvenci 50 Hz je jedna perioda takového signálu rovna 20 ms, přičemž právě délkou doby, kterou je z této periody PWM signál na vysoké logické úrovni je řízeno natočení servomotoru. Servomotor MG996R je řízen obdobně jako řada dalších servomotorů, tzn. že 1,5 ms trvání vysoké logické úrovně a po zbytek periody nízká logická úroveň představuje neutrální pozici servomotoru, 1 ms dlouhá vysoká logická úrovně odpovídá plnému otočení doleva, zatímco 2 ms plnému otočení doprava (řídící signál je zobrazen na obrázku 5.3). Jakýkoliv PWM signál mezi těmito krajními hodnotami pak vede k natočení servomotoru na tomu odpovídající pozici.

Obdobným způsobem, jakým je řízen servomotor, je řízen i bezkartáčový motor prostřednictvím regulátoru. V tomto případě představuje 1,5 ms trvání vysoké logické úrovně neutrální hodnotu a tedy stojící motor, 2 ms představuje maximální dopřednou rychlost, zatímco periody s 1 ms vysoké logické úrovně představují maximální rychlost vzad. Signály s dobou trvání vysoké úrovně mezi 1,5-2 ms tedy určují dopřednou rychlost pohybu, zatímco signály s dobou vysoké úrovně v rozmezí 1-1,5 ms určují rychlost pohybu vzad.

Řízení bezkartáčových elektromotorů prostřednictvím ESC (*Electronic Speed Control*) regulátorů však má na rozdíl od řízení servomotoru svá specifika. Tato specifika jsou dána původem ESC regulátorů, které jsou hojně používané pro řízení dálkově ovládaných létajících modelů a dronů, kdy zde je pohyb takového prostředku řízen prostřednictvím páček na ovladači, kterými pilot plynule pohybuje a mění tak rychlost a směr modelu. Právě onen



Obrázek 5.3: Perioda signálu pro řízení servomotoru MG996R (převzato z [17]).

plynulý pohyb pomyslnou plynovou páčkou je potřeba napodobit v řídícím programu pro prvotní rozpohybování vozítka. Z nulové rychlosti není možné ihned roztočit elektromotory nastavením PWM signálu s 1,75 ms trvající vysokou logickou úrovní, namísto toho je nutné po krůčcích (například o velikosti 0,01 ms) zvyšovat délku trvání vysoké logické úrovně v rámci periody z 1,5 ms až na hodnotu 1,75 ms pro dosažení požadované rychlosti a to s rozumnými časovými intervaly mezi jednotlivými změnami PWM signálu. Po tomto prvotním rozběhnutí je již možné motory zastavit okamžitou změnou na periodu s 1,5 ms trvající vysokou logickou úrovní a opět rozběhnout pouhou změnou například zpátky na hodnotu 1,75 ms bez nutnosti opakovat pozvolné zvyšování doby trvání vysoké logické úrovně na požadovanou hodnotu.

Kromě tohoto specifika je ještě před samotným rozběhnutím motorů nutné provézt aktivační sekvenci ESC regulátorů. Ta se v závislosti na modelu ESC regulátoru skládá z nastavení PWM signálu bez jakékoliv vysoké logické úrovně po dobu 1 s následované změnou na signál s maximálním výkonem vpřed, tj. perioda s 2 ms trvající vysokou logickou úrovní opět po dobu 1 s, následované signálem s minimálním výkonem, tj. periodou s 1 ms trvající vysokou logickou úrovní taktéž po dobu 1 s. Po takto provedené sekvenci, jež je doprovázena patřičným kontrolním zvukovým signálem ze strany ESC regulátorů, dojde k odjištění ovládání rychlosti motorů a je tedy možné provézt plynulé zvýšení dopředné či zpáteční rychlosti a rozpohybovat tak motory. Po tomto prvotním rozpohybování, je již možné motory řídit pouhou změnou na signál odpovídající požadované rychlosti bez nutnosti jeho postupné změny, jak již bylo zmíněno výše.

Řízení elektromotorů na platformě Jetson Nano

Platforma Jetson Nano se prezentuje jako obdoba platformy Raspberry Pi a je zde snaha o co nejvíce podobné ovládání a vlastnosti s cílem, aby uživatelé, kteří jsou již seznámeni s Raspberry Pi, mohli jednoduše vyměnit platformu a bez obtíží pokračovat na platformě Jetson Nano. Z pohledu této práce je viditelnou součástí těchto snah například možnost využívat kamery se snímačem IMX219 a IMX477, na nichž jsou postaveny velmi rozšířené kamery pro RPi. Dalším podobným bodem je snaha o co nejpodobnější programování. Pro účely této práce, byl zvolen programovací jazyk Python[1] (volba bude více rozebrána v dalších sekcích), pro který existuje knihovna RPi.GPIO[2], která umožňuje pracovat s GPIO

[1] https://www.python.org/
[2] https://pypi.org/project/RPi.GPIO/

piny platformy Jetson Nano prostřednictvím stejných funkcí jako na platformě Raspberry Pi a to včetně stejného chování jako na platformě Raspberry Pi a tedy přechod z programování Raspberry Pi na Jetson Nano by měl být bez sebemenších problémů.

Zde nastává problém v tom, že samotná hardwarová implementace některých funkcí a mechanismů je na platformě Jetson Nano realizována odlišným, v rámci této práce dosud neobjasněným, způsobem a tak principy a řízení, které fungují na platformě Raspberry Pi nemusejí fungovat na platformě Jetson Nano, přestože se tak Jetson Nano prezentuje.

Nejvýrazněji se tyto rozdílnosti projevily v případě generování PWM signálu, kdy se ukázalo, že je v podstatě nemožné řídit přední servomotor a provádět přesné úpravy rychlosti zadních elektromotorů prostřednictvím dostupného PWM generátoru.

```
import RPi.GPIO as GPIO
import time

#instead of GPIO pin number the number represents physical pin on board
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
#GPIO.setmode(GPIO.BCM)

#PWM output on pin 7 with 50Hz frequency
GPIO.setup(7, GPIO.OUT)
servo_motor = GPIO.PWM(7,50)
servo_motor.start(0)

while 1:
    #rotate shaft to the NEUTRAL position
    servo_motor.ChangeDutyCycle(7.5)
    time.sleep(2)

    #rotate shaft to the LEFT
    servo_motor.ChangeDutyCycle(6.0)
    time.sleep(2)
    servo_motor.ChangeDutyCycle(5.0)
    time.sleep(2)
    servo_motor.ChangeDutyCycle(4.0)
    time.sleep(2)
    servo_motor.ChangeDutyCycle(3.0)
    time.sleep(2)
    servo_motor.ChangeDutyCycle(2.5)
    time.sleep(2)
```

Na ukázce kódu výše se nachází krátký program, v programovacím jazyce Python, určený k testování ovládání servomotoru. Tento kód je teoreticky přenositelný z Raspberry Pi na Jetson Nano, jelikož by měl v této podobě fungovat stejně na obou platformách, ovšem není tomu tak. Tento program nastaví pin číslo 7 jako PWM výstup s frekvencí 50 Hz, následně v hlavní smyčce dochází nastavení pozice servomotoru na výchozí pozici, kdy po uplynutí doby 2 s dojde k pootočení hřídele servomotoru vlevo (při pohledu shora) a poté každé další 2 s se hřídel otočí o další krok vlevo až se dostane na maximální otočení. Nekonečná smyčka tuto sekvenci stále opakuje. Takto servomotor funguje v případě, kdy je tímto programem řízen z platformy Raspberry Pi, po přenesení na platformu Jetson Nano a úpravě používaného pinu na GPIO pin s PWM výstupem (například pin 32 nebo 33, jelikož PWM je na Jetson Nano dostupné na jiných pinech) bylo zjištěno, že ve chvíli kdy se nastaví PWM signál, který by měl pouze jemně otočit hřídel a zastavit na cílové pozici, tak namísto toho otáčení hřídele pokračuje až do momentu, kdy se otočí na maximální

úhel a začne docházet k přetáčení servomotoru. Pro zabránění poškození hřídele a vnitřních součástí motoru je nutné odpojit napájení a následně vrátit hřídel ručně do výchozí pozice. Jakékoliv pokusy s modifikací výše ukázaného kódu nevedly na platformě Jetson Nano k výsledku, jakého bylo dosahováno na platformě Raspberry Pi. Pokusy se samotným PWM pro jiné použití, například pro regulaci jasu LED, ukázaly, že na platformě Raspberry Pi i Jetson Nano je v takovém scénáři dosahováno stejných výsledků. Regulace jasu fungovala na obou platformách stejně, ovšem v případě řízení servomotoru se stejná implementace chovala zcela rozdílně.

```
import time
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)

#right rear motor setup
GPIO.setup(33, GPIO.OUT)
right_motor = GPIO.PWM(33, 50)

#left rear motor setup
GPIO.setup(32, GPIO.OUT)
left_motor = GPIO.PWM(32, 50)


#arming sequence - NO PWM 1s / MAX POWER 1s / MIN POWER 1s
right_motor.start(0)
left_motor.start(0)
print ("ARMING sequence - 0 (1s)")
time.sleep(1)

right_motor.ChangeDutyCycle(10)
left_motor.ChangeDutyCycle(10)
print("ARMING sequence - MAX power (1s)")
time.sleep(1)

right_motor.ChangeDutyCycle(5)
left_motor.ChangeDutyCycle(5)
print("ARMING sequence - MIN power (1s)")
time.sleep(1)

print("Should be ARMED")

i =7.0

while True:
  while i<7.65:
    print(i)
    right_motor.ChangeDutyCycle(i)
    left_motor.ChangeDutyCycle(i)
    time.sleep(.05)
    i +=.01
    #print("PAUSE")
  time.sleep(4)

  while i>7.2:
    print(i)
    right_motor.ChangeDutyCycle(i)
    left_motor.ChangeDutyCycle(i)
```

```
49      time.sleep(.05)
50      i -=.05
51    time.sleep(4)
```

Problémy s řízením se však týkaly i zadních elektromotorů řízených prostřednictvím ESC regulátorů. Na ukázce kódu výše je program určený pro testování motorů, který funguje na platformě Jetson Nano i na platformě Raspberry Pi, ovšem na každé platformě vykazuje jiné chování. Na úvod jsou piny 32 a 33 nastaveny jako PWM výstup o frekvenci 50 Hz následně proběhne aktivační sekvence ESC regulátorů, po jejímž provedení je možné spustit motory. V hlavní nekonečné smyčce programu se nacházejí dvě smyčky, jedna z nich roztočí motor pro pohyb autíčka směrem dopředu, zatímco druhá pro pohyb dozadu. Tato sekvence pohybu se neustále opakuje.

Zatímco v případě platformy Raspberry Pi bylo možné při pokusech dosáhnout řízení rychlosti motorů po nejmenších možných krůčcích tak, že bylo možné pohybovat autíčkem dopředu v nízkých jednotkách centimetrů za sekundu, ale také rychlostí v řádech metrů za sekundu, tak na platfromě Jetson Nano takové kontroly nebylo možné dosáhnout. Nejnižší dopředná rychlost jaké se podařilo v experimentech na platformě Jetson Nano dosáhnout byla přibližně 3 m/s, přičemž regulace rychlosti byla velmi obtížná. Nejnižší dosažená zpáteční rychlost byla podstatně nižší, než rychlost dopředná, proto se nabízelo vyzkoušet prohození zadních motorů mezi sebou a pro dopřednou jízdu používat zpáteční rychlost, která se zpočátku jevila jako použitelná. Další experimenty s řídícím algoritmem však ukázaly, že se autíčko pohybuje příliš rychle na to, aby bylo možné postupně vyladit řídící algoritmus jelikož autíčko neustále opouštělo závodní dráhu vysokou rychlostí, která se pro experimenty nehodila.

Výsledkem veškerých experimentů s řízením servomotoru a elektromotorů byl poznatek, že na platformě Jetson Nano není možné prostřednictvím integrovaného PWM generátoru ovládat servomotor a je velmi obtížné regulovat rychlost zadních motorů pro pomalou jízdu. Na druhou stranu bylo zjištěno, že na platformě RPi je možné bez problémů ovládat jak servomotor tak elektromotory. Z tohoto důvodu bylo přijato rozhodnutí zkusit přenést, v tu dobu již plně implementovanou, detekci čar z platformy Jetson Nano na Raspberry Pi, ovšem to se setkalo s neúspěchem, jelikož na platformě Raspberry Pi nebylo možné zprovoznit kameru (problém je podrobněji rozebrán v sekci 5.2). Vývoj se tedy dostal do další slepé uličky - fungující detekce čar na platformě Jetson Nano a fungující řízení veškerých motorů na platformě Raspberry Pi. Bylo nutné nalézt řešení, jak na jedné platformě zprovoznit obě věci současně.

Řešení problémů s řízením

Z výzkumu dalších pohyblivých modelů realizovaných na platformě Jetson Nano a Raspberry Pi a podobných problémů s řízením elektromotorů dalších uživatelů, bylo zjištěno, že Jetson Nano ve scénáři, kdy má být prostřednictvím jeho PWM výstupu řízen servomotor, případně jiný elektromotor není, z blíže neznámého důvodu, schopen poskytovat PWM signál v takové podobě, která by umožňovala řízení těchto prvků stejně, jako je to možné na platformě Raspberry Pi. Pro řízení těchto prvků, jež vyžadují přesný PWM signál, je nutné využít externí modul generující PWM signál, který je zapojen mezi Jetson Nano a prvky které tento PWM signál přijímají.

Takovým modulem je například modul PCA9685 (tabulka 5.3), což je 16 kanálový 12-bitový PWM driver. S modulem je ze strany Jetson Nano možné komunikovat rozhraním I2C, kterým jsou předávány vstupní informace jako adresa cílového zařízení, frekvence a

požadovaná střída PWM signálu. K tomuto zařízení jsou jak pro platformu Jetson Nano, tak i pro Raspberry Pi dostupné Python knihovny adafruit-circuitpython-pca9685[3] a adafruit-circuitpython-servokit[4]. Tyto knihovny poskytují jednoduché rozhraní pro práci s modulem PCA9685.

Frevence	40-1000 Hz
Počet kanálů	16
Rozlišení	12 bitů
Napětí	DC 5-10
Rozměry	60 mm × 25 mm x 20 mm

Tabulka 5.3: Parametry modulu PCA9685 (čerpáno z [13]).

Objevenou komplikací týkající se těchto knihoven však bylo, že již využívají některé prvky jazyka Python verze 3.7, zatímco na platformě Jetson Nano je standardně distribuována verze 3.6, ve které nelze skripty, využívající některé funkce těchto knihoven, spustit. Pro úspěšné spuštění skriptu využívajícího těchto knihoven bylo nutné doinstalovat vývojovou verzi jazyka Python 3.7, ve kterém již bylo možné následující kód spustit a otestovat.

```
from adafruit_servokit import ServoKit
from time import sleep

kit = ServoKit(channels=16)

kit.servo[0].angle=100
print("100 - neutral")
sleep(2)

#turn RIGHT
print("RIGHT")
kit.servo[0].angle=110
print("110")
sleep(2)

kit.servo[0].angle=120
print("120")
sleep(2)

kit.servo[0].angle=130
print("130")
sleep(2)

kit.servo[0].angle=140
print("140")
sleep(2)

kit.servo[0].angle=150
print("150")
sleep(2)

kit.servo[0].angle=160
print("160")
sleep(2)
```

[3]https://pypi.org/project/adafruit-circuitpython-pca9685/
[4]https://pypi.org/project/adafruit-circuitpython-servokit/

```
35
36 kit.servo[0].angle=170
37 print("170")
38 sleep(2)
39
40 kit.servo[0].angle=180
41 print("180")
42 sleep(2)
```

Kód z ukázky výše provede natočení servomotoru do výchozí pozice a následně po 10° krůčcích každé 2 s otáčí hřídelí až do maximálního vytočení vpravo. Vývoj se tedy nyní dostal do fáze, kdy již na platformě Jetson Nano fungovala kamera s detekcí čar v jazyce Python verze 3.6 a řízení servomotoru ve verzi 3.7 a bylo nutné spojit tyto dva skripty do jednoho a to přenesením již fungující detekce čar do novější verze jazyka Python. Tento krok se však ukázal být nepřekonatelnou překážkou, jelikož při spuštění skriptu obstarávajícího detekci čar v Pythonu 3.7 nedošlo k inicializaci a připojení kamery a skript vždy skončil bez výsledku.

Na ukázce kódu níže je vidět úsek programu pro detekci čar využívajícího knihovny nanocamera[5] pro práci s kamerou, který funguje v Pythonu verze 3.6 ovšem nikoliv verze 3.7, jelikož nedojde ke správnému provedení řádku 12 (pozn. v ukázce je používána inicializace s parametry pro kameru se senzorem IMX477, jelikož v průběhu vývoje došlo k přechodu ze snímače IMX219 na IMX477 a následně zpět na IMX219).

```
1  #ROAD LANE DETECTION
2
3  import cv2
4  import matplotlib.pyplot as plt
5  import numpy as np
6  import nanocamera as nano
7
8  #IMX 219
9  #camera = nano.Camera(flip = 0, width = 1280, height = 720, fps = 120)
10
11 #IMX477
12 camera = nano.Camera(flip = 0, width = 1920, height = 1080, fps = 60)
13
14 image1 = camera.read()
```

Snaha využít jiné řešení než knivovnu nanocamera a to openCV[6] pro získávání snímků z kamery se rovněž nesetkala s úspěchem a vývoj se tedy opět dostal do slepé větve, nyní však již nebyla detekce čar na jiné platformě než řízení motorů. Byla na stejné platformě, ale byla v jiné verzi programovacího jazyka, která vylučovala spojení dvou funkčních částí do jednoho řídícího programu.

Pokusy o úpravu knihoven a nahrazení prvků jazyka Python 3.7 nepřinesly žádný pokrok. Jako slibná varianta se jevila možnost nainstalovat starší verze těchto knihoven, jelikož dostupné internetové tutoriály s modulem PCA9685 na platformě Jetson Nano z minulých let neměly tento problém a knihovny fungovaly ve verzi jazyka 3.6. Odinstalace knihoven a instalace starších verzí ale tento problém nedokázala odstranit, jelikož tyto knihovny mají určité závislosti a stále docházelo k instalaci modulů v novějších verzích vyžadujících Python 3.7.

[5] https://pypi.org/project/nanocamera/
[6] https://pypi.org/project/opencv-python/

Bylo tedy nutné buďto zprovoznit kameru v Pythonu 3.7, nebo nějakým jiným způsobem dokázat ovládat modul PCA9685 z Pythonu 3.6. Konečné řešení tohoto problému bylo nalezeno v knihovně smbus[7], která poskytuje relativně nízkoúrovňové rozhraní pro práci s I2C rozhraním, kdy prostřednictvím posílání informací v bytech je možné komunikovat s modulem PCA9685 a díky tomu, že tato knihovna funguje v Pythonu verze 3.6, tak je možné řídit motory i provádět detekci čar na jedné platformě v jednom skriptu.

```
import smbus

#------------------------------------------------
# front servo motor communication via PCA9685
#------------------------------------------------

#centered servo position
neutral_servo_angle = 305
#actual servo angle
servo_angle = 305

BOARD_I2C_ADDR = 0x40
CHANNEL_0_START = 0x06
CHANNEL_0_END = 0x08
MODE1_REG_ADDR = 0
PRE_SCALE_REG_ADDR = 0xFE

bus = smbus.SMBus(1)

# Enable prescaler change
bus.write_byte_data(BOARD_I2C_ADDR, MODE1_REG_ADDR, 0x10)

# Set prescaler to 50Hz
bus.write_byte_data(BOARD_I2C_ADDR, PRE_SCALE_REG_ADDR, 0x80)
time.sleep(.25)

# Enable word writes
bus.write_byte_data(BOARD_I2C_ADDR, MODE1_REG_ADDR, 0x20)

# Set channel start times
bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_0_START, 0)  # 0us

#neutral servo position
bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_0_END, int(
    neutral_servo_angle))
```

Vyjmutá část kódu výše provede inicilaizaci modulu PCA9685, který má adresu 0x40 a nultého kanálu začínajícího na adrese 0x06 a končícího na adrese 0x08. Následně dojde k nastavení frekvence pro generování PWM signálu na 50 Hz, nastavení komunikace s nultým kanálem a na závěr dojde k natočení servomotoru do jeho neutrální pozice.

Obdobným způsobem bylo přepracováno i ovládání zadních elektromotorů, které byly zapojeny na kanály 1 a 2, což přineslo mnohem přesnější řízení rychlosti motorů a bylo tak možné dosáhnout stejných výsledků, jakých bylo dosahováno na platformě Raspberry Pi prostřednictvím integrovaného PWM generátoru. Modul PCA9685 umožňuje touto cestou dosáhnout pohybu autíčka po centimetrech za sekundu, stejně jako po metrech za sekundu a poskytuje velmi přesnou regulaci rychlosti elektromotorů, která dříve, prostřednictvím integrovaného PWM generátoru platformy Jetson Nano, nebyla možná.

[7] https://pypi.org/project/smbus2/

Níže je implementace aktivační sekvence ESC regulátorů zadních elektromotorů prostřednictvím knihovny smbus, komunikující s modulem PCA9685. Na úvod je po dobu 1 s držen výstup s nulovou střídou, následuje 1 s blok se střídou odpovídající maximální rychlosti vpřed (jelikož v průběhu vývoje a pokusů došlo k již zmíněnému prohození zadních elektromotorů ve snaze využít jejich pomalejší rychlost vzad pro pohyb dopředu a již nebyla pozice motorů vrácena, tak po aktivaci ESC regulátorů zadání hodnoty 450 pro maximální rychlost vpřed vyvolá pohyb autíčka maximální rychlostí vzad), zakončené 1 s blokem s maximální zpáteční rychlostí (vzhledem k otočeným motorům se ve skutečnosti jedná o pohyb autíčka dopředu), čímž dojde k aktivaci řízení elektromotorů a následně je možné provézt prvotní pozvolné zvýšení výkonu na požadovanou hodnotu a rozpohybovat autíčko.

```
CHANNEL_1_START = 0x0A
CHANNEL_1_END = 0x0C
CHANNEL_2_START = 0x0E
CHANNEL_2_END = 0x10

# Set channel start times
bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_1_START, 0)
bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_2_START, 0)

bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_1_END, 0)
bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_2_END, 0)
print("NO PWM")
time.sleep(1)

bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_1_END, 450)
bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_2_END, 450)
print("MAX POWER")
time.sleep(1)

bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_1_END, 100)
bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_2_END, 100)
print("MIN POWER")
time.sleep(1)

print("should be ARMED")
```

5.3 Detekce závodní dráhy

Hardware systému pro detekci závodní dráhy, vycházející z návrhu v sekci 4.1, prodělal v průběhu vývoje rovněž četné změny, než se ustálil ve své konečné podobě. Původně navrhovaný systém počítal s využitím platformy Jetson Nano a obrazového snímače IMX219, vzhledem ke komplikacím nastíněným v sekci 5.2 byla platforma Jetson Nano na krátkou dobu vyměněna za platformu Raspberry Pi. Nakonec se však vývoj ustálil na původně avizované platformě Jetson Nano, ovšem původně plánovaný snímač IMX219 byl v jedné z fází vývoje nahrazen snímačem IMX477. K tomuto došlo z důvodu snah o snížení výšky stožáru na původním modelu (obrázek 4.1), na kterém byla umístěna kamera. Snímač IMX477 (viz 3.4) má větší činnou plochu než snímač IMX219 (viz 3.7) spolu s vyšším rozlišením, což se projevovalo v možnosti pokrýt celou šířku závodní dráhy i z nižšího umístění než v případě snímače IMX219. Další výhodou snímače IMX477 byla větší odolnost vůči změnám osvětlení závodní dráhy, což umožňovalo dosahovat lepších výsledků její detekce.

Bohužel se však snímač IMX477 ukázal být překážkou ve zvládání detekce zatáček, ve kterých detekční algoritmus velmi často neposkytoval žádnou detekovanou čáru, případně ji detekoval jen velmi sporadicky. Řídící algoritmus autíčka tak při průjezdu zatáčkou nedostával informaci o svojí aktuální pozici vůči čarám a nemohl tak provádět potřebné korekce směru jízdy a často se tak autíčko dostávalo mimo dráhu. Toto bylo možné zlepšit zvýšením pozice kamery, ovšem výsledkem tohoto bylo, že snímač byl umístěný zhruba v poloviční výšce oproti autíčku sestaveném panem Ing. Steingartem a stále nebylo dosahováno dostatečně jisté detekce čar. Jelikož snížení výšky stožáru bylo jedním z cílů této práce, tak bylo nutné najít jiné řešení, které za prvé umožní dostatečně snížit stožár pro umístění snímače a za druhé bude poskytovat spolehlivější informace o pozici na dráze.

Řešení bylo nalezeno v podobě snímače IMX219 disponujícího širokoúhlým objektivem (udávaný úhel záběru je 200°), který oproti původně používanému snímači s úhlem záběru 120°, umožňuje i z výšky 8 cm nad úrovní vrchního dílu autíčka pokrýt celou šířku dráhy. Ve spolupráci s drobně upraveným detekčním algoritmem pro tento snímač bylo možné dosáhnout v podstatě kontinuální detekce obou čar při jízdě po rovných úsecích tratě a detekce alespoň jedné čáry při průjezdu zatáčkou. Spojení snímače s širokoúhlým objektivem společně s pro něj vyladěným detekčním algoritmem tedy položilo spolehlivý základ, na němž bylo následně možné rozvíjet řídící algoritmus autíčka.

Tato podkapitola je rozdělena do tří částí, první část (5.3) rozebírá zvolený programovací jazyk a knihovny, druhá část (5.3) se zabývá samotnou detekcí čar vymezujících závodní dráhu, třetí část (5.3) rozebírá detekci značek označujících začátek a konec úseku s omezenou rychlostí.

Programovací jazyk a knihovny

Jak již bylo nastíněno v průběhu podkapitoly 5.2, tak jako programovací jazyk pro implementaci řídícího systému autíčka byl zvolen Python, konkrétně ve verzi 3.6. Tento jazyk byl zvolen se záměrem co nejvíce urychlit vývoj a prototypování detekčního systému vymezujících čar s pomocí dostupných knihoven. V počátcích vývoje bylo navíc stále možné, že pro některý řešený problém bude využito algoritmů strojového učení, případně neuronové sítě, přičemž pro vývoj v těchto oblastech je jazyk Python vhodný. Verze 3.6 pak byla zvolena z důvodu své dostupnosti na platformě Jetson Nano, kde představuje nejnovější podporovanou verzi.

Jelikož se jazyk Python řadí mezi tzv. vysokoúrovňové jazyky, které typicky nepředstavují nejlepší řešení pro výkonově efektivní úlohy, přichází, vzhledem na poměr výkonu platformy Jetson Nano a náročnosti vzniklé implementace, na řadu otázka, týkající se výkonosti výsledného řešení. Tato otázka musela být pro dosažení použitelného výsledku vyřešena a způsob jejího vyřešení bude postupně rozebrán dále v textu společně se samotnou implementací.

Pro samotnou detekci čar a značek jsou využívány knihovny OpenCV[8] a NumPy[9], jejichž prostřednictvím jsou zpracovávána obrazová data, získávaná ze snímače IMX219. Snímky jsou ze snímače získávány za pomoci knihovny NanoCamera[10], jež poskytuje jednoduché rozhraní pro práci s kamerami s CSI i USB rozhraním. Také je tato knihovna přímo připravena pro svoje nasazení společně s knihovnou OpenCV a tak je například možné zís-

[8] https://pypi.org/project/opencv-python/
[9] https://numpy.org/
[10] https://github.com/thehapyone/NanoCamera

kaný snímek z kamery rovnou zpracovávat prostřednictvím funkcí knihovny OpenCV díky využití stejného datového typu.

Přestože již byly některé části implementace, zejména týkající se ovládání motorů představeny v podkapitole 5.2 pro poskytnutí představy o objevených komplikacích, tak ve zbytku textu je výsledný řídící program rozebírán po částech odpovídající dané problematice. Vzhledem k podobě programu, kdy jeden program v sobě spojuje veškeré funkce pro detekci a řízení by bylo možné tento program rozebrat jako jeden velký celek, jelikož jednotlivé části na sebe bezprostředně navazují, případně jsou přímo propojeny, avšak pro snadnější pochopení celkové funkce bude program rozebírán po iteracích, ve kterých byl vyvíjen, kdy každá iterace představuje jednu funkci z výsledné funkcionality. První takovou iterací je následující část 5.3, zabývající se detekcí vymezujících čar, přičemž následná sekce 5.3, rozebírající detekci značek, je v konečném programu přímo zakomponována do algoritmu pro detekci čar, jelikož využívá některé z jeho mezivýsledků pro vlastní zpracování. Obdobně poté i následující kapitola 6, věnující se systému pro detekci překážek, postupně doplňují a rozvíjejí části představené v této kapitole.

Detekce vymezujících čar

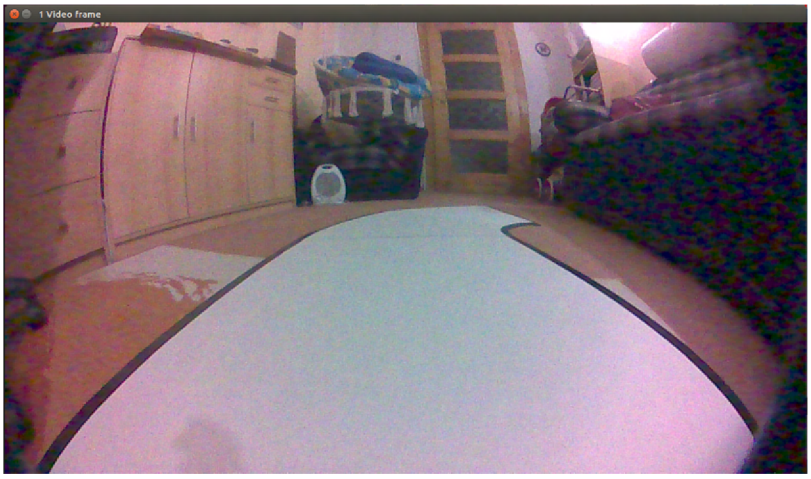
Část programu, zajišťující detekci čar v pořizovaných snímcích, čerpá ze dvou tutoriálů [38, 37], z nichž byl převzat postup společně s funkcemi pro zpracování vstupních snímků za účelem detekce vymezujících čar. Některé funkce byly převzaty v původní, některé v pozměněné podobě a následně jako celek upraveny, nebo doplněny pro požití v této práci. Každá taková funkce je ve zdrojovém souboru viditelně označena.

Pro detekci čar vymezujících závodní dráhu je využíván princip Houghovy transformace rozebraného v podkapitole 4.2, který je využívám prostřednictvím funkcí knihovny OpenCV. Algoritmus pro detekci se primárně skládá z hlavní nekonečné smyčky programu, zajišťující kontinuální získávání snímků z obrazového snímače, jejich zpracovávání a posléze využití výsledků pro řízení směru a rychlosti autíčka.

```
#------------------------------------
# camera initialization
#------------------------------------

#IMX 219
camera = nano.Camera(flip = 0, width = 1280, height = 720, fps = 120)

#------------------------------------
# MAIN loop
#------------------------------------
while camera.isReady():
  try:
    #read camera feed
    frame = camera.read()

    #lowering input resolution
    resized_frame = frameResize(frame)

    #noise removal with gaussian blur
    gauss_frame = gauss(resized_frame)

    #convert to gray image
    gray_frame= cv2.cvtColor(gauss_frame, cv2.COLOR_RGB2GRAY)

```

```
25      #canny edges
26      canny_edges_frame = canny(gray_frame)
27
28      #isolate region of interest
29      isolated_frame_region = region(canny_edges_frame)
30
31      #detect line segments in isolated region
32      line_segments = detectLineSegments(isolated_frame_region)
33
34      #calculate left and right line from gathered segments
35      final_lines = averageSlopeIntercept(resized_frame, line_segments)
36
37    except KeyboardInterrupt:
38      break
```

Na ukázce kódu výše dochází na začátku k inicializaci kamery s parametry s požadovaným rozlišením 1280 × 720 při 120 snímcích za sekundu. Snímač IMX219 sice ve své specifikaci umožňuje i jiné snímkové frekvence při požadovaném rozlišení, ovšem tyto nejsou na platformě Jetson Nano, na rozdíl od platformy Raspberry Pi, snadno dostupné. V následné hlavní smyčce dochází prostřednictvím funkce `read` k získání snímku z kamery (obrázek 5.4), jehož postupné zpracování bude rozebráno z pohledu jednotlivých funkcí, které jej zpracovávají.



Obrázek 5.4: Snímek získaný ze snímače IMX219 v rozlišení 1280 × 720, v rozích jsou patrná slepá místa, která jsou způsobena použitým širokoúhlým objektivem (galerie autora).

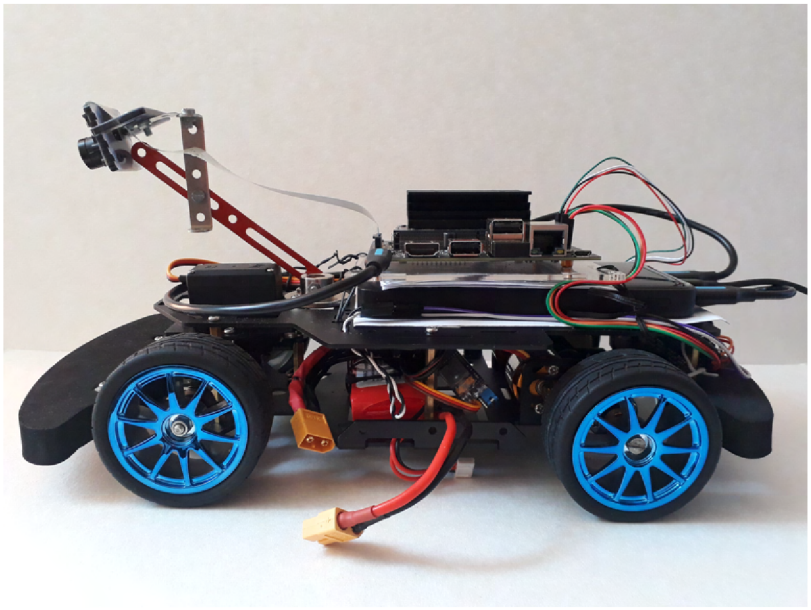
`frameResize`

Rozlišení vstupního snímku je prostřednictvím funkce `frameResize` sníženo s cílem zvýšit množství snímků zpracovaných za 1 s. V průběhu vývoje bylo zjištěno, že při využívání plného vstupního rozlišení poskytuje implementovaný algoritmus snímkové frekvence v jednotkách snímků za sekundu. Zejména to bylo patrné ve fázi vývoje, kdy byl snímač IMX219 nahrazen snímačem IMX477, se kterým bylo využíváno ještě vyšší rozlišení 1920 × 1080.

Například v případě, kdy je zpracováno 5 snímků za vteřinu, je získána informace o aktuální pozici autíčka na dráze pouze každých 200 ms. Za tuto dobu je autíčko schopno urazit nezanedbatelnou vzdálenost a v nejhorším případě se dostat i do takové pozice vůči vymezujícím čarám, kdy po zpracování dalšího snímku a zjištění pozice autíčka vůči hra-

ničním čarám nebude možné provézt včasnou korekci směru a autíčko vyjede z dráhy. Pro získání častější aktualizace pozice autíčka v rámci závodní dráhy bylo nutné dosáhnout vyšší snímkovací frekvence při zpracování. Zpracovávání vysokého rozlišení také spotřebovávalo naprostou většinu dostupného procesorového výkonu platformy Jetson Nano a jelikož v této fázi vývoje ještě nebyla implementována detekce značek, řídící algoritmus a detekce překážek, tak bylo nutné snížit výkonovou náročnost detekce čar tak, aby pro tyto činnosti zbyl dostatek výkonu.

Za tímto účelem byla provedena řada experimentů. Jedním ze zvažovaných způsobů, jak zrychlit zpracování snímků, bylo provézt ořezání vstupního snímku tak, aby byla zpracovávána pouze zájmová část, případně alespoň vypustit většinu nepotřebných obrazových dat. Obrazový snímač ze svého umístění na vozítku (obrázek 5.5) totiž zabírá nejen samotnou závodní dráhu, ale i její okolí, jak je možné vidět na obrázku 5.4, které však pro detekci hraničních čar není nutné analyzovat každou funkcí, kterou získaný snímek postupně prochází, než jsou v něm detekovány hraniční čáry.



Obrázek 5.5: Boční pohled na autíčko a umístění kamery sledující dráhu (galerie autora).

Tyto kroky ovšem vedly pouze k zanedbatelnému, či velmi nízkému zvýšení snímkové frekvence. Nejvyšších výkonových nárůstů bylo dosaženo prostřednictvím snížení rozlišení vstupního snímku, při kterém je celá zachycená scéna transformována na snímek o nižším rozlišení, který je následně dále zpracováván stejným způsobem, jako byl v dřívějších fázích vývoje zpracováván surový snímek přímo ze snímače.

Tato změna se udála ještě předtím, než byl místo snímače IMX477 opět použit snímač IMX219, tentokrát se širokoúhlým objektivem. Jelikož v případě snímače IMX477 je výchozí rozlišení 1920×1080 tak testovaná zmenšená rozlišení vycházejí z něj a jelikož bylo nutné detekční algoritmus pro změněné rozlišení odladit, tak po výměně senzoru za IMX219 bylo zachováno snížené rozlišení vycházející ze snímače IMX477. Pro první kolo testování bylo rozlišení 1920×1080 zmenšeno $4\times$ na rozlišení 480×270 čímž bylo dosaženo zvýšení snímkové frekvence z původních 5 snímků za sekundu na 15. V tabulce 5.4 jsou uvedené průměrné snímkové frekvence pro jednotlivá zmenšená rozlišení a to jak pro snímač IMX477 tak i pro snímač IMX219.

Zajímavého zrychlení bylo dosaženo i v případě pouhé výměny snímače IMX477 za snímač IMX219, kdy v případě výchozího rozlišení 1920×1080 snímače IMX477 bylo dosahováno průměrně již zmíněných 5 snímků za sekundu, zatímco v případě výchozího rozlišení 1280×720 snímače IMX219 bylo dosaženo v průměru 11 snímků za sekundu. Na základě testování schopností detekčního algoritmu, se sníženými vstupními rozlišeními, bylo zjištěno, že jako nejlepší kompromis mezi dosaženou snímkovou frekvencí a spolehlivostí detekce hraničních čar je rozlišení 240×135, při kterém je při použití snímače IMX219 dosahováno průměrné snímkové frekvence 25 snímků za sekundu. Při nižších rozlišeních již algoritmus velmi často nebyl schopen detekovat žádnou čáru a ztrácela se tak informace o pozici autíčka na dráze.

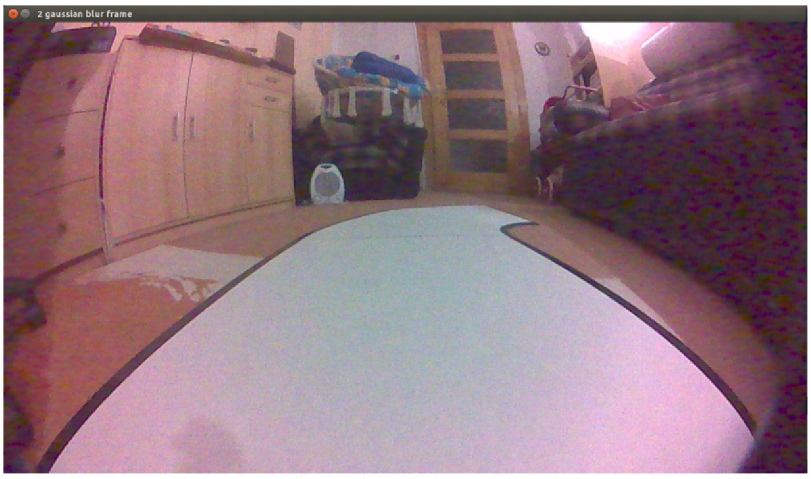
dělící koeficient	$\frac{1920 \times 1080}{\text{dělící koeficient}}$	IMX477 (1920×1080) [snímků/s]	IMX219 (1280×720) [snímků/s]
1	výchozí rozlišení snímače	5	11
4	480×270	15	21
6	320×180	19	23
8	240×135	22	25
10	192×108	25	27

Tabulka 5.4: Dosažené snímkové frekvence se snímači IMX477 a IMX219 v závislosti na sníženém rozlišení.

`gauss`

Funkce `gauss`, na ukázce níže, využívá funkci `GaussianBlur` z knihovny OpenCV, jejímž prostřednictvím je prováděno Gaussovské rozostření vstupního snímku pro odstranění šumu.

```
1 def gauss(frame):
2   kernel_size = (5, 5)
3   gauss_frame = cv2.GaussianBlur(frame, kernel_size, 0)
4   return gauss_frame
```



Obrázek 5.6: Snímek po zpracování funkcí `gauss` (galerie autora).

Tímto krokem je do určité míry ztracena ostrost snímku, což však není pro další zpracování překážkou. Jako nejvhodnější velikost kernelu, byla stanovena matice o velikosti 5×5 pixelů, se kterou bylo dosahováno nejlepších výsledků detekce čar. Výstupem této funkce je snímek na obrázku 5.6 (pozn. pro účely dokumentace jsou veškeré snímky zobrazující výstupy jednotlivých funkcí v tomto textu v rozlišení 1280×720, zatímco v samotném programu jsou již od průchodu funkcí `frameResize` zpracovávány v rozlišení 240×135)

`cvtColor`

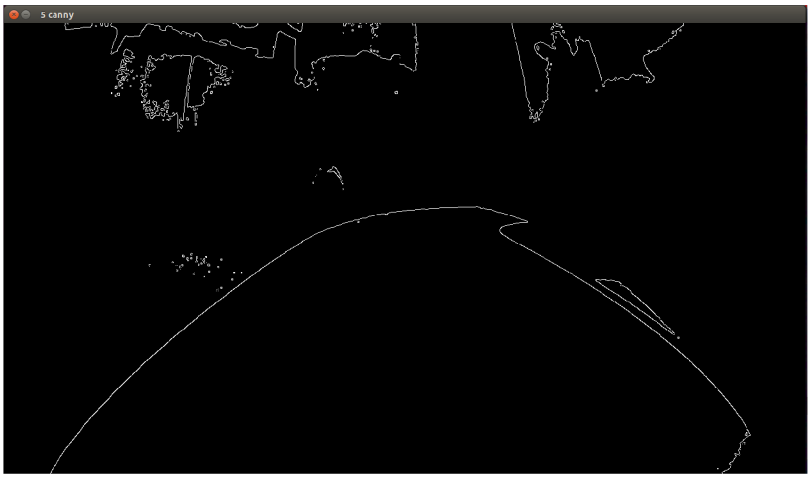
Jedná se o jednu z funkcí knihovny OpenCV, která na základě zadaného parametru převádí vstupní snímek do cílového barevného spektra. V tomto případě je vstupní snímek v RGB spektru převeden do snímku v odstínech šedé, kdy každý pixel je převeden na hodnotu v rozmezí 0-255, kde číslo 0 odpovídá černé barvě a číslo 255 odpovídá bílé. Výstup této funkce je zachycen na obrázku 5.7.

Obrázek 5.7: Výstupní snímek z funkce `cvtColor` (galerie autora).

`canny`

V rámci funkce `canny` je využívána funkce Canny z knihovny OpenCV, která ve vstupním snímku detekuje hrany, představované přechody mezi bílou a černou barvou. V této funkci je pro správné fungování nutné nastavit vhodně hodnoty parametrů *lowThreshold* a *highThreshold* v tomto případě nastavené na hodnotu 50 a 150. Často používané hodnoty, ve spojení s touto funkcí, jsou také hodnoty 200 a 400, ovšem v takovém případě již docházelo k tomu, že hrana představující hranu dráhy nebyla detekována a při testování bylo s hodnotami 50 a 150 dosahováno nejlepších výsledků. Detekované hrany jsou znázorněny na obrázku 5.8

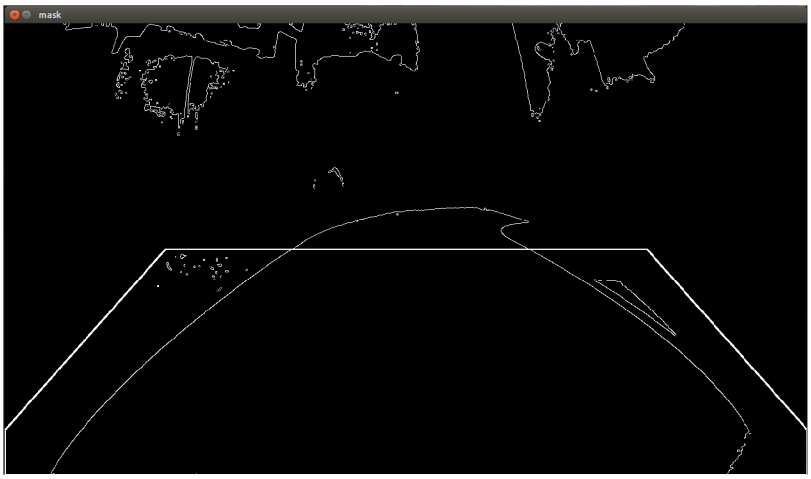
```
1 def canny(frame):
2   low_threshold = 50
3   high_threshold = 150
4   canny_frame = cv2.Canny(frame, low_threshold, high_threshold)
5   return canny_frame
```



Obrázek 5.8: Výstup funkce `canny` zobrazující detekované hrany (galerie autora).

`region`

Tato funkce aplikuje na zpracovávaný snímek masku znázorněnou na obrázku 5.9, kdy po její aplikaci zbude na výstupním snímku převážně samotná závodní dráha s jejím minimálním okolím, což je možné vidět na obrázku 5.10. Účelem tohoto kroku je odstranění většiny hran, které nevyznačují závodní dráhu a zasahovaly by v dalších krocích do výpočtů čímž by přispěly k nepřesným výpočtům hraničních čar dráhy. Tvar a velikost zájmové oblasti, která je ve výstupním snímku ponechána byla stanovena s cílem zajistit co největší rezervu okolo dráhy tak, aby v případě, kdy se autíčko nenachází na středu dráhy, bylo stále možné zachytit na rovných úsecích obě hraniční čáry a nedošlo tak kvůli příliš striktnímu ořezání snímku ke ztrátě cenných dat. Na druhou stranu však byla maska ponechána co nejmenší, aby do následného zpracování zasahovalo co nejméně nepotřebných hran.



Obrázek 5.9: Maska použitá ve funkci `region` pro ořezání vsuptního snímku (galerie autora).

Obrázek 5.10: Výstupní snímek funkce `region` zachycující pouze oblast závodní dráhy (galerie autora).

`detectLineSegments`

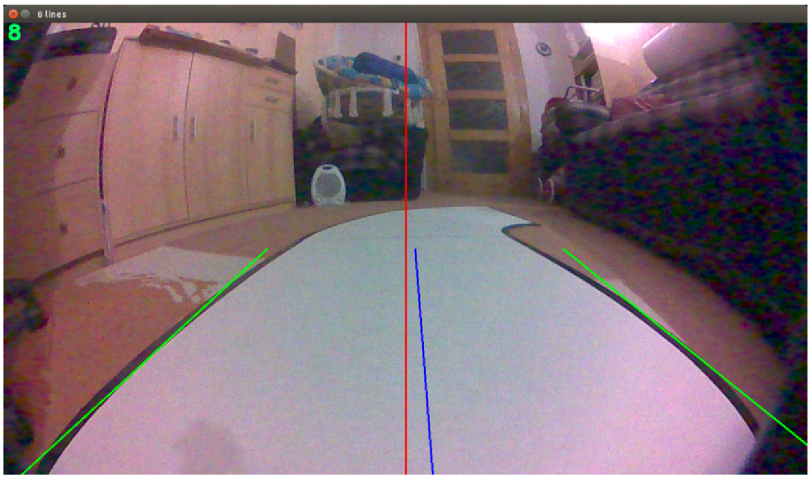
Jak je možné odvodit z názvu této funkce, tak je zodpovědná za detekci segmentů čar v obdrženém snímku. Její základ tvoří funkce `HoughLinesP` z knihovny OpenCV, přičemž je volána s parametry odladěnými pro detekci zejména krajních čar dráhy. S ohledem na detekci těchto čar byly například stanoveny parametry jako *minLineLenght*, určující jakou minimální délku musí mít detekovaná čára a *maxLineGap*, udávající maximání rozestup mezi na sebe navazujícími čarami. Také parametry jako požadovaná přesnost určení vzdálenosti a úhlu jsou důležitou součástí, do značné míry ovlivňující úspěšnost detekce čar. Například při zvýšení parametru *rho* dojde k detekování mnohem většího množství čar a také ke zvýšení úspěšnosti detekce čar při průjezdu zatáčkou, ovšem dojde ke snížení přesnosti detekce pozice čar a mezi jednotlivými snímky bude často jedna a ta samá detekovaná čára na pozici lišící se i o desítky pixelů. V případě snížení tohoto parametru dojde ke zvýšení přesnosti a pozice čáry je mezi jednotlivými snímky mnohem konzistentnější, ovšem při hodnotách *rho* pod 10 již byla detekce čar při průjezdu zatáčkou značně nespo-

lehlivá a často nedocházelo k detekci žádné hraniční čáry a tím ke ztrátě informace o pozici. Parametr *angle* se projevoval výrazně na přesnosti detekce čáry v zatáčkách, ovšem při zvyšování této přesnosti docházelo k velmi razantním výkonovým poklesům a tím pádem je tento parametr kompromisem mezi přesností detekce čar a snímkovou frekvencí. Také parametr *min_threshold*, udávající minimální počet hlasů, který musí detekovaná čára mít velmi ovlivňoval množství detekovaných čar, snížení parametru vedlo na spolehlivější detekci v zatáčkách, ovšem mnohdy velmi nepřesnou, zatímco vyšší hodnoty parametru snižují množství detekovaných čar a zvyšují přesnost jejich detekce. Výsledný snímek této funkce je na obrázku 5.11

```
def detectLineSegments(inputImage):
  rho = 10 #distance precision in pixel units
  angle = np.pi/180 #angular precision in radian, i.e. 1 degree
  min_threshold = 75 # minimal number of votes
  minLineLength = 10 # minimal line length
  maxLineGap = 1 # maximal gap between lines
  line_segments = cv2.HoughLinesP(inputImage, rho, angle, min_threshold,
    np.array([]), minLineLength, maxLineGap)

  return line_segments
```

Obrázek 5.11: Detekované segmenty čar, zobrazené nad vstupním snímkem, prostřednictvím funkce `detectLineSegments`. V pravé části obrázku je možné vidět kousek bílého koberce, na kterém byly také detekovány segmenty čar (galerie autora).

`averageSlopeIntercept`



Obrázek 5.12: Výsledné detekované čáry za pomoci funkce `averageSlopeIntercept`. Levá a pravá zelená čára představují detekované vymezující čáry. Modrá čára uprostřed představuje průměr těchto čar a tím pádem pozici, jakou by mělo autíčko na dráze mít aby bylo přesně v jejím středu. Červená čára přestavuje přesný střed obrazu a tedy aktuální pozici autíčka na dráze. Z této situace je patrné, že pro vycentrování autíčka na dráze musí dojít k tomu aby se modrá a červená čára překrývaly a tím pádem tedy autíčko musí zatočit mírně vpravo směrem k modré čáře (galerie autora).

Výstupem této funkce jsou již detekované hraniční čáry, kterých může být 0-2. Tato funkce získává na vstupu detekované segmenty čar, které následně spojuje do dvou krajních čar. Z tohoto důvodu dojde v tomto případě k započítání i segmentů čar odpovídajících části bílého koberce, které je možné vidět na obrázku 5.11. Tímto dojde k vytvoření určité odchylky pozice pravé čáry oproti její skutečné pozici, jelikož však dochází k průměrování detekovaných segmentů, tak je vliv takovýchto falešných segmentů na výslednou detekovanou čáru minimální a ze získaného výsledku je možné přesně řídit směr jízdy autíčka. Výstup této funkce je zachycen na obrázku 5.12.

Detekce značek

V pravidlech soutěže NXP Cup jsou specifikovány dvě značky (obrázek 2.9), které musí umět autíčko rozeznávat a reagovat na jejich výskyt. Značka skládající se ze 4 černých pruhů znamená začátek úseku s omezenou rychlostí, kde autíčko musí zpomalit zhruba na poloviční rychlost, zatímco značka složená ze 3 černých pruhů značí konec úseku s omezenou rychlostí.

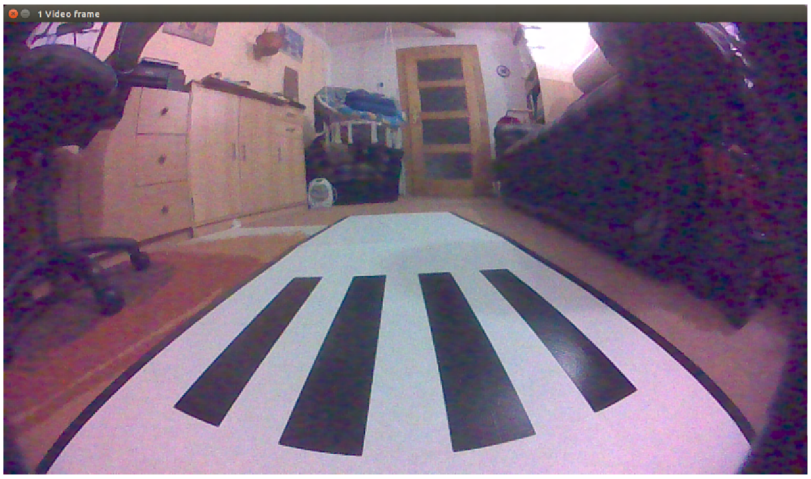
V průběhu vývoje jejich detekce se ze začátku ukázalo být komplikací to, že jsou obě značky podobné a liší se pouze počtem pruhů. Prvotní pokusy využívaly funkcí knihovny OpenCV pro detekci obrysů, kdy při vhodně zvolených parametrech bylo ve vstupním snímku nalezeno více objektů, které se však od sebe daly rozlišit svojí plochou. Při vhodně zvolené minimální velikosti plochy detekovaného obrysu bylo možné odfiltrovat veškeré nežádoucí obrysy z okolí dráhy a poté podle počtu zbylých obrysů rozlišit, zda se v záběru nacházejí 3 nebo 4 pruhy. Problémem tohoto řešení však bylo, že ve spojení se snímačem IMX219 docházelo často k tomu, že ve snímku byly ze 4 pruhů identifikovány pouze 3 a

značka tak byla často nesprávně identifikována a tedy autíčko vykazovalo nesprávné chování. Nejčastěji k tomuto docházelo ve chvíli, kdy autíčko nejelo přes díl se značkou přesně uprostřed dráhy, ale nacházelo se blíže jednomu, nebo druhému okraji dráhy a tedy z pohledu kamery nejvzdálenější značka splývala s černou čarou vymezující okraj dráhy a nebyla správně identifikována.

Řešení bylo nalezeno ve zcela přepracovaném způsobu detekce značek, který využívá mnohem jednoduššího principu fungování a vedl na zjednodušení detekce značek bez využívání detekce obrysů, která se s určitými hodnotami parametrů ukázala být jako značně výkonově náročnou a snižovala výkonnost celého systému. Nový způsob rozpoznání značek využívá prahování snímku v odstínech šedi (pomocí funkce `inRange`), které bylo následně zpětně testováno i pro detekci hraničních čar dráhy, kde díky němu docházelo zejména v nočních hodinách s horším osvětlením ke zpřesnění detekce hraničních čar mezi jednotlivými snímky, avšak ve všech ostatních situacích tento krok detekci čar naopak zhoršoval a tedy nakonec nebyl při detekci čar použit. V případě detekce značek na dráze je však tento krok základem pro celý proces detekce značek.

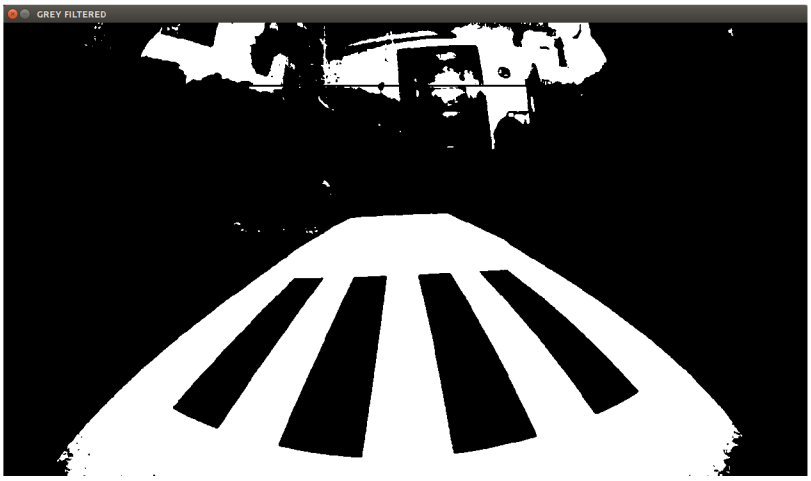
`inRange`

Funkce `inRange` je součástí knihovny OpenCV a slouží k prahování. V toto konkrétním případě, kdy každý pixel ve vstupním snímku má hodnotu v rozmezí 0-255 je možné snímek v odstínech šedi převézt na snímek obsahující pouze bílou nebo černou barvu nastavením vhodné prahové hodnoty. Na základě testování bylo zjištěno, že nejlepších výsledků je dosahováno s prahovou hodnotou 150, kdy všechny pixely s nižší hodnotou budou mít nově hodnotu 0 a tedy černou barvu, zatímco pixely s vyšší hodnotou budou mít hodnotu 255 odpovídající bílé barvě. Tímto je dosaženo toho, že ve výsledném snímku bude bílou barvou zachycena pouze závodní dráha a v okolí, které není pro další zpracování potřeba, bude naprostá většina pixelů černých, čímž dojde ke zjednodušení scény, ve které zůstane pouze bílý povrch závodní dráhy, na kterém se nacházejí černé značky.



Obrázek 5.13: Vstupní snímek zachycující značku označující začátek úseku s omezenou rychlostí (galerie autora).

Na obrázku 5.13 je pohled z kamery autíčka na dráhu na které se nachází značka označující začátek úseku s omezenou rychlostí. Tento snímek je zpracováván již pro účely detekce hraničních čar a tedy projde procesem snížení rozlišení, Gausovským rozostřením a je převeden do odstínů šedi. Snímek v odstínech šedi, který v části zabývající se detekcí čar pokračuje jiným procesem, je v tuto chvíli zkopírován pro účely detekce značek a na kopii snímku je následně provedeno prahování funkcí `inRange`, jehož výsledek se nachází na obrázku 5.14.



Obrázek 5.14: Snímek se značkou úseku s omezenou rychlostí po svém zpracování a převedení na černo-bílý snímek (galerie autora).

Snímek zpracovaný prahováním je následně analyzován velmi jednoduchým algoritmem pro detekci značek. Tento algoritmus využívá toho, že ve snímku jenž prošel procesem prahování se v jeho spodní části nachází pouze černá barva mimo samotného bílého povrchu dráhy, na které se však nacházejí černé oblasti odpovídající černým pruhům. V tomto případě je důležitým jevem to, že i v případě, kdy se autíčko nachází například úplně na pravém okraji dráhy, tak je mezi levým černým krajem dráhy a k němu nejbližšímu černému pruhu značky mezera, tvořená bílými pixely představujícími povrch dráhy.

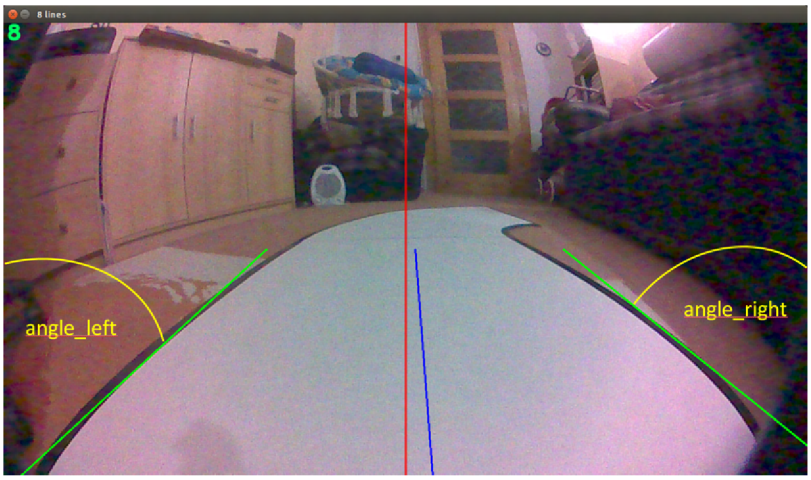
Následná analýza, na ukázce kódu níže, poté ve snímku zjistí, kolik je na vybraném řádku (v tomto případě byl zvolen 100. řádek ve snímku se sníženým rozlišením) přechodů mezi černou a bílou barvou. V případě, že kamera snímá značku tvořenou 4 pruhy tak při čtení řádku zleva doprava bude na snímku zjištěno 10 přechodů černá-bílá. V případě značky tvořené 3 pruhy bude těchto přechodů 8. Pro další zpřesnění detekce značky je za detekovanou značku považován pouze případ, kdy daný počet přechodů se vyskytl v 5 po sobě jdoucích snímcích. Tento počet snímků vykazoval v testech nejlepší poměr chybovosti a rychlosti identifikace značky. Například v situaci, kdy se autíčko zrovna snaží dostat z kraje dráhy na její střed a jede šikmým směrem vůči značce, tak alespoň jeden snímek z kamery může na 100. řádku mít pouze 8 přechodů černá-bílá a tedy identifikovat pouze 3 ze 4 přítomných pruhů. Toto by poté v případě nepřítomnosti pojistky, vyžadující 5 po sobě jdoucích snímků s daným počtem přechodů černá-bílá, vedlo k chybné interpretaci značky.

```
1     was_black = False
2     was_white = False
3     black_white_transition = 0
4
5     for pixel in gray_filtered[100]:
```

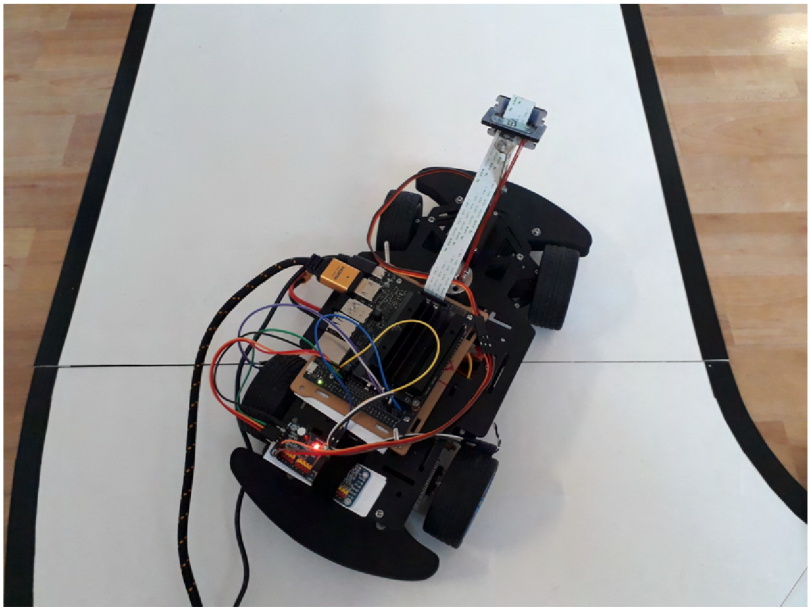
```
6       if was_black is False and was_white is False:
7         if pixel == 0:
8           was_black = True
9         elif pixel == 255:
10          was_white = True
11      elif was_black is True:
12        if pixel == 0:
13          was_black = True
14          was_white == False
15        elif pixel == 255:
16          was_black = False
17          was_white == True
18          black_white_transition += 1
19      elif was_white is True:
20        if pixel == 0:
21          was_black = True
22          was_white = False
23          black_white_transition += 1
24        elif pixel == 255:
25          was_black = False
26          was_white = True
27    print("black_white_transition: ", black_white_transition)
```

5.4 Řízení pohybu

Algoritmus řídící pohyb autíčka po dráze se snaží stále držet autíčko uprostřed dráhy. V prvotní fázi svého vývoje byl založen na výpočtu toho jaký úhel svírá levá krajní čára čára s levým okrajem snímku a pravá krajní čára s pravým okrajem obrazu, což je znázorněno na obrázku 5.15. Následně byl pravý úhel odečten od levého a výsledek byl z úhlu převeden na číselný rozsah 0-115, což je rozsah pohybu předního servomotoru na každou stranu z výchozí pozice, jejíž hodnota je 305. Tento způsob natáčení přední nápravy fungoval správně v situacích, kdy se autíčko nacházelo blíže ke středu dráhy, než k jejímu okraji, a také v situaci, kdy směřovalo šikmo na jeden z okrajů závodní dráhy, jako je možné vidět na obrázku 5.16.

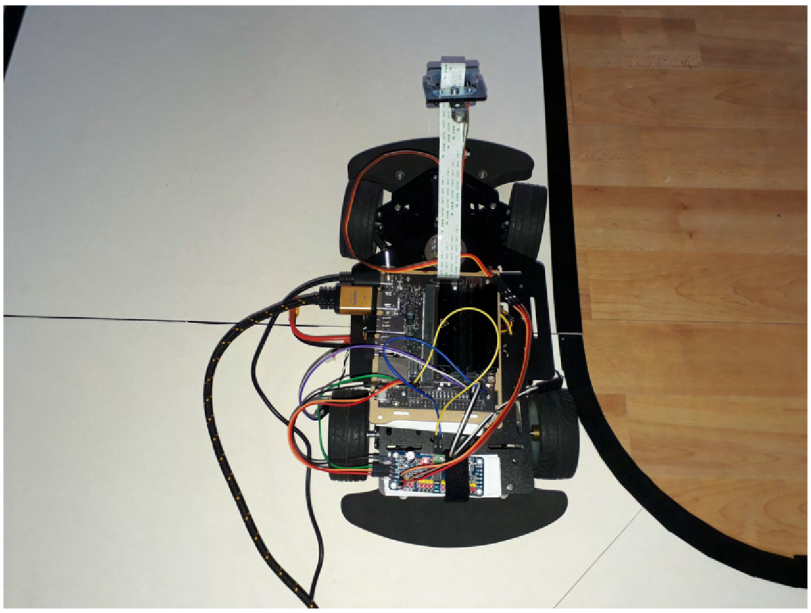


Obrázek 5.15: Počítané úhly *angle_left* a *angle_right*, jejichž rozdíl určoval natočení přední nápravy. Červená čára vyznačuje střed obrazu a tedy pozici autíčka. Modrá čára je průměr detekované levé a pravé čáry, jenž ve spojení s červenou čarou vizuálně informuje kam by mělo autíčko směřovat.



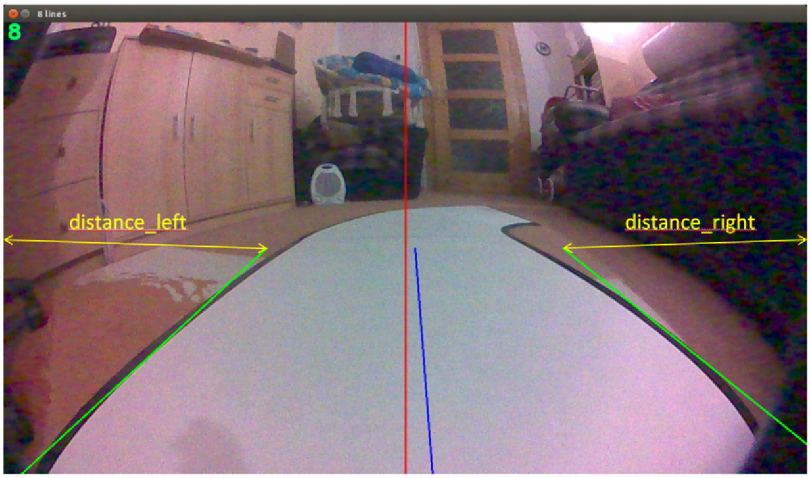
Obrázek 5.16: Správné natočení zadní nápravy při řízení na základě rozdílu úhlu hraničních čar (galerie autora).

V případě, kdy se autíčko pohybovalo rovnoběžně s krajem dráhy avšak přesně po jejím kraji, tak zmíněný přepočet udával hodnoty, které způsobovaly zatáčení autíčka ven z dráhy namísto do jejího středu, což je zachyceno na obrázku 5.17. Toto bylo dáno tím, že v situaci kdy se vozítko nacházelo u jednoho okraje, tak čára značící vzdálenější okraj dráhy svírá s okrajem snímku větší úhel než čára, která je bližší a přepočet tak funguje opačně a jeho hodnotu by bylo nutné nějakým způsobem převádět na opačnou.



Obrázek 5.17: Nesprávné natočení přední nápravy směrem ven z dráhy při jízdě po jejím okraji (galerie autora).

K vyřešení tohoto problému bylo nutné výpočet pozice mezi čarami přepracovat tak, aby za každé situace byla přední náprava natočena správným směrem. Tohoto bylo dosaženo tím, že byla opuštěna myšlenka počítat úhel který detekované čáry svírají s příslušným krajem snímku. Namísto toho byla spočítána vzdálenost od autíčka nejvzdálenějšího bodu detekované čáry vůči bodu nacházejícímu se na levé, respektive pravé straně snímku přesně v polovině výšky snímku, tak jako je znázorněno na obrázku 5.18.



Obrázek 5.18: Vzdálenosti *distance_left* a *distance_right* počítané v řídícím programu.

Na ukázce kódu níže je znázorněn výpočet vzdálenosti levé a pravé čáry od okraje (jedná se o Euklidovskou vzdálenost). Další část kódu poté v případě, že byla detekována levá i pravá čára v daném snímku spočítá rozdíl levé a pravé vzdálenosti. Rozdíl vzdálenosti je dále přepočítán na úhel natočení předního servomotoru. Pokud je vzdálenost levé čáry od okraje větší než vzdálenost pravé čáry od okraje, což znamená, že autíčko se nachází blíže

levému okraji, tak výsledkem je kladná hodnota, která po svém přičtení k výchozí hodnotě udávající neutrální pozici servomotoru způsobí, že je náprava natočena úměrně vzhledem ke vzdálenosti doprava, aby se autíčko dostalo na střed dráhy. V případě opačné situace dojde k odečtení rozdílu vzdáleností a servomotor natočí přední nápravu doleva a dojde k vyrovnání pozice autíčka na střed dráhy. Pro přepočet rozdílu vzdáleností na hodnotu udávající natočení servomotoru je používána konstanta *multiplying_constant*. V případě, že by přepočítaná vzdálenost přesáhla rozsah servomotoru, tak je tato situace ošetřena nastavením maximálního možného natočení servomotoru v daném směru tak, aby nedošlo k jeho přetočení.

```
distance_left = 0
distance_right = 0
lane_distance_diff = 0

if left_line:
  distance_left = math.sqrt(((left_line_x2 - (0)) ** 2) + ((left_line_y2
    - height/2) ** 2))
if right_line:
  distance_right = math.sqrt(((right_line_x2 - (width)) ** 2) + ((
    right_line_y2 - height/2) ** 2))


if left_line and right_line:
  lane_distance_diff = distance_left - distance_right

  if (neutral_servo_angle + (lane_distance_diff * multiplying_constant)
    >= 190) and (neutral_servo_angle + (lane_distance_diff *
    multiplying_constant) <= 420):
    servo_angle =  neutral_servo_angle + (lane_distance_diff *
    multiplying_constant)
  elif ((neutral_servo_angle + lane_distance_diff * multiplying_constant)
     <= 190):
    servo_angle = 190
  elif ((neutral_servo_angle + lane_distance_diff * multiplying_constant)
     >= 420):
    servo_angle = 420
  prev_lane_distance_diff = lane_distance_diff
  prev_distance_left = distance_left
```

Obdobně je počítáno natočení přední nápravy i v situaci, kdy je detekována pouze jedna krajní čára, k čemuž dochází typicky při průjezdu zatáčkou, kdy je detekována pouze vnější čára značící dráhu. V takovém případě již nedochází k výpočtu rozdílu, ale natočení přední nápravy je určováno pouze vzdáleností detekované čáry od okraje. Při průjezdu zatáčkou je také aplikována jiná násobící konstanta pro přepočet na hodnotu natočení servomotoru. Na rovinkách je tato konstanta menší s cílem zajistit pozvolnější a plynulejší pohyb autíčka na střed dráhy. V případě použití větší konstanty docházelo k tomu, že autíčko se na rovných úsecích pohybovalo z jedné strany dráhy na druhou, než se po několika opakováních tohoto pohybu ustálilo na středu dráhy.

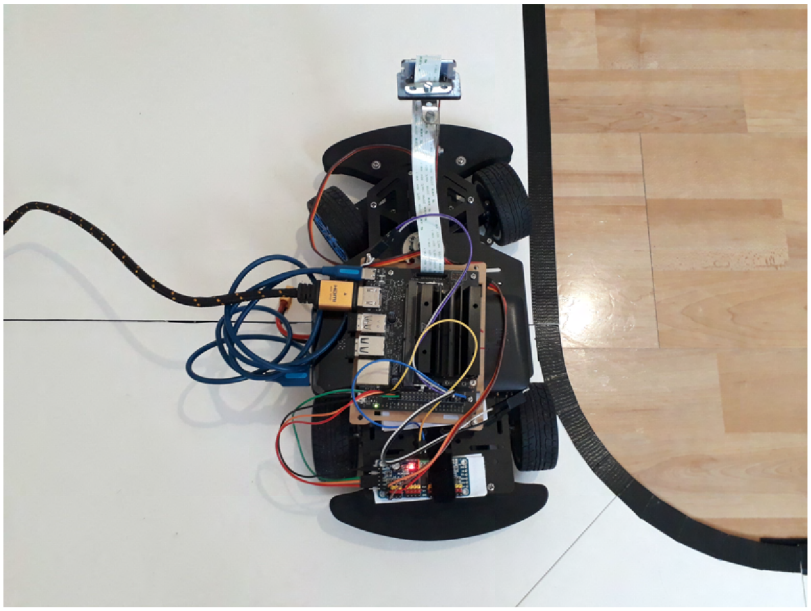
Na základě aktualizované hodnoty *servo_angle*, udávající natočení servomotoru, je provedena úprava PWM signálu řídícího natočení servomotoru, zachycená v útržku kódu níže na řádku 9. V této ukázce je na řádku 1-7 také zachycena část, která je zodpovědná za řízení rychlosti zadních elektromotorů. V případě, kdy autíčko projíždí prudkou zatáčkou, kdy je detekována pouze jedna z krajních čar, tak je vnitřní zadní motor zpomalen oproti vnějšímu motoru, jehož kolo musí objet delší dráhu tak, aby byla suplována funkce diferenciálu. V

případě malých korekcí směru toto není aplikováno. Na řádcích 11-16 je prováděna úprava rychlosti obou motorů na základě toho, zda byla detekována značka začátku, nebo konce úseku s omezenou rychlostí.

```
1  if left_line is True:
2    bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_2_END, int(i + 2))
3  if right_line is True:
4    bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_1_END, int(i + 2))
5  else:
6    bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_1_END, int(i + 1))
7    bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_2_END, int(i + 1))
8
9  bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_0_END, int(servo_angle))
10
11 if speed_sign is True:
12   bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_1_END, int(i + 1))
13   bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_2_END, int(i + 1))
14 if slow_sign is True:
15   bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_1_END, int(287))
16   bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_2_END, int(287))
```

Tento řídící systém je dále ještě rozšířen o části, které umožňují ovládání autíčka prostřednictvím bezdrátové klávesnice. Po spuštění algoritmu autíčko vyčkává na stisknutí klávesy **E**, po jejímž stisknutí dojde k aktivaci ESC regulátorů zadních elektromotorů společně s předním servomotorem. Dále je možné prostřednictvím klávesy **P** pohyb autíčka zastavit a stiskem klávesy **O** jej opět uvézt do pohybu. Jednou ze zajímavých možností je prostřednictvím klávesy **S** zastavení zadních elektromotorů, kdy přední servomotor zůstane aktivní a na základě detekovaných čar dále natáčí přední nápravu tak, aby udržel autíčko na dráze. Stiskem klávesy **W** poté dojde k opětovnému spuštění zadních elektromotorů. Tato možnost byla implementována pro účely vývoje, kdy možnost zastavení zadních motorů a následné ruční tlačení autíčka po dráze, které však stále pohybovalo přední nápravou, umožňovalo mnohem snadnější opakované průjezdy stejným úsekem při analyzování chování autíčka pro ladění řídícího algoritmu.

Takto realizovaný řídící algoritmus je již schopen si správně poradit se situací, kdy se autíčko nachází blízko jednoho z okrajů závodní dráhy (obrázek 5.19), se kterým si původně navržený systém řízení přední nápravy nedokázal poradit (viz obrázek 5.17).



Obrázek 5.19: Správné natočení přední nápravy, při jejím natáčení na základě rozdílu vzdáleností detekovaných čar (galerie autora).

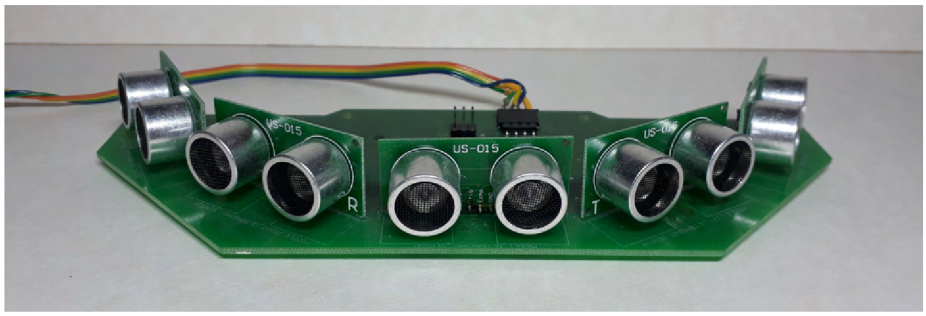
Kapitola 6

Detekce překážek

Tato kapitola se věnuje návrhu a realizaci systému pro detekci překážek na závodní dráze a vychází z poznatků představených v podkapitole 3.3. Tato kapitola je rozdělena na dvě části, první podkapitola s číslem 6.1 se zabývá návrhem detekčního systému na základě získaných poznatků. Podkapitola 6.2 rozebírá ve své první části implementaci navrženého senzoru a ve druhé části jeho zakomponování do již implementovaného systému, pro detekci závodní dráhy a navigaci po ní, představeného v kapitole 5.

6.1 Návrh systému

Z počátku návrh počítal s tím, že systém pro detekci překážek bude tvořen senzorovým polem vytvořeným z ultrazvukových senzorů a dojde ke vzniku obdobného modulu (obrázek 6.1), jaký pro účely detekce překážek realizoval Ing. Viktor Steingart [40] ve své původní realizaci robotického autíčka pro soutěž NXP Cup. Tento modul byl složen z 5 ultrazvukových senzorů US-015, přičemž úhel záběru jednoho senzoru je v rozmezí 15-35°.



Obrázek 6.1: Senzorové pole složené z ultrazvukových senzorů sestrojené Ing. Viktorem Steingartem (galerie autora).

V případě systému navrhovaného v této práci byly pro toto použití uvažovány senzory HC-SR04 představené v části 3.3. Proti použití těchto senzorů však hovořila nutnost použití konvertoru úrovně napětí, jelikož GPIO piny platformy Jetson Nano mají výstupní napětí 3,3 V, které je pro senzor HC-SR04 (tabulka 3.8, obrázek 3.5) nutné převézt na 5 V. Vhodným konvertorem je například více-kanálový BSS138 [5]. Nevýhodou tohoto řešení je vzhledem k množství komplikací, které byly objeveny v souvislosti s realizací systému pro

detekci čar, přidávání dalších modulů a součástek, jež potencionálně mohou přinést spoustu dalších komplikací a dále zdržet vývoj systému.

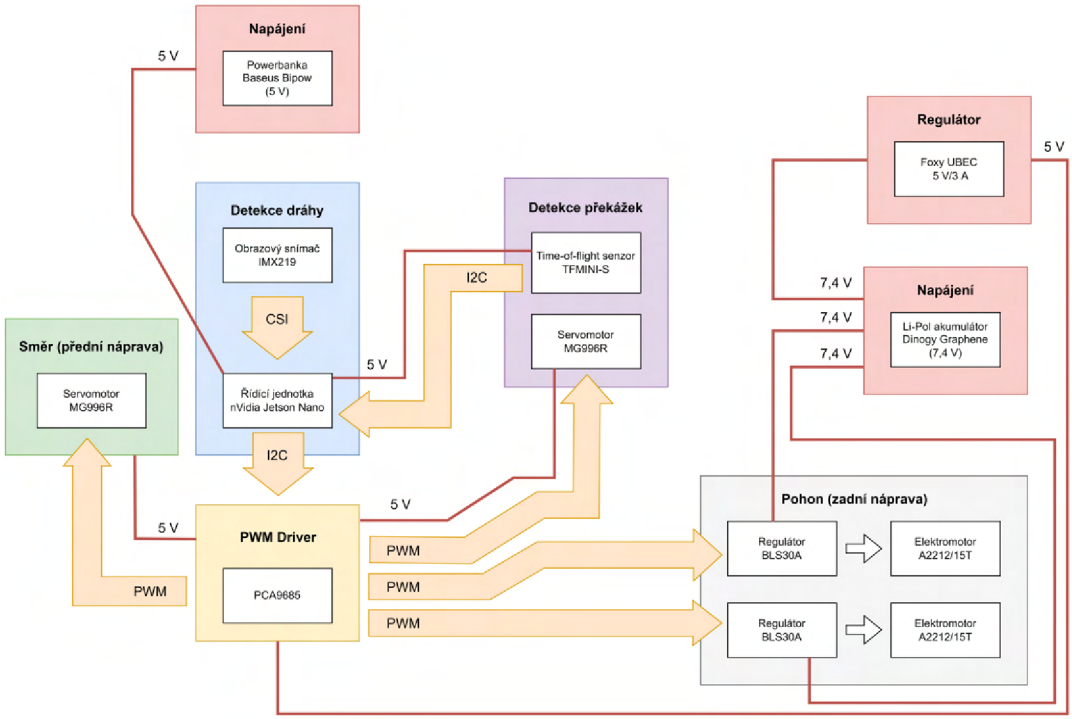
Pod vlivem těchto okolností společně s cílem této práce, jímž není pouze samotná realizace autonomního robotického vozítka, ale i vyzkoušení různých postupů a možností jak takové vozítko realizovat, bylo přehodnoceno použití ultrazvukových senzorů ve prospěch ToF senzoru (viz 3.3). Vhodným senzorem se jeví senzor TFMINI-S (tabulka 3.9, obrázek 3.6), jehož udávaná minimální detekční vzdálenost je 10 cm, což je více, než 2 cm udávané v případě ultrazvukových senzorů HC-SR04, ovšem stále je možňé zachytit i překážku umístěnou velmi blízko před autíčkem.

Nevýhodou senzoru TFMINI-S je úzký úhel záběru pouhých 2° ve srovnání s 12° u senzoru HC-SR04. Toto ve spojení s tím, že k dispozici byl pouze jeden senzor TFMINI-S, přičemž jeho jednotková cena je ve srovnání se senzorem HC-SR04 mnohonásobně vyšší, přinášelo výzvu v podobě toho, jak zajistit dostatečné pokrytí přední polosféry autíčka pro detekci překážky.

Odpověď se nabízí v podobě sestrojení zjednodušeného LIDARu (3.3) za využití servomotoru, na kterém bude připevněn senzor TFMINI-S, jehož měření bude sladěno s pohybem servomotoru takovým způsobem, aby byl získáván dostatečný přehled o situaci před autíčkem. Tímto způsobem je možné jedním senzorem provézt měření vzdálenosti překážek po celé přední polosféře autíčka a překonat nedostatek dostupných senzorů. Toto řešení také umožňuje jistou úroveň škálovatelnosti na základě toho, zda se v další fázi vývoje ukáže být původně zvolený počet jednotlivých měření jako nízký, nebo vysoký a jednoduše podle tohoto přizpůsobit senzor. Nevýhodou tohoto řešení je, že pohyb servomotoru zabere jistý čas a teoreticky tak vznikají na trati po určitou dobu slepá místa do doby, než senzor odrotuje do potřebné pozice. Z tohoto důvodu může být sestrojený senzor náchylnější na situaci, kdy je překážka vložena bezprostředně před vozítko, jelikož do doby, kdy vozítko stihne narazit do překážky, nemusí být senzor schopen překážku detekovat, protože bude natočený jiným směrem.

Vhodným servomotorem pro realizaci senzoru je typ MG996R (tabulka 4.2), který již byl při realizaci použit na řízení přední natáčecí nápravy. Jeho řízení rovněž bude zajištěno prostřednictvím modulu PCA9685, přes něhož jsou řízeny všechny elektromotory autíčka a po zapojení dalšího servomotoru bude mít modul stále k dispozici 12 volných kanálů. Právě použití servomotoru MG996R společně s modulem PCA9685 přináší výhodu v minimalizaci možných komplikací při vývoji, jelikož je do celého systému přidávána pouze jedna nová součástka, senzor TFMINI-S, zatímco zbývající součástky již byly v rámci vývoje použity.

Na obrázku 6.2 je v blokovém schématu znázorněna podoba řídícího a detekčního systému autíčka po svém rozšíření o systém pro detekci překážek. Napájení servomotoru bude obdobně jako v případě zbylých elektromotorů autíčka zajišťováno přes modul PCA9685 (tabulka 5.3), který je napájen z akumulátoru Dinogy Graphene (tabulka 5.1) prostřednictvím regulátoru Foxy UBEC. Napájení senzoru TFMINI-S bude realizováno skrze platformu Jetson Nano přes přítomný 5 V napájecí pin. Komunikace mezi platformou Jetson Nano a senzorem TFMINI-S bude probíhat skrze I2C rozhraní, kdy získaná data budou zpracovávána na platformě Jetson Nano.



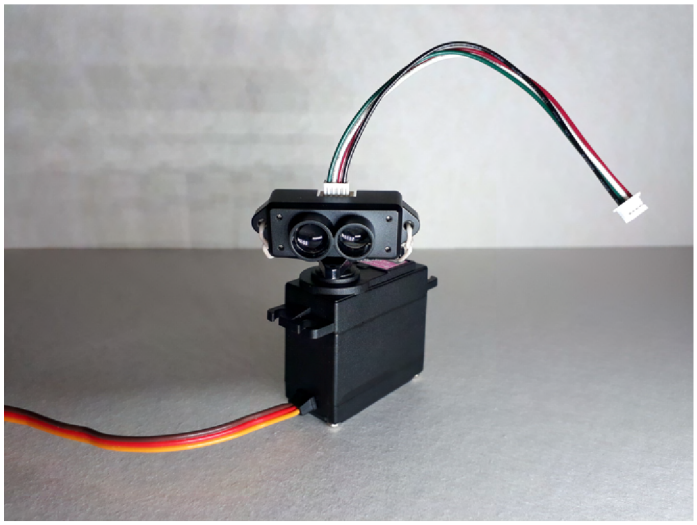
Obrázek 6.2: Blokové schéma celého systému autonomního robotického autíčka.

6.2 Realizace systému

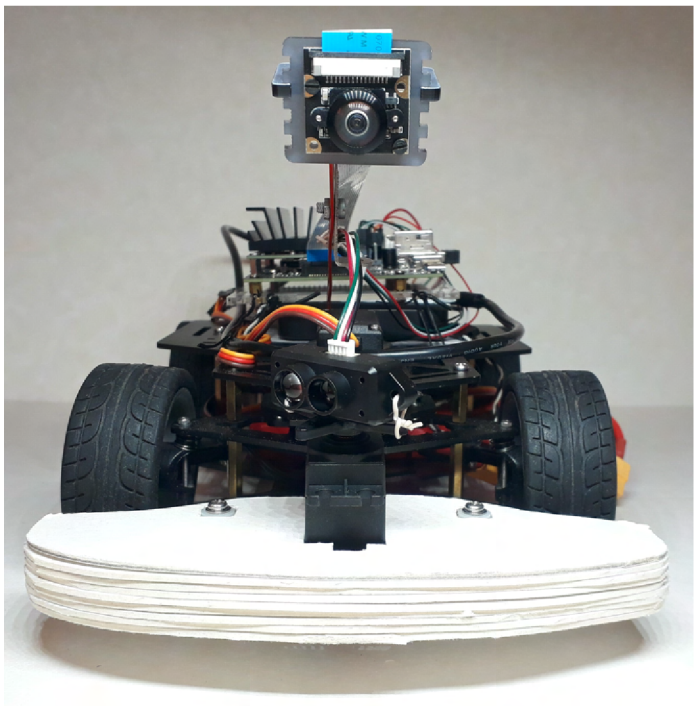
Realizace systému pro detekci překážek byla rozdělena do dvou částí. První z nich bylo sestrojení samotného LIDARu a implementace programu pro práci s ním, čemuž se věnuje sekce 6.2. Následně bylo nutné implementovat úpravy řídícího algoritmu, představeného v podkapitole 5.4, aby autíčko vhodně reagovalo na detekovanou překážku úpravou trajektorie, nebo úplným zastavením. Této problematice je věnována sekce 6.2.

Implementace senzoru

Na obrázku 6.3 je zachycena fyzická realizace navrženého LIDARu (viz 6.1), skládající se ze senzoru TFMINI-S, jenž je uchycen na hřídeli servomotoru MG996R prostřednictvím kousku drátku. Tento senzor bude následně umístěn na přední část autíčka, odkud bude snímat překážky umístěné na trati. Umístění senzoru na autíčku je zachyceno na obrázku 6.4.



Obrázek 6.3: Sestrojený LIDAR, skládající se ze servomotoru MG996R a senzoru TFMINI-S. Menší čočka vpravo je vysílač světelného paprsku, větší čočka vlevo přijímač odraženého paprsku (galerie autora).

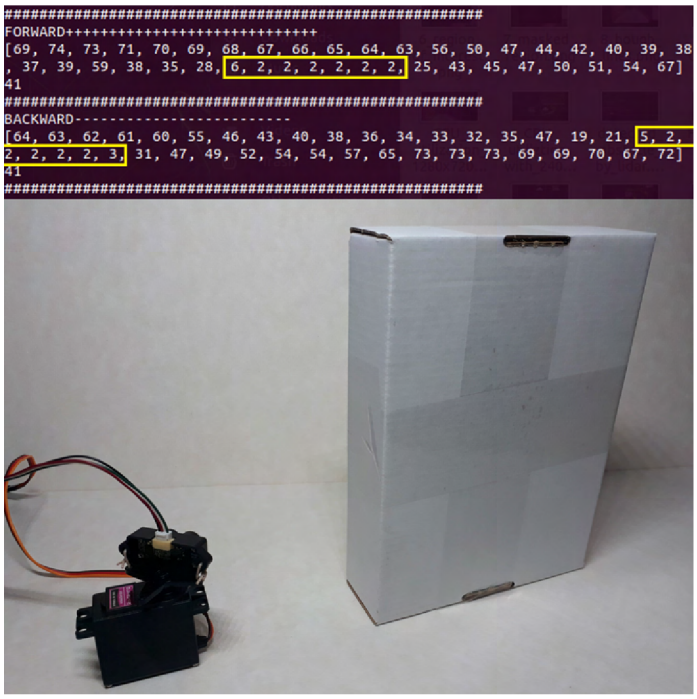


Obrázek 6.4: Umístění sestrojeného LIDARu na přední části autíčka. Pro ochranu senzoru a celé konstrukce byl z kartonu vyřezán nárazník na míru, jelikož originální nárazník z pěnové hmoty by bylo potřeba pro umístění senzoru upravit a navíc by, vzhledem k jeho velikosti, došlo při nárazu ke kontaktu LIDARu s překážkou (galerie autora).

Zatímco fyzická realizace navrženého LIDARu nebyla nikterak komplikovaná, tak softwarová část řešení se ukázala být hlavní výzvou. Jedním z problémů je sladění čtení ze senzoru s pohybem servomotoru takovým způsobem, aby aktualizace informace probíhala

dostatečně často, zároveň aby měření bylo dostatečně přesné a také aby bylo provedeno měření s potřebnou četností, jelikož úhel záběru senzoru TFMINI-S jsou pouhé 2°. Další komplikací je skutečnost, že použitý servomotor neumožňuje rotaci o 360° a vždy je nutné servomotor po provedení měřícího cyklu vrátit na výchozí pozici. Na výchozí pozici je možné se vrátit dvěma způsoby. V prvním případě, kdy například senzor rotuje a měří vzdálenosti zleva doprava, je možné po dokončení měřícího cyklu servomotor pouze otočit zpět doleva a další cyklus měření zleva doprava zahájit po návratu do výchozí pozice. Druhou možností je po provedení měření zleva doprava provézt toto měření vzdáleností i při pohybu zpět na výchozí pozici. V takovém případě, je však vzhledem k přesnosti servomotoru problematické dosáhnout toho, že každé jednotlivé měření se odehraje při přesně stejném natočení senzoru, jako tomu bylo při pohybu opačným směrem.

Tato synchronizace mezi měřeními zleva doprava a zprava doleva se ukázala být hlavní překážkou v dosažení vyšších rychlostí rotace senzoru, jelikož při změření vzdáleností v obou směrech byla v každém směru měření překážka detekována na jiném místě. Například zatímco podle měření zleva doprava byla překážka detekována na pravé straně dráhy, tak při pohybu zleva doprava již byla překážka detekována uprostřed dráhy. Tato situace je zachycena na obrázku 6.5. Problematická korelace mezi jednotlivými měřením poté znesnadňuje určení směru, jakým by se autíčko mělo pohybovat, pro úspěšné vyhnutí se překážce. V průběhu vývoje algoritmu nebyl nalezen spolehlivý způsob jak tento posun eliminovat bez razantního snížení rychlosti otáčení servomotoru, bylo možné jej pouze co nejvíce minimalizovat.



Obrázek 6.5: V horní části obrázku je přiložen výpis dat získaných sestrojeným LIDA-Rem při jednotlivých měřících cyklech. Cyklus nazvaný *FORWARD* provádí měření zleva doprava, zatímco cyklus *BACKWARD* opačným směrem. Žlutě jsou označena jednotlivá měření, která odpovídají překážce umístěné před senzorem, zachycené na obrázku pod výpisem výsledků. Ve výpisu výsledků je vidět patrný posun pozice dat, které odpovídají umístěné překážce (galerie autora).

V realizovaném systému bylo, pro minimalizaci slepého časového okna, využito měření při rotaci senzoru zleva doprava i zprava doleva. Tímto způsobem je tedy minimalizována doba, kdy je ve stejném bodě provedeno další měření vzdálenosti. V průběhu testování bylo zjištěno, že rychlost rotace servomotoru velmi ovlivňuje to, jaký posun detekované překážky bude mezi rotací jedním a druhým směrem v získaných datech nastávat. Při pomalejší rotaci servomotoru, kdy pohyb zleva doprava trval přibližně 1 s docházelo k velmi přesné detekci překážky. Na druhou stranu však takto pomalý pohyb senzoru vytvářel slepá místa, ve kterých se mohla překážka vyhnout detekci, případně nebyla detekována včas na to, aby na ni mohlo autíčko zareagovat.

I samotný počet jednotlivých měření provedených v průběhu rotace senzoru velmi ovlivňoval dobu, jakou tato rotace bude trvat. Nižší počet jednotlivých měření umožňoval zkrátit dobu jednoho měřícího cyklu, ovšem vzhledem k úzkému úhlu záběru senzoru TFMINI-S poskytoval pouze hrubou informaci o pozici překážky. S vyšším počtem měření je možné přesněji zjistit pozici překážky a teoreticky odhadnout i její velikost, avšak s vyšším počtem měření opět roste náročnost na synchronizaci mezi jednotlivými měřícími cykly.

Výsledná implementace měřícího cyklu poskytuje zorné pole o šířce cca 90°, tedy 45° na každou stranu od střední podélné osy autíčka. Toto zorné pole poskytuje dostatečný přehled o situaci před autíčkem a splňuje požadavky na detekci překážky, umístěné na rovných úsecích závodní trati, dle pravidel soutěže NXP Cup [20]. Při jednom měřícím cyklu je provedeno 41 individuálních měření a tedy krok mezi jednotlivými měřeními je po zaokrouhlení okolo 2°. S rostoucí vzdáleností překážky od senzoru samozřejmě dochází k tomu, že z pohledu vzdálenosti jsou jednotlivé světelné paprsky od sebe vzdáleny tím více, čím dále od senzoru se nacházejí. Překážka o velikosti $20 \times 20 \times 20$ cm se tedy projeví na tím větším počtu jednotlivých měřeních z celkových 41, čím blíže se nachází autíčku.

Na ukázce kódu níže se nachází funkce `lidarFunctionForwards` provádějící měřící cyklus při rotaci senzoru zleva doprava. Funkce provádí rotaci senzoru v rozsahu hodnot 420 až 220, který odpovídá fyzické šířce záběru 90°, přičemž krok o velikosti 5 bodů odpovídá 2°. Jednotlivá měření jsou ukládána do sdíleného seznamu, který je po svém dokončení zpracován v hlavní smyčce řídícího programu. Pro rotaci opačným směrem je využita podobná funkce `lidarFunctionBackwards`, která provádí rotaci opačným směrem, pouze na konci měřícího cyklu provede obrácení pořadí prvků v seznamu tak, aby jednotlivá měření byla ve sdíleném seznamu pro další zpracování řazena vždy zleva doprava. Jeden měřící cyklus trvá přibližně 0,5 s z čehož lze odvodit, že sestrojený LIDAR analyzuje prostor před autíčkem 2× za sekundu.

```
1  def lidarFunctionForwards(lidar_shared_list):
2    i = LIDAR_LEFT_START_POINT
3
4    while i >= LIDAR_RIGHT_END_POINT:
5
6      bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_3_END, int(i))
7      lidar_shared_list.append(readLidarData())
8      i -= 5
9
10   bus.write_word_data(BOARD_I2C_ADDR, CHANNEL_3_END, int(i+40))
11   time.sleep(0.01)
```

Jelikož se autíčko pohybuje po dráze a provádí neustále její detekci a provádí korekce směru a rychlosti průměrně 25× za 1 s, tak je nutné zajistit, aby cyklus měření vzdálenosti, prostřednictvím sestrojeného senzoru, neblokoval vykonávání zbylých částí programu. Če-

kání na provedení měřícího cyklu LIDARu by při sekvenčním vykonávání znemožňovalo jakékoliv řízení autíčka. Z tohoto důvodu je nutné řídící program paralelizovat takovým způsobem, aby jednotlivé měřící cykly LIDARu probíhaly v průběhu času, po který jsou zpracovávány snímky z obrazového snímače a řízen pohyb autíčka a nedocházelo k blokování vykonávání těchto činností. Realizovaný program tedy realizuje asynchroní obsluhu událostí, kdy do doby, než senzor dokončí měřící cyklus, nedochází k jeho obsluze. Jakmile je zjištěno, že senzor dokončil měření a sdílený seznam obsahuje 41 hodnot, jsou tato data zpracována a je spuštěn měřící cyklus opačným směrem.

Za účelem paralelizace byla použita knihovna multiprocessing[1], jež umožňuje v rámci programu vytvářet a spouštět procesy, které jsou vykonávané paralelně k procesu, z něhož jsou spouštěny. Způsob realizace paralelního zpracování (na ukázce níže) využívá pro předávání dat mezi procesy sdílený seznam, který je předán formou argumentu funkci `lidarFunctionForwards` nebo `lidarFunctionBackwards`. Proces nazvaný `process_lidar` je spuštěn s jednou z těchto funkcí (řádek 5, případně 9) a následně jakmile dojde k dokončení měřícího cyklu, tak je tento proces ukončen (řádek 14), načež dojde ke kopii výsledků, vymazání sdíleného seznamu a na základě změny proměnné *lidar_backwards* dojde v dalším cyklu hlavní smyčky programu k vytvoření nového procesu s funkcí, která provede cyklus měření opačným směrem.

```
1  if lidar_status is True:
2    if ser.is_open == False:
3      ser.open()
4
5    if len(lidar_shared_list) == 0 and lidar_backwards is False:
6      process_lidar= multiprocessing.Process(target=lidarFunctionForwards,
       args=[lidar_shared_list])
7      process_lidar.start()
8
9    if len(lidar_shared_list) == 0 and lidar_backwards is True:
10     process_lidar = multiprocessing.Process(target=lidarFunctionBackwards
       , args=[lidar_shared_list])
11     process_lidar.start()
12
13   if len(lidar_shared_list) == 41:
14     process_lidar.join()
15
16     if lidar_status is True:
17       if keyboard.is_pressed('n'):
18         lidar_status = False
19
20     lidar_shared_list_copy = lidar_shared_list.__deepcopy__({})
21     lidar_shared_list = manager.list()
22
23     if lidar_backwards is False:
24       lidar_backwards = True
25     elif lidar_backwards is True:
26       lidar_backwards = False
```

Samotné vytváření a správa paralelního procesu vyžaduje jistou část dostupného výkonu. Rovněž bylo v průběhu vývoje zjištěno, že implementované ovládání v podobě aktivace funkcí autíčka stiskem klávesy **E**, které provede prvotní spuštění elektromotorů a předního servomotoru, naruší běžící proces obstarávající funkci LIDARu, případně i pře-

[1] https://docs.python.org/3/library/multiprocessing.html

ruší komunikaci se senzorem TFMINI-S. Z důvodu této komplikace a také snížení průměrné snímkové frekvence detekce čar, při spuštěném LIDARu, je LIDAR v rámci řídícího systému autonomního autíčka doplňkovou funkcí, kterou je možné aktivovat klávesou **M**. V případě nutnosti lze také senzor deaktivovat dlouhým stiskem klávesy **N** až do doby, kdy se rotace senzoru zastaví v nejbližší krajní pozici aby byl proces, v rámci kterého je LIDAR řízen, korektně ukončen, načež je posléze možné LIDAR opět aktivovat. Tato možnost volitelné aktivace LIDARu umožňuje v soutěžních disciplínách, ve kterých není nutné LIDAR mít aktivovaný, používat autíčko bez této funkce a dosáhnout vyšších snímkových frekvencí při zpracování obrazu. V případě potřeby aktivace LIDARu je nutné nejdříve provézt aktivaci motorů autíčka klávesou **E** a až poté může dojít ke spuštění LIDARu klávesou **M**. Pro účely vývoje a testování byla v programu ponechána možnost spustit LIDAR klávesou **M** bez nutnosti provézt nejdříve spuštění klávesou **E** pro situace, kdy není vyžadováno spuštění motorů a proto v situaci, kdy je vyžadováno spuštění motorů je nutné, aby správnou sekvenci spuštění zajistil uživatel.

Implementace reakce na překážku

Data získaná prostřednictvím sestrojeného LIDARu, představeného v předchozí části 6.2, jsou využívána pro úpravy úhlu natočení přední nápravy. V případě, kdy je LIDAR aktivován a dojde k provedení jednoho měřícího cyklu, tak jsou veškerá měření uložená v seznamu přefiltrována a uložena do nového pomocného seznamu. Při tomto procesu jsou všechna měření, která zaznamenávají vzdálenost větší než 100 cm, přepsána na hodnotu 100 cm a veškerá měření menší než 100 cm mají zanechánu původní hodnotu. Tento proces zejména sloužil ke zpřehlednění vstupních dat při ladění programu, avšak byl ve výsledné implementaci ponechán pro možné budoucí využití a také proto, že pro zbývající implementaci jsou hodnoty větší než 100 cm nepodstatné.

V další fázi zpracování dochází k průchodu jednotlivými měření a v případě, že je nalezeno alespoň jedno měření menší než 5 cm, je nastavena hodnota proměnné *FULL_STOP* na hodnotu *True* a smyčka procházející jednotlivé naměřené hodnoty je ukončena a program přechází rovnou do části, ve které dochází k úpravě úhlu natočení přední nápravy a rychlosti, kde na základě hodnoty této proměnné dojde k okamžitému zastavení autíčka. Pro jeho opětovné spuštění je nutné stisknout klávesu **O**, která opět spustí pohyb autíčka, pokud se již v zorném poli senzoru nebude nacházet překážka.

Implementovaný způsob interpretace naměřených výsledků předpokládá, že se autíčko bude podle čar držet uprostřed dráhy a podle tohoto předpokladu rozlišuje, mimo již představenou reakci na velmi blízkou překážku, kdekoliv v zorném poli, další čtyři situace, které vycházejí z požadavků na detekci překážek v soutěži NXP Cup [20].

Nejdříve dochází k zjištění překážek nacházejících se u levého, nebo pravého okraje dráhy. Tuto jednoduchou situaci znázorňuje ukázka kódu níže, kdy z celého seznamu 41 jednotlivých měření jsou posouzena první tři měření, zda jsou menší než 50 cm (překážka na levé straně dráhy). Obdobně jsou porovnána také poslední tři měření v seznamu (překážka na pravé straně dráhy).

```
1 tmp_first_3_elems = tmp_lidar_shared_list[0:3]
2
3 for elem in tmp_first_3_elems:
4   if elem < 50:
5     left_obstacle_cnt += 1
```

Pokud jsou všechna tři první porovnávaná měření menší než 50 cm (ukázka níže) dojde k úpravě hodnoty proměnné `servo_angle` určující úhel natočení přední nápravy o definovanou konstantu, což ve výsledku způsobí zatočení autíčka doprava a jeho vyhnutí se překážce. Obdobně pokud jsou všechna tři poslední měření menší než 50 cm a překážka se tedy nachází na pravé straně dráhy, dojde k zatočení doleva.

```
if left_obstacle_cnt >= 3:
  turn_right_const = 80
  if (servo_angle + turn_right_const >= 190) and (servo_angle +
    turn_right_const <= 420):
    servo_angle = servo_angle + turn_right_const
  elif (servo_angle + turn_right_const <= 190):
    servo_angle = 190
  elif (servo_angle + turn_right_const >= 420):
    servo_angle = 420
```

Další situací je moment, kdy ze získaných 41 měření jsou v alespoň 25 případech naměřeny hodnoty nižší než 50 cm. Tato situace značí velkou překážku zabírající celou dráhu a proto dojde opět k zastavení autíčka. Poslední situací je překážka nacházející se uprostřed dráhy, kterou není možné objet bez vyjetí z dráhy. V případě, kdy je zaznamenáno 10 měření s hodnotou menší než 50 cm a zároveň nejsou všechna 3 první a všechna 3 poslední měření v seznamu menší než 50 cm, tak dojde k zastavení autíčka.

Autíčko tedy rozlišuje celkem pět situací - překážku blíže než 5 cm, překážku přes celou dráhu a překážku uprostřed dráhy, přičemž v těchto situacích je nutné zastavit, a překážku na levé nebo na pravé straně, kdy je proveden úhybný manévr.

Veškeré uvedené hodnoty vzdáleností a počtu měření jsou výsledkem experimentů s autíčkem při jízdě po dráze. Sestrojený senzor a na jeho základě implementovaný systém detekce překážek vykazoval při testování nižší úspěšnost detekce překážky a následné reakce na ní, než jaké dosahuje systém detekce dráhy a značek na dráze. Jedním z důvodů pro toto jsou omezení plynoucí ze samotné konstrukce senzoru, který, pro zachování akceptovatelného posunu překážky mezi jednotlivými měřícími cykly, provádí pouze dva měřící cykly za sekundu. Tato skutečnost poskytuje dostatečně velké časové okno pro situaci, kdy je překážka mimo zorné pole senzoru a ve chvíli, kdy dojde k rozeznání překážky řídícím systémem, je autíčko již příliš blízko na úspěšné vyhnutí se překážce. Vyšší úspěšnosti, než při vyhýbání se překážce, je dosahováno v případě, kdy je nutné před překážkou zastavit. S těmito situacemi si navržený senzor a implementovaný systém dokáže poradit úspěšněji.

Přes nedostatky tohoto řešení se však povedlo zkonstruovat zajímavý senzor, který rozšiřuje možnosti použití jednoho statického ToF senzoru TFMINI-S, kdy při zhruba poloviční pořizovacích nákladech oproti pořízení některého z nejlevnějších komerčních LIDARů, jakým může být například RPLIDAR A1 (obrázek 3.10) představený v podkapitole 3.3, je možné alespoň do určité míry získat podobné schopnosti, jako při použití komerčního řešení.

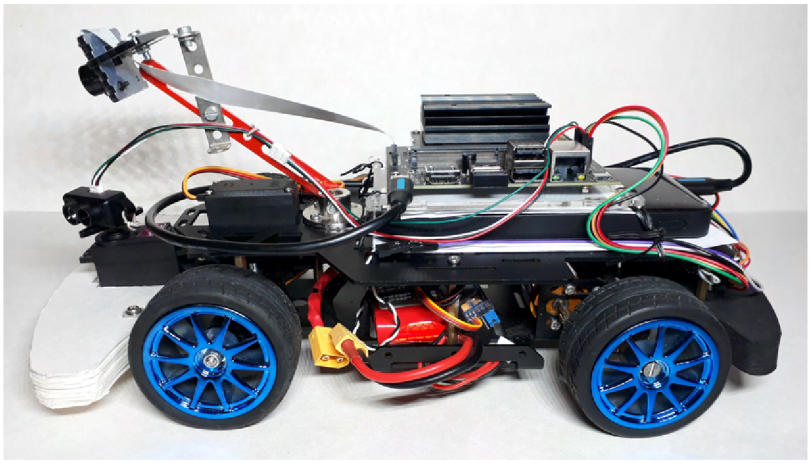
Kapitola 7

Výsledný systém a jeho testování

Kapitola představuje výsledky celé práce a je rozdělena do tří samostatných podkapitol. První podkapitola 7.1 je věnována výslednému zapojení realizovaného autíčka. Podkapitola 7.2 se zaměřuje na realizaci testovací závodní dráhy z pohledu použitých materiálů a způsobu její výroby. Poslední podkapitola s číslem 7.3 se zabývá testováním výsledného systému na 5 různých testovacích tratích.

7.1 Výsledná realizace autonomního autíčka

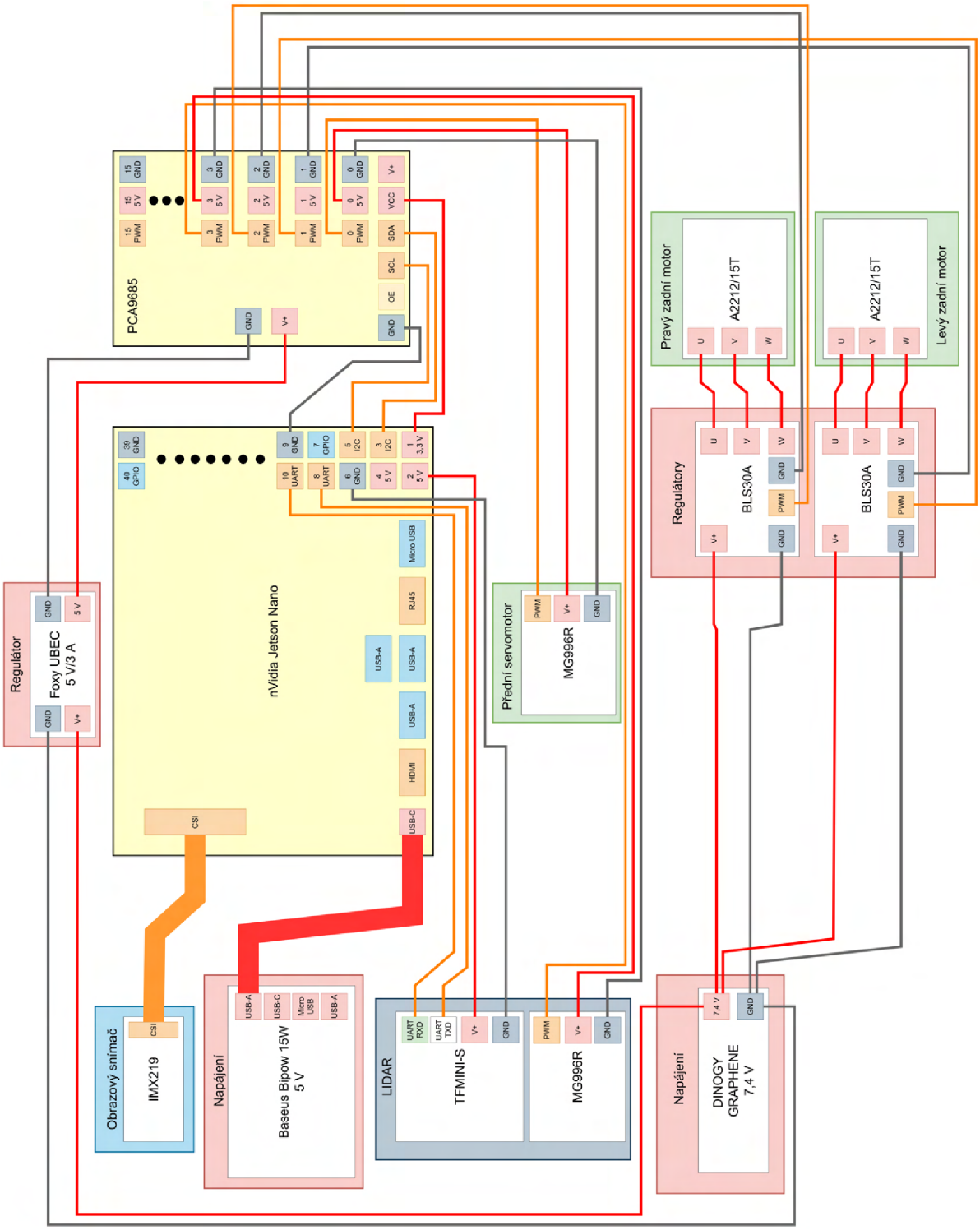
Obrázek 7.1 zachycuje výslednou fyzickou podobu autíčka při pohledu na jeho levý bok. Oproti původní podobě autíčka realizované Ing. Viktorem Steingartem (obrázek 4.1) se podařilo dosáhnout mnohem nižší výšky umístění kamery. Čočka kamery se nachází ve výšce 14 cm nad zemí, přičemž její úhel vůči zemi je 70°. Kompaktnější podoba autíčka však přináší přináší drobnou nevýhodu v podobě vzdálenosti, do jaké kamera ze svého umístění dohlédne, což do jisté míry limituje schopnost detekce krajních čar dráhy. Tato skutečnost může například v situaci kdy autíčko směřuje na jednu z krajních čar vézt k tomu, že druhá krajní čára není detekována, zatímco v případě kdy by kamera byla umístěna výše a snímala dráhu pod větším úhlem, tak by druhá krajní čára byla stále v zorném poli.



Obrázek 7.1: Pohled na levý bok sestrojeného autíčka (galerie autora).

Na obrázku 7.2 je schéma zapojení celého systému. Pro usnadnění orientace jsou veškeré součásti napájecího sytému ohraničeny červeným čtyřúhelníkem. Červenou barvou jsou rov-

něž značeny veškeré napájecí spoje. Oranžové vodiče jsou komunikačními linkami pro PWM signál, UART, nebo I2C. Prvky pohonného ústrojí jsou označeny zeleným čtyřúhelníkem a komponenty tvořící sestrojený LIDAR šedým čtyřúhelníkem. Řízení všech komponent je zajišťováno pomocí žlutě zvýrazněné platformy Jetson Nano a modulu PCA9685.



Obrázek 7.2: Schéma zapojení veškerých součástí celého systému.

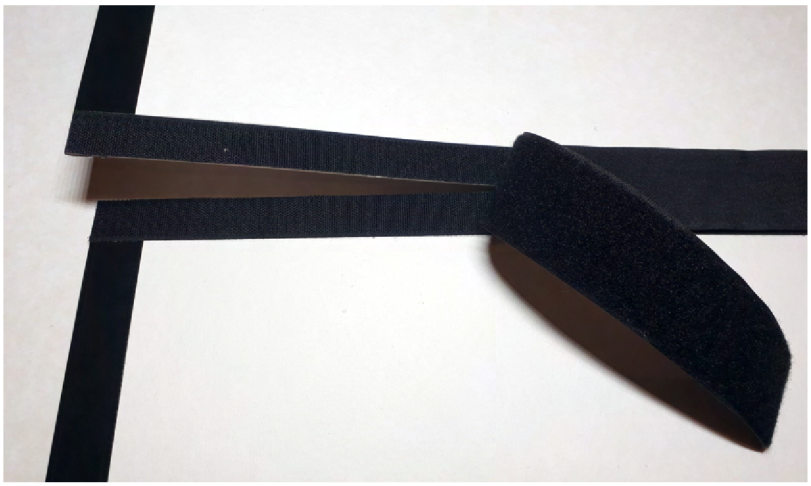
7.2 Realizace závodní dráhy

V rámci této práce bylo pro účely vývoje a testování nutné vytvořit závodní dráhu dle specifikací soutěže NXP Cup [20, 23], částečně představených v podkapitole 2.3. Většina závodních okruhů představených ve specifikaci dráhy je složena pouze ze dvou typů dílů, jimiž jsou rovinka a zatáčka. Tyto díly je posléze možné spojovat do okruhů nejrůznějších tvarů a velikostí, případně rozšířit o speciální díly jako například křižovatku, případně o díly se značkami začátku a konce úseku s omezenou rychlostí. Pro účely této práce byly vyrobeny díly právě těchto typů.

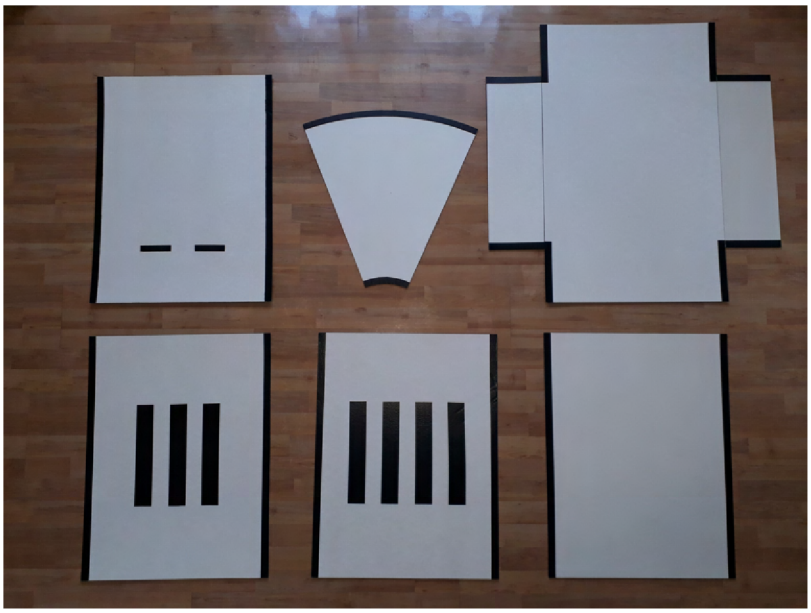
Jako materiál pro výrobu jednotlivých dílů byl zvolen karton bílé barvy s tloušťkou 2 mm. Původně bylo plánováno vyřezat jednotlivé díly na laserové řezačce, ovšem rozměry kartonu 70×100 cm vyžadovaly, aby byly kartonové desky zmenšeny ještě před samotným řezáním, jelikož takto velké desky nebylo možné do řezačky umístit. Také se u prvních dvou ručně vyrobených prototypových dílů ukázalo, že systém spojování dílů podle specifikací je problematický pro použitý druh kartonu, jelikož jednotlivé díly v sobě vzhledem k nízké tloušťce kartonu nedržely zajištěné.

Vzhledem k těmto poznatkům bylo rozhodnuto nepoužívat spojovací sytém dle specifikací dráhy NXP Cup a místo toho na spoji mezi jednotlivými díly ponechat rovnou hranu a z druhé strany kartonu přilepit oboustrannou lepící páskou pás suchého zipu a jednotlivé díly tedy spolu spojovat s jeho pomocí, jako na obrázku 7.3. Potřebné 2 cm široké černé okraje každého dílu, ohraničující závodní dráhu, byly vytvořeny černou lepící páskou.

Zjednodušení hran spojujících jednotlivé díly, oproti jejich specifikaci, umožnilo plně vynechat potřebu vyřezávat díly na laserové řezačce, jelikož řezání na laserové řezačce by, vzhledem k času nutnému k přepravě materiálu a přípravě modelů pro řezání, nepřineslo výraznou úsporu času při jejich výrobě a tak byly veškeré díly z kartonových desek vyřezány ručně. Veškeré vyrobené typy dílů jsou zachyceny na obrázku 7.4



Obrázek 7.3: Pohled na spodní stranu dílů dráhy a detail jejich způsobu spojení suchým zipem v jeden celek (galerie autora).



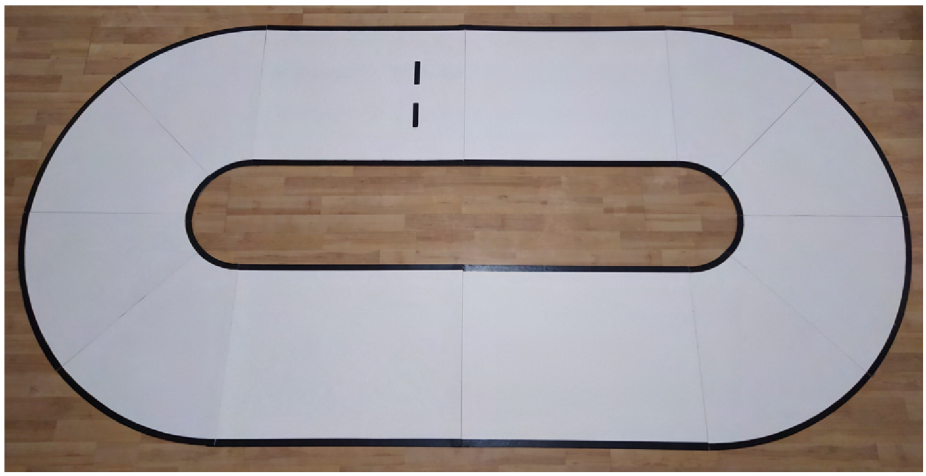
Obrázek 7.4: Veškeré typy dílů vyrobené pro účely testování. Při pohledu po řádcích zleva se jedná o rovný díl se značkou začátku dráhy, zatáčku, křižovatku, rovný díl se značkou konce úseku s omezenou rychlostí, rovný díl se značkou pro začátek úseku s omezenou rychlostí a základní rovný díl (galerie autora).

7.3 Testování

Testování autíčka probíhalo na několika tvarech závodní dráhy. Prvním a zároveň nejjednodušším okruhem je ovál tvořený dvěma rovnými úseky, skládajících se ze dvou rovných dílů, které jsou spolu propojeny zatáčkou o 180° na obou koncích (část 7.3). Druhou testovací dráhou je ovál se značkami značící začátek a konec úseku z omezenou rychlostí pro testování detekce značek, představený v části 7.3. Třetím okruhem je upravená varianta oválného okruhu s upravenou pasáží zatáčky v části 7.3. Čtvrtou testovací trať tvoří okruh s křižovatkou (7.3) a posledním testovacím okruhem je trať s umístěnou překážkou (viz 7.3). Na každé testovací trati byla měřena průměrná rychlost jednoho kola a také sledována úspěšnost s jakou se autíčko drží na vymezené dráze a daří se mu plnit případnou detekci značek nebo překážek.

Oválný okruh

Oválný okruh, jehož podoba je zachycena na obrázku 7.5, je nejjednodušší variantou závodního okruhu použitou pro testování. Testovací proces se skládal z ujetí 100 koleček po závodním okruhu, z čehož prvních 50 bylo po směru a zbylých 50 kol proti směru hodinových ručiček. V tabulce 7.1 jsou shrnuty výsledky testování, z nichž je patrné, že ani v jednom testovacím kole nedošlo k vyjetí z dráhy, dokonce ani k najetí na čáru vymezující dráhu a tedy autíčko dosahovalo na tomto okruhu v rámci 100 testovacích kol 100 % úspěšnost, při průměrném času ujetí jednoho kola 14,6 s.



Obrázek 7.5: Oválný závodní okruh (galerie autora).

Čísla kol	Směr jízdy	Počet najetí na čáru	Úspěšnost [%]	Počet vyjetí z dráhy	Úspěšnost [%]
1-50	po směru	0	100	0	100
51-100	proti směru	0	100	0	100
1-100	-	0	100	0	100

Tabulka 7.1: Výsledky testování na oválném okruhu.

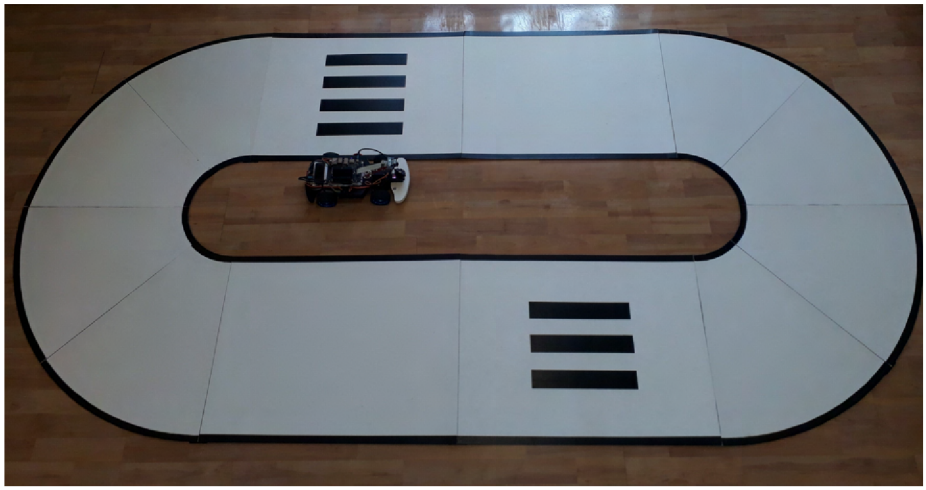
Okruh s rychlostními značkami

Okruh s rychlostními značkami na obrázku 7.6 představuje upravenou variantu oválného okruhu, ve kterém byly dva rovné díly nahrazeny rovnými díly se značkou označující začátek a konec úseku s omezenou rychlostí. Na tomto okruhu bylo rovněž provedeno 100 testovacích kol, přičemž prvních 50 z nich bylo ve směru, kdy se značka, kterou bylo nutné rozpoznat, nacházela bezprostředně za zatáčkou. Zbylých 50 kol autíčko jezdilo opačným směrem, kdy k rozeznání značky muselo dojít před zatáčkou. Při testování bylo kromě vyjetí z dráhy a najetí na okrajovou čáru sledováno, zda autíčko správně zareaguje na danou značku.

Při testování na tomto okruhu již, na rozdíl od jednoduchého oválného okruhu, došlo k situaci, kdy autíčko vyjelo z dráhy (tabulka 7.2). K vyjetí došlo celkem 5× a tedy úspěšnost držení se na dráze dosahovala 95 %. Autíčko také celkem 5 × najelo na krajní čáru bez vyjetí z dráhy, což je opět úspěšnost 95 %, ovšem tento parametr je sledován pouze pro analýzu výsledků, jelikož najetí na hraniční čáru neznamená diskvalifikaci. K těmto situacím dochází v případě tohoto testovacího okruhu vzhledem ke způsobu, jakým autíčko detekuje hraniční čáry dráhy. Detekované pozice těchto čár jsou výsledkem průměru všech detekovaných hran a tedy jsou do nich započítány i samotné značky, které v analyzovaném snímku vytváří nezanedbatelný počet hran, které se promítnou do pozice krajních čar a autíčko tak v této situaci nemá zcela přesnou informaci o aktuální pozici na dráze. Tento jev je do jisté míry kompenzován zavedením mechanismu, který, v případě kdy je na dráze detekována značka, zamezí zatáčení autíčka s cílem minimalizovat případnou možnost zatočení mimo dráhu.

V tabulce 7.3 jsou uvedeny výsledky detekce značek. V případě značky pro zpomalení došlo pouze k jednomu případu, kdy nebyla rozeznána a tedy úspěšnost její detekce je 99 %.

Značka konce zóny s rychlostním omezením pak nebyla rozeznána ve 2 případech, což značí úspěšnost 98 %. Průměrný čas, který autíčku trvalo zajet na okruhu jedno kolo, dosahoval hodnoty 18,7 s. Z testování při vývoji bylo zjištěno, že úspěšnost detekce značek závisí do značné míry na osvětlení dráhy. Mnohdy přesvícené okolí dráhy vytváří falešné obrazce, které jsou interpretovány jako značky. Na druhou stranu však při nedostatečném osvětlení značky na dráze dochází k tomu, že není autíčkem rozeznána.



Obrázek 7.6: Závodní okruh s rychlostními značkami (galerie autora).

Čísla kol	Směr jízdy	Počet najetí na čáru	Úspěšnost [%]	Počet vyjetí z dráhy	Úspěšnost [%]
1-50	po směru	4	92	3	94
51-100	proti směru	1	98	2	96
1-100	-	5	95	5	95

Tabulka 7.2: Výsledky testování na okruhu se značkami z pohledu úspěšnosti, s jakou se autíčko drželo na dráze.

Čísla kol	Směr jízdy	Počet nerozeznání značky pro zpomalení	Úspěšnost [%]	Počet nerozeznání značky pro zrychlení	Úspěšnost [%]
1-50	po směru	0	100	0	100
51-100	proti směru	1	98	2	96
1-100	-	1	99	2	98

Tabulka 7.3: Výsledky testování na okruhu se značkami z pohledu úspěšnosti detekce značek.

Pokročilý oválný okruh

Na obrázku 7.7 se nachází upravený oválný okruh, jehož jedna hlavní zatáčka byla rozšířena o zatáčky opačným směrem. I na tomto okruhu bylo provedeno 100 testovacích kol, kdy první polovinu kol jezdilo autíčko jedním směrem a druhou polovinu druhým směrem.

V tabulce 7.4 jsou zaznamenány výsledky testovacích kol. Celkově došlo ke 2 případům, kdy autíčko vyjelo z dráhy, což značí úspěšnost 98 %. Dohromady došlo k poměrně vysokému množství najetí na krajní čáru a to celkem 18× z čehož plyne úspěšnost 82 %, ovšem jak již bylo zmíněno, tak najetí na čáru dle pravidel soutěže NXP Cup neznamená diskvalifikaci. Průměrný čas potřebný k zajetí jednoho kola byl velmi podobný času na jednoduchém oválném okruhu (viz 7.3) a dosahoval hodnoty 15,2 s.

Obrázek 7.7: Pokročilý oválný okruh (galerie autora).

Čísla kol	Směr jízdy	Počet najetí na čáru	Úspěšnost [%]	Počet vyjetí z dráhy	Úspěšnost [%]
1-50	po směru	10	80	1	98
51-100	proti směru	8	84	1	98
1-100	-	18	82	2	98

Tabulka 7.4: Výsledky testování na upraveném oválném okruhu z pohledu úspěšnosti, s jakou se autíčko drželo na dráze.

Okruh s křižovatkou

Okruh s křižovatkou, na obrázku 7.8, představuje nejdelší z testovacích okruhů, na kterém představuje hlavní výzvu křížení cest, na němž by autíčko mělo vždy pokračovat rovně. I na tomto okruhu bylo provedeno 100 testovacích kol, vždy 50 kol jedním směrem, jejichž výsledek odpovídá očekávání. V případě, kdy autíčko jede ve směru, ve kterém do křižovatky najíždí z dlouhé rovinky je úspěšnost velmi vysoká, jelikož autíčko při ztrátě informace o krajních čarách pouze pokračuje rovně až do chvíle, kdy opět narazí na krajní čáry za křižovatkou. Pokud autíčko do křižovatky najíždí přímo ze zatáčky, tak je jeho úspěšnost

podstatně nižší a často dochází k vyjetí z dráhy. Tento jev je způsoben zejména umístěním kamery nízko nad autíčko, kdy po vyjetí ze zatáčky a vjetí do křižovatky nemá autíčko k dispozici informaci o své pozici na dráze, jelikož v záběru kamery se nenacházejí žádné krajní čáry. Autíčko tedy pokračuje rovně směrem, kterým vyjelo ze zatáčky až do chvíle, kdy projede křižovatkou a opět dojde k detekování krajních čar. Jelikož však v situaci, kdy autíčko do křižovatky vjíždí bezprostředně po zatáčce, není dostatek času a informací aby došlo ke srovnání pozice na střed dráhy, tak v mnoha případech stačí autíčko vyjet z dráhy dříve, než se dostane do bodu, kdy je opět možné detekovat hraniční čáry za křižovatkou.

V tabulce 7.5 jsou zaznamenány výsledky, kdy v situaci kdy autíčko do křižovatky najíždí z dlouhé rovinky byla úspěšnost držení se na dráze 100 %. Došlo pouze ke 4 situacím, kdy autíčko najelo na krajní čáru a v tomto případě tedy byla z 50 pokusů úspěšnost 92 %. Při jízdě opačným směrem pak z 50 pokusů došlo v 19 případech na najetí na čáru, avšak autíčko se ještě zvládlo udržet na dráze, což značí úspěšnost 62 %. Případů vyjetí z dráhy bylo 16, z čehož lze odvodit úspěšnost držení se na dráze na pouhých 68 %. V případě okruhu s křižovatkou pak autíčku trvalo ujetí jednoho kola průměrných 25,1 s.

Obrázek 7.8: Závodní okruh s křižovatkou (galerie autora).

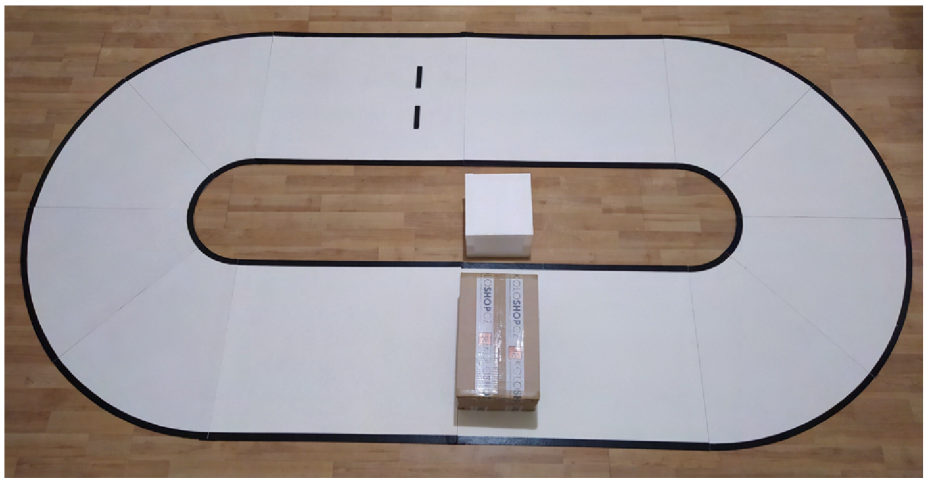
Čísla kol	Směr jízdy	Počet najetí na čáru	Úspěšnost [%]	Počet vyjetí z dráhy	Úspěšnost [%]
1-50	z dlouhé rovinky do křižovatky	4	92	0	100
51-100	ze zatáčky do křižovatky	19	62	16	68
1-100	-	23	77	16	84

Tabulka 7.5: Výsledky testování na okruhu s křižovatkou.

Okruh s překážkou

Okruh s překážkou sloužil k testování reakce autíčka na několik typů překážek. Celkem bylo provedeno 100 testovacích kol, avšak v případě tohoto okruhu bylo těchto 100 kol děleno na 4 skupiny po 25 testovacích kolech, kdy v každé skupině bylo testováno jiné umístění překážky, namísto obvyklého dělení na 2 skupiny po 50 kolech. Rozdílem, oproti předchozím testovacím okruhům, také byla skutečnost, že již nedocházelo k jízdě po dráze v obou směrech, ale autíčko po dráze jelo všech 100 kol stále stejným směrem. Pro ujetí jednoho testovacího kola muselo autíčko ujet jedno kolo po dráze bez překážky a ihned jakmile projelo místem, kde měla být překážka umístěna, došlo k jejímu umístění tak, aby v následujícím kole již muselo autíčko na tuto překážku reagovat.

Nejdříve byla testována velká překážka blokující takřka celou šířku dráhy. Následovalo testování s překážkou dle specifikací soutěže NXP Cup [20] o rozměrech $20 \times 20 \times 20$ cm umístěnou na střed dráhy, posléze k pravému okraji dráhy a na závěr k levému okraji dráhy. Obrázek 7.9 zachycuje oválný okruh s umístěnou velkou překážkou. Na stejné místo byla umísťována i menší překážka zachycená na stejném snímku, v jejímž případě docházelo k jejímu posunu do stran pro vytvoření překážky na levé, nebo pravé straně dráhy.



Obrázek 7.9: Závodní okruh s překážkou. Na dráze je umístěna velká překážka, zatímco překážka o rozměrech $20 \times 20 \times 20$ se pro ukázku nachází uvnitř dráhy a při testovacích jízdách byla překážka vždy umístěna na stejné místo, na jakém se na obrázku nachází velká překážka, pouze docházelo k jejímu posunu k jednomu z okrajů (galerie autora).

Autíčko po dráze jezdilo po směru hodinových ručiček a tedy překážka se na rovném úseku nacházela blíže výjezdu ze zatáčky, než následujícímu vjezdu do zatáčky, ovšem nebyla umístěna bezprostředně za zatáčkou. V tabulce 7.6 jsou zaznamenány výsledky z pohledu situací kdy došlo k vyjetí ze závodní dráhy. K vyjetí z dráhy došlo v dohromady pouze ve 3 případech ze 100 pokusů a to 2× na okruhu s malou překážkou uprostřed dráhy a v jednom případě když byla překážka umístěna u levého okraje dráhy. Zajímavou situací, která při testování nastala, bylo vyjetí z dráhy těsně před překážkou, k němuž došlo ve dvou případech na okruhu s velkou překážkou zabírající celou šířku dráhy, což je způsobeno tím, že algoritmus detekující krajní čáry dráhy vnímal umístěnou krabici odlišné barvy jako hranu a detekovaná krajní čára se tak jevila jako přímo před autíčkem. Toto vedlo k tomu, že autíčko ve snaze udržet se na dráze ve skutečnosti vyjelo z dráhy dříve, než LIDAR zachytil překážku a tím pádem již nedošlo na zastavení před překážkou, ale autíčko

rovnou pokračovalo mimo překážku ven z dráhy. Při sečtení všech případů vyjetí z dráhy, bez rozlišování důvodu vyjetí, vychází úspěšnost držení se na dráze, při celkem 5 případech vyjetí z dráhy, na 95 %.

Čísla kol	Typ překážky	Počet vyjetí z dráhy	Úspěšnost [%]	Počet vyjetí z dráhy před překážkou	Úspěšnost [%]
1-25	Velká	0	100	2	92
26-50	Uprostřed	2	92	0	100
51-75	Vpravo	0	100	0	100
76-100	Vlevo	1	96	0	100
1-100	-	3	97	2	98

Tabulka 7.6: Výsledky testování na okruhu s překážkou z pohledu úspěšnosti držení se na dráze.

Čísla kol	Počet nárazů	Počet nárazů z boku	Počet zastavení s dotykem	Chyb celkem [Počet]	Úspěšnost celkem [%]	Počet zastavení	Počet zastavení [%]	Počet objetí	Počet objetí [%]	Úspěšnost jakékoliv reakce [%]	Úspěšnost očekávané reakce [%]
1-25	0	0	2	2	92	21	84	0	0	84	84
26-50	0	0	3	3	88	16	64	4	16	74	68
51-75	0	6	0	6	76	7	28	12	48	76	48
76-100	0	0	1	1	96	16	64	7	28	92	28
1-100	0	6	6	12	88	60	60	23	23	83	57

Tabulka 7.7: Výsledky testování na okruhu s překážkou z pohledu reakce na překážku.

Výsledky z pohledu úspěšnosti detekce a reakce na překážku jsou v tabulce 7.7. V případě velké překážky na dráze došlo celkem ke 2 případům, kdy autíčko zastavilo přesně ve chvíli, kdy se dotklo překážky. Tento jev je způsoben frekvencí snímání sestrojeného LIDARu, kdy může dojít k tomu, že překážka se skryje mezi jednotlivými měřícími cykly a k jejímu vyhodnocení dojde až bezprostředně před nárazem do ní. Obecně však při tomto

testu autíčko na překážku zareagovalo v 84 % případů, což je i procento případů, kdy došlo k předpokládané reakci a to zastavení před překážkou.

Větší výzvou pro autíčko byla v testech situace, kdy byla na středu dráhy umístěna překážka o rozměrech $20 \times 20 \times 20$ cm. V tomto případě došlo ke 3 případům zastavení s dotekem. V 16 případech autíčko před překážkou zastavilo, což představuje 64 % případů, a ve 4 případech (16 %) dokonce překážku zvládlo těsně objet, aniž by došlo ke kontaktu s překážkou. Celkově autíčko reagovalo na překážku v 80 % případů, přičemž očekávanou reakcí, kterou bylo zastavení, reagovalo v 64 % případů. Jelikož však v případě, kdy autíčko překážku objelo došlo ke zcela čistému objetí, dá se za celkový výsledek v tohoto testu považovat úspěšnost 80 %, jelikož ve 20 případech z 25 autíčko do překážky nenarazilo ani nevyjelo z dráhy.

Zajímavé výsledky přinesl třetí test, kdy byla překážka umístěna na pravou stranu dráhy, jehož cílem tedy bylo, aby autíčko překážku objelo zleva. Při tomto došlo k 6 případům, kdy autíčko začalo překážku objíždět, avšak poté donutil autíčko algoritmus detekující krajní čáry autíčko zatočil přímo do překážky z boku. Autíčko zastavilo v 7 případech, představujících 28 %, a ve zbylých 12, zastupujících 48 % případů, došlo k očekávanému objetí překážky. K reakci na překážku tedy došlo v 76 % případů, zatímco k očekávané reakci v podobě objetí překážky, došlo pouze ve 48 % případů.

Posledním testem byla překážka umístěná na levé straně dráhy, kterou bylo nutné objet zprava. Došlo na jeden případ, kdy autíčko zastavilo s dotykem s překážkou. V 16 případech došlo k zastavení autíčka před překážkou, což představuje 64 %. K tomuto jevu dochází vzhledem k velikosti zorného pole LIDARu a skutečnosti, že autíčko po vyjetí z pravé zatáčky směřuje přímo na překážku, která se tak řídícímu algoritmu jeví umístěná uprostřed dráhy, nikoliv na její levé straně. Pouze v 7 případech došlo na objetí překážky, což představuje úspěšnost pouhých 28 %. V případe, kdy bude jako úspěch počítán i případ, kdy autíčko před překážkou zastaví, tak je úspěšnost 92 %.

Testovací kola a jednotlivé typy překážek ukázaly silné a slabé stránky realizovaného systému detekce překážek. Systém je mnohem lépe schopný si poradit v situaci, kdy je cílem zastavení před překážkou. Obtížněji reaguje na situace, kdy je nutné překážku objet. Úspěšnost systému se však silně odvíjí od umístění překážky, pokud bude překážka umístěna na závěru dlouhé rovinky, bude úspěšnost jejího objetí vyšší, jelikož autíčko bude na dráze vycentrováno, zatímco v případě, kdy bude těsně za zatáčkou, bude úspěšnost nižší než v zaznamenaných testech. Důvodem k tomuto je jednak skutečnost, že autíčko zatáčí a senzor vzhledem ke svému zornému úhlu a aktuální pozici nemusí překážku za zatáčkou zaznamenat. Dalším důvodem pro nižší úspěšnost jsou pouhé 2 měřící cykly za 1 s, zatímco by bylo vhodně, aby k aktualizaci docházelo častěji, ovšem toto již nebylo možné realizovat s použitým servomotorem MG996R. Výsledná rychlost, přesnost a počet měření v jednom cyklu je kompromisem všech požadavků na měřící systém a synchronizaci mezi měřícími cykly, kdy pro zvýšení frekvence snímání byla v průběhu vývoje omezena šířka zorného pole, ale pro zachování synchronizace nebylo možné výrazněji zvýšit rychlost otáčení senzoru.

Kapitola 8

Závěr

Cílem této práce bylo nastudovat způsoby vytyčení závodní dráhy v různých soutěžích autonomních vozítek a následně se na jejich základě seznámit s principy detekce takto vytyčené dráhy. Na základě získaných poznatků navrhnout systém, který bude umožňovat autonomní jízdu modelu autíčka po vytyčené závodní dráze a bude umět rozpoznávat vodorovné značky na dráze. Tento systém posléze implementovat a otestovat na zvolené platformě. Navrhnout systém detekce fyzických překážek, umístěných na závodní dráze, a o tento systém následně rozšířit již implementovaný řídící systém a na závěr analyzovat výsledné vlastnosti realizovaného řešení.

V práci byly probrány rozdíly a podobnosti mezi jednotlivými soutěžemi a to včetně výzev, které některé ze specifik jednotlivých soutěží představují pro vývojáře autonomního autíčka. Na základě analýzy jednotlivých soutěží a způsobu vytyčení závodní dráhy byly stanoveny požadavky na detekci dráhy, na základě kterých bylo představeno několik možných technických řešení této detekce. V další části práce byly analyzovány požadavky na detekci fyzické překážky na dráze a byly prezentovány možnosti, jak se s detekcí takové překážky na dráze vypořádat. Na základě získaných poznatků byla zvolena základní platforma autíčka pro další vývoj včetně předpokládané řídící jednotky a obrazového snímače, na jejichž základě byl navržen systém schopný autonomní jízdy po dráze. Navazující rozsáhlá část práce byla věnována realizaci navrženého systému po softwarové i hardwarové stránce a komplikacím a výzvám, které byly spojeny se zvolenou platformou a změnám ve výsledné realizaci oproti původně avizovanému návrhu.

Pro doplnění implementovaného systému byl navržen senzorový systém pro detekci překážek, který při využití pouze jednoho ToF senzoru umožňuje pokrýt celý prostor dráhy před autíčkem. Navržený senzorový systém byl v další části práce realizován a již implementovaný řídící systém autíčka byl rozšířen o funkcionalitu obstarávající detekci překážek a byly zhodnoceny silné i slabé stránky výsledného řešení. Na závěr bylo výsledné řešení autonomního autíčka otestováno na různých typech závodní dráhy, při kterém byla kromě schopnosti držení se na dráze testována i schopnost detekce značek, překážek a reakce na tyto prvky.

Pokračování této práce je možné vidět v dalším zlepšení již implementovaných částí z hlediska jejich přesnosti a rychlosti zpracování veškerých vstupních dat s cílem zvýšit celkovou výkonnost systému a výslednou rychlost pohybu autíčka po závodní dráze. Z celého systému však zejména navržený senzorový systém pro detekci překážek skrývá dostatek potenciálu pro další experimentování a vývoj.

Literatura

[1] *AWS DeepRacer Track Design Templates* [online]. Amazon Web Services [cit. 2022-01-05]. Dostupné z: https://docs.aws.amazon.com/deepracer/latest/developerguide/deepracer-track-examples.html.

[2] *AWS DeepRacer—powered by AWS RoboMaker—helps individuals of all skill levels get hands-on experience with machine learning in interactive competitions* [online]. [cit. 2022-01-05]. Dostupné z: https://www.aboutamazon.com/news/aws/army-and-navy-teams-compete-in-first-annual-aws-deepracer-competition.

[3] *Baseus Bipow Digital Display Power bank 10000mAh 15W Black* [online]. alza.cz [cit. 2022-04-14]. Dostupné z: https://www.alza.cz/baseus-bipow-digital-display-power-bank-10000mah-15w-black-d6658497.htm#parametry.

[4] *BLDC motor outrunnerový 930 kV A2212/15T* [online]. ECLIPSERA s.r.o. [cit. 2022-01-17]. Dostupné z: https://dratek.cz/docs/produkty/0/117/1496220893.pdf.

[5] *BSS138 N-Channel Logic Level Enhancement Mode Field Effect Transistor* [online]. Adafruit [cit. 2022-04-28]. Dostupné z: https://cdn-shop.adafruit.com/datasheets/BSS138.pdf.

[6] *Build Your Physical Track for AWS DeepRacer* [online]. Amazon Web Services [cit. 2022-01-05]. Dostupné z: https://docs.aws.amazon.com/deepracer/latest/developerguide/deepracer-build-your-track.html.

[7] *DATASHEET Raspberry Pi 4 Model B* [online]. Raspberry Pi Foundation [cit. 2022-01-16]. Dostupné z: https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf.

[8] *DFRobot ROB0165 Smart Robot Brushless Motor Racing Car* [online]. Mouser Electronics, Inc. [cit. 2022-01-17]. Dostupné z: https://cz.mouser.com/new/dfrobot/dfrobot-rob0165-brushless-motor-racing-car/.

[9] *Dinogy Graphene 1500mAh 2S 65C* [online]. eMotors [cit. 2022-04-12]. Dostupné z: https://emotors.cz/cs/dinogy-graphene-65c/6896-dinogy-graphene-1000mah-3s-65c.html.

[10] *Eses ultrazvukový měřič vzdálenosti HC-04 pro jednodeskové počítače* [online]. ECLIPSERA s.r.o [cit. 2022-01-16]. Dostupné z: https://dratek.cz/docs/produkty/0/773/eses1500636000.pdf.

[11] *Formula Pi* [online]. Formula Pi [cit. 2022-01-05]. Dostupné z: https://www.formulapi.com/.

[12] *Getting Started with Jetson Nano 2GB Developer Kit* [online]. NVIDIA Corporation [cit. 2022-05-15]. Dostupné z: https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-2gb-devkit#intro.

[13] *IIC I2C Modulový driver servo motoru pro Arduino - PCA9685 16 kanálů 12-bit PWM* [online]. ECLIPSERA s.r.o. [cit. 2022-04-17]. Dostupné z: https://dratek.cz/arduino/1686-iic-i2c-modulovy-driver-servo-motoru-pro-arduino-pca9685-16-kanalu-12-bit-pwm.html?gclid=Cj0KCQjw0umSBhDrARIsAH7FCocUAA0S7P1tetrfZ6IYQ3zd-yH_xZsfdSl3UZij2TOAW8DyLgTSVBAaApiTEALw_wcB.

[14] *Introducing the Raspberry Pi Cameras* [online]. Raspberry Pi Foundation [cit. 2022-01-16]. Dostupné z: https://www.raspberrypi.com/documentation/accessories/camera.html.

[15] *Jetson Partner Supported Cameras* [online]. NVIDIA Corporation [cit. 2022-01-15]. Dostupné z: https://developer.nvidia.com/embedded/jetson-partner-supported-cameras?t1_max-resolution=4K.

[16] *JETSON STORE* [online]. NVIDIA Corporation [cit. 2022-01-15]. Dostupné z: https://www.nvidia.com/cs-cz/autonomous-machines/jetson-store/.

[17] *MG996R High Torque Metal Gear Dual Ball Bearing Servo* [online]. GM electronic, spol. s.r.o. [cit. 2022-01-17]. Dostupné z: https://www.gme.cz/data/attachments/dsh.772-293.1.pdf.

[18] *MicroWave Sensor SKU SEN0192* [online]. DFROBOT [cit. 2022-01-16]. Dostupné z: https://wiki.dfrobot.com/MicroWave_Sensor_SKU__SEN0192#target_7.

[19] *NVIDIA's DIY Autonomous Car Race* [online]. NVIDIA Corporation [cit. 2022-01-04]. Dostupné z: https://developer.nvidia.com/embedded/diy-ai-race.

[20] *The NXP Cup Official Rules* [online]. NXP [cit. 2022-01-05]. Dostupné z: https://community.nxp.com/t5/The-NXP-Cup-EMEA/NXP-Cup-Rules-2021-2022/ta-p/1351213.

[21] *NXP RDDRONE-FMUK66 FMU* [online]. PX4 Autopilot [cit. 2022-01-15]. Dostupné z: https://docs.px4.io/master/en/flight_controller/nxp_rddrone_fmuk66.html.

[22] *Pixy2 Overview* [online]. PixyCam [cit. 2022-01-15]. Dostupné z: https://docs.pixycam.com/wiki/doku.php?id=wiki:v2:overview.

[23] *Purchasing a NXP Cup Track* [online]. NXP [cit. 2022-01-05]. Dostupné z: https://community.nxp.com/t5/University-Programs-Knowledge/Purchasing-a-NXP-Cup-Track/ta-p/1121390.

[24] *Raspberry Pi 3 Model B* [online]. Raspberry Pi Foundation [cit. 2022-01-16]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/.

[25] *Raspberry Pi Camera Module 2* [online]. Raspberry Pi Foundation [cit. 2022-01-16]. Dostupné z: https://www.raspberrypi.com/products/camera-module-v2/.

[26] *Raspberry Pi High Quality Camera* [online]. Raspberry Pi Foundation [cit. 2022-01-16]. Dostupné z: https://www.raspberrypi.com/products/raspberry-pi-high-quality-camera/.

[27] *Road 4K Wallpapers* [online]. WallpaperAccess [cit. 2022-01-17]. Dostupné z: https://wallpaperaccess.com/road-4k.

[28] *RPLIDAR A1 Low Cost 360 Degree Laser Range Scanner* [online]. Shanghai Slamtec.Co.,Ltd [cit. 2022-01-17]. Dostupné z: http://bucket.download.slamtec.com/e9e096e9d9f30205d665260abe2cfb0c2dd62efa/LD108_SLAMTEC_rplidar_datasheet_A1M8_v1.0_en.pdf.

[29] *RPLiDAR A1M8 360° laserový scanner* [online]. RPishop.cz [cit. 2022-01-17]. Dostupné z: https://rpishop.cz/lidary/1631-rplidar-a1m8-360stupnovy-laserovy-scanner-kit-dosah-12m.html.

[30] *SONY [Product Brief] IMX 219 ver.1.0* [online]. Sony Corporation [cit. 2022-01-16]. Dostupné z: https://github.com/rellimmot/Sony-IMX219-Raspberry-Pi-V2-CMOS/blob/master/IMX219%20Product%20Brief.pdf.

[31] *TFmini-S LiDAR Module* [online]. Mouser Electronics, Inc. [cit. 2022-01-16]. Dostupné z: https://cz.mouser.com/datasheet/2/1099/Benewake_10152020_TFmini_S-1954051.pdf.

[32] *Ultrazvukový měřič vzdálenosti HC-SR04 pro jednodeskové počítače* [online]. ECLIPSERA s.r.o [cit. 2022-01-16]. Dostupné z: https://dratek.cz/arduino/846-eses-ultrazvukovy-meric-vzdalenosti-hc-04-pro-jednodeskove-pocitace.html?gclid=Cj0KCQiAoY-PBhCNARIsABcz7720MolQ9t5C_jwdo8Vf4_dhh3RLS9q3djZhD1YofNdRyhsyY6UUTTgaAh3WEALw_wcB.

[33] *What Is AWS DeepRacer?* [online]. Amazon Web Services [cit. 2022-01-05]. Dostupné z: https://docs.aws.amazon.com/deepracer/latest/developerguide/what-is-deepracer.html.

[34] Anderson, C. *DIY Autonomous Car Racing with NVIDIA Jetson* [online]. NVIDIA Developer [cit. 2022-01-04]. Dostupné z: https://www.youtube.com/watch?v=4B2ipB7n5Nk.

[35] Chen, V. C. *The Micro-Doppler Effect in Radar, Second Edition.* Artech House, 2019. 1–17 s. ISBN 1630815489, 9781630815486. Dostupné z: https://books.google.cz/books?id=SVCQDwAAQBAJ.

[36] Churchill, A. *Track 1 layout* [online]. Formula Pi, 2016 [cit. 2022-01-05]. Dostupné z: https://www.formulapi.com/track-1/layout.

[37] Peterson, Z. *Lane Recognition and Tracking with NVIDIA Jetson Nano* [online]. Altium Limited, 2020 [cit. 2022-05-15]. Dostupné z: https://resources.altium.com/p/lane-recognition-and-tracking-nvidia-jetson-nano.

[38] Purakkatt, A. *Building a lane detection system* [online]. Medium, 2021 [cit. 2022-05-15]. Dostupné z: https://medium.com/analytics-vidhya/building-a-lane-detection-system-f7a727c6694.

[39] Staff, H. *3D Scan a Room With This LIDAR Rig* [online]. Hackster.io, an Avnet Community [cit. 2022-01-17]. Dostupné z: https://www.hackster.io/news/3d-scan-a-room-with-this-lidar-rig-a630c1a65087.

[40] Steingart, V. *Systém pro autonomní řízení modelu autíčka na závodní dráze* [online]. Brno, 2020. [cit. 2022-01-17]. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Václav Šimek. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=232608.

[41] Zhang, W., Li, H., Yan, X. a Liu, Z. A Method of Recognizing Curve Direction Based on Hough Transform. In: *2016 9th International Symposium on Computational Intelligence and Design (ISCID)*. 2016, sv. 2, s. 3–6. DOI: 10.1109/ISCID.2016.2010.

Příloha A

Obsah přiloženého DVD

/text - zdrojové soubory tohoto dokumentu

/src - zdrojové soubory výsledného programu

/videa_z_testovani - videa ukazující jízdu autíčka na 5 typech testovací trati

/zaznamy_z_testovani - tabulky zachycující průběh testovacích kol

/fotografie_auticka - fotografie z přílohy B zachycující zapojení autíčka

readme.txt - popis programu, jeho ovládání a spouštění

xkatru00.pdf - tento dokument

Příloha B

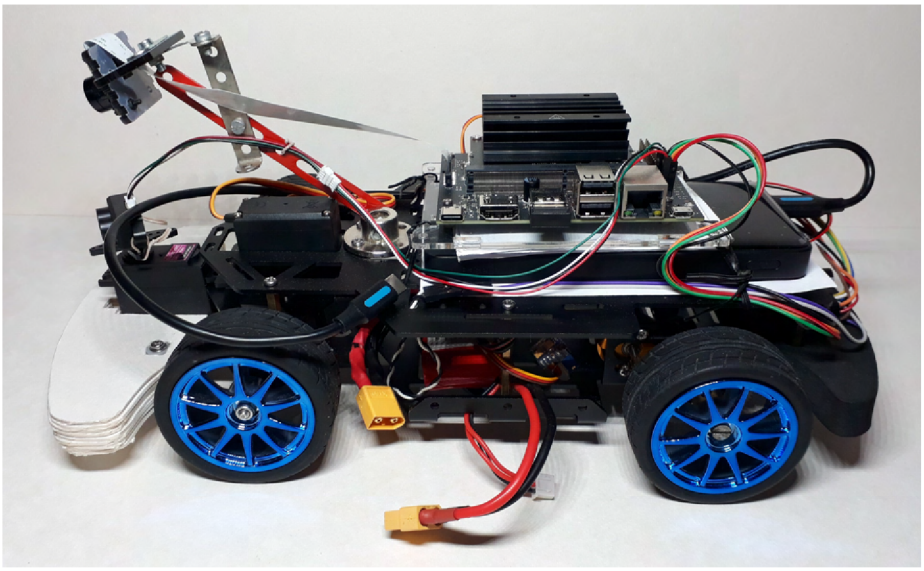
Spuštění a instalace

Následující text ve své první části (B.1) probírá, jakým způsobem je možné spustit řídící systém autíčka a ovládat jej. Druhá část textu (B.2) je věnována případné instalaci systému a veškerých náležitostí potřebných k běhu hlavního programu.

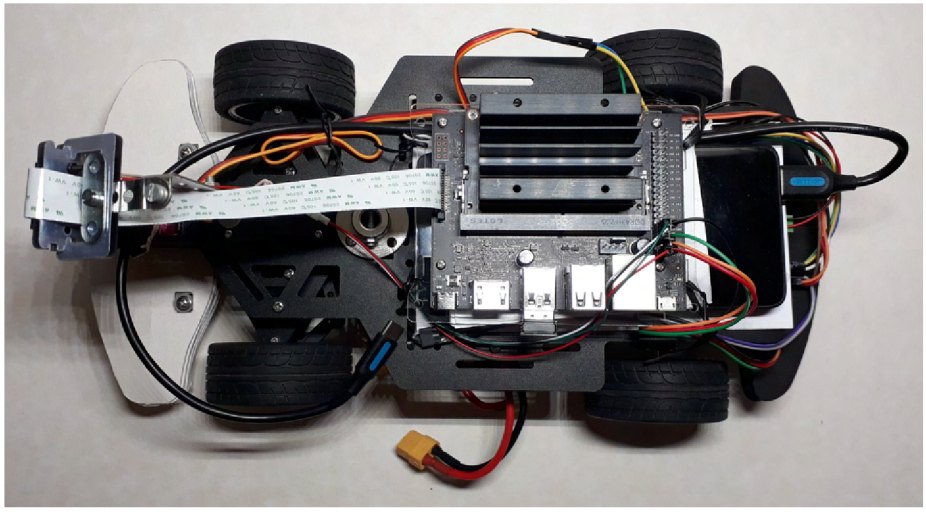
B.1 Spuštění systému

Pro úspěšné spuštění systému je zapotřebí zapojit veškeré jeho součásti podle schématu 7.2 popsaného v podkapitole 7.1, přičemž k připojení akumulátoru Dinogy Graphene a powerbanky Baseus Bipow by mělo dojít až ve chvíli, kdy je celý systém zapojený a má dojít k jeho spuštění (viz dále).

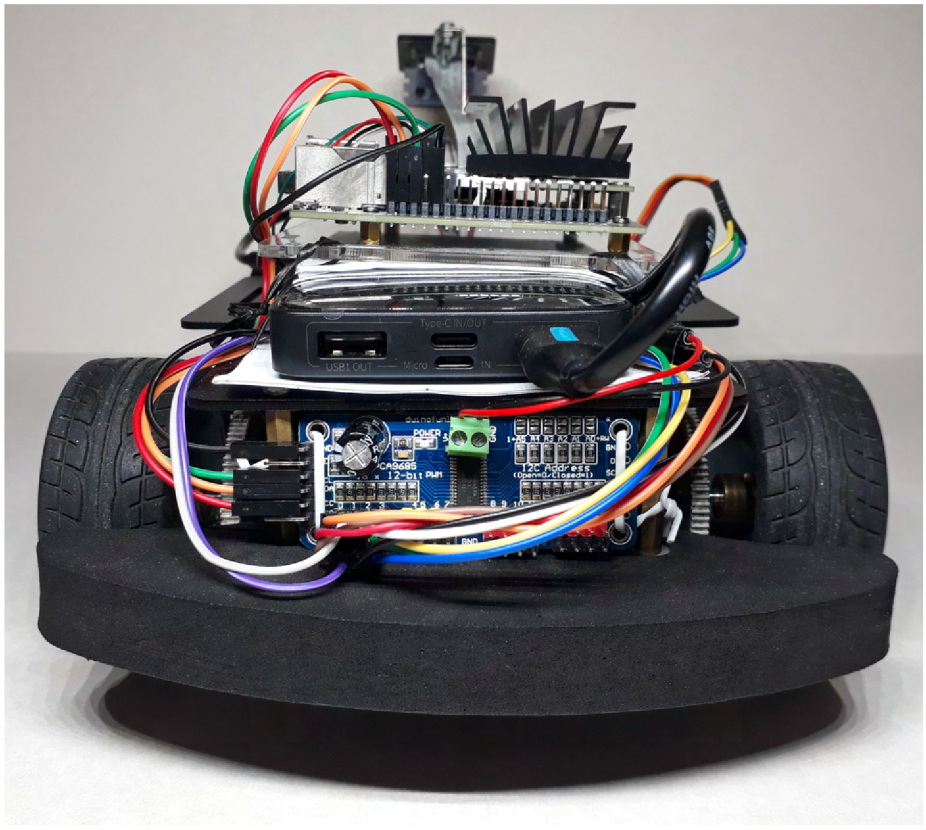
Na obrázku B.1 je pohled na zapojené komponenty autíčka z levého boku, na obrázku B.2 pohled shora, obrázek B.3 zachycuje zadní část autíčka, obrázek B.4 pravou stranu a na závěr obrázek B.5 představuje pohled zepředu.



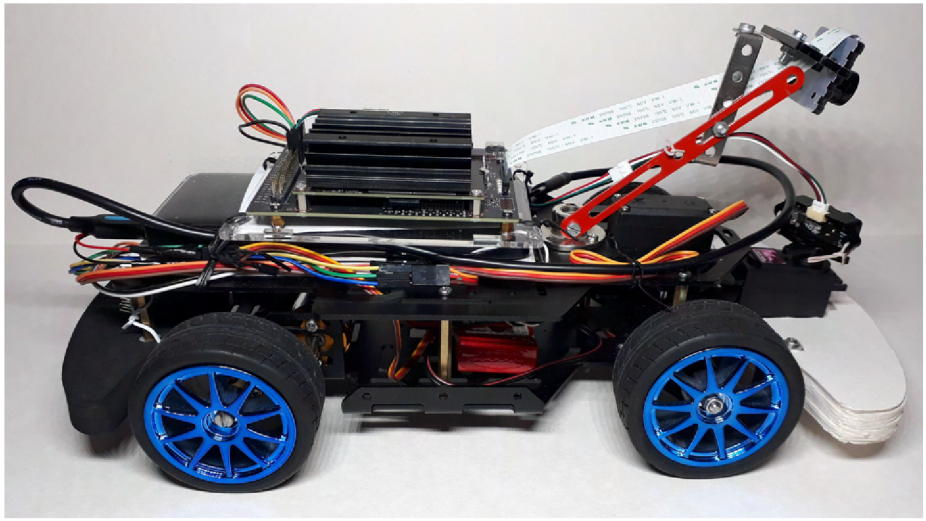
Obrázek B.1: Pohled na levý bok autíčka (galerie autora).



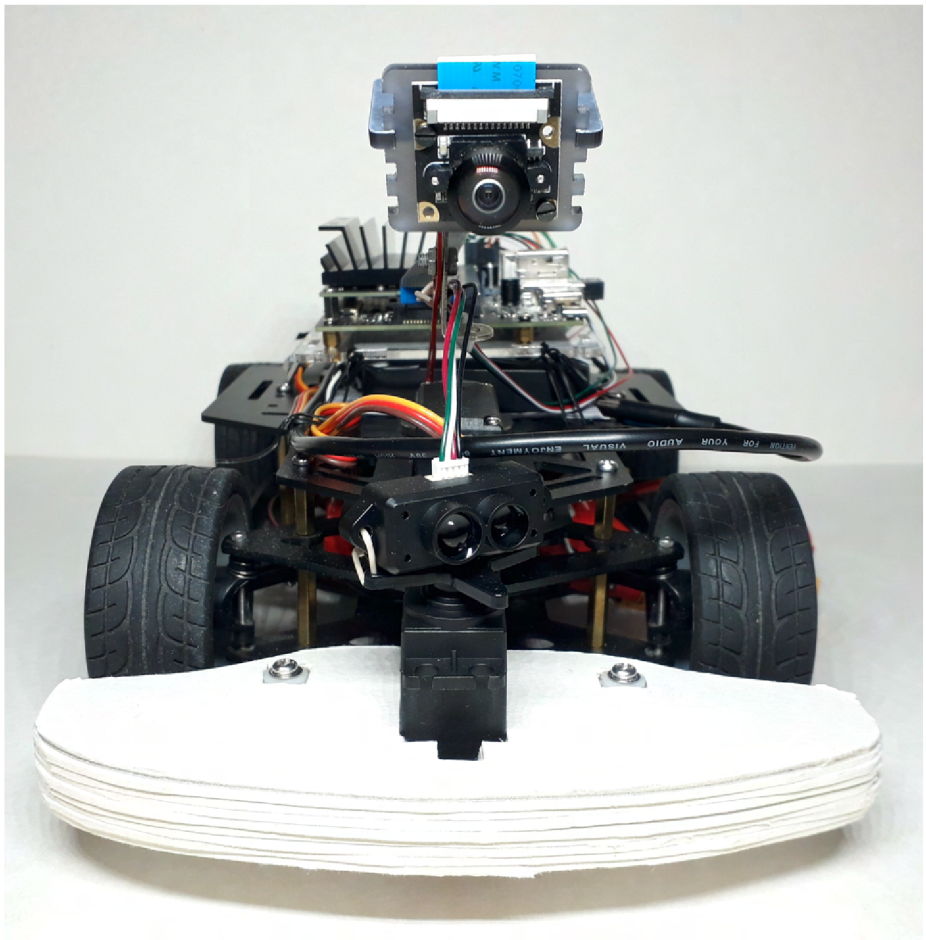
Obrázek B.2: Pohled na autíčko shora (galerie autora).



Obrázek B.3: Pohled na autíčko zezadu (galerie autora).



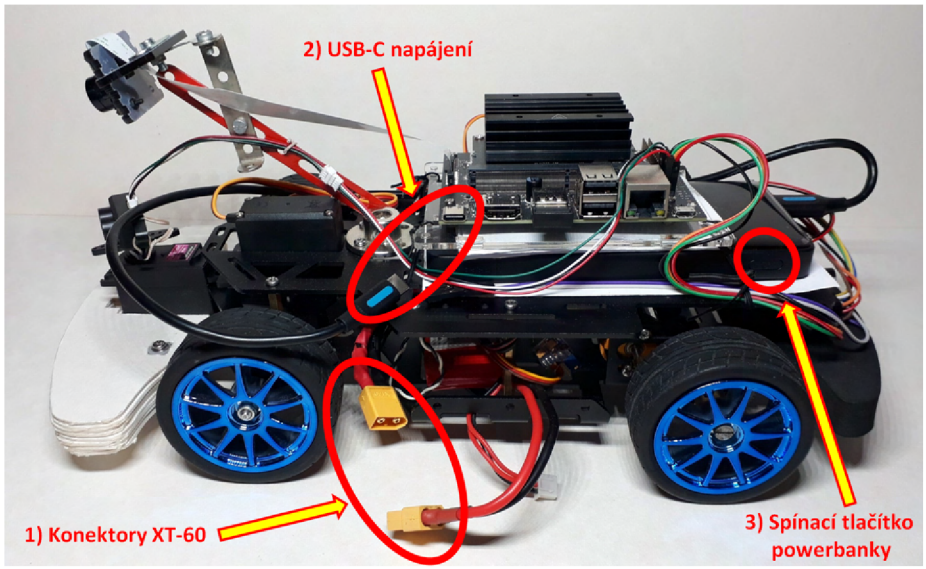
Obrázek B.4: Pohled na pravý bok autíčka (galerie autora).



Obrázek B.5: Pohled na autíčko zepředu (galerie autora).

Pro spuštění autíčka je, po jeho správném obvodovém zapojení, nutné připojit napájení, které se skládá ze dvou samostatných okruhů, blíže představených v podkapitole 5.1.

Jako první krok je nutné připojit akumulátor Dinogy Graphene pomocí žlutých konektorů XT-60, následně je možné zapojit napájení řídící desky Jetson Nano, realizované pomocí powerbanky Baseus Bipow. V jejím případě je nutné připojení na řídící desku realizovat prostřednictvím USB-C konektoru. Po připojení USB-C napájení do řídící desky by mělo dojít k jejímu spuštění. Pokud se tak nestane, tak musí být powerbanka ručně spuštěna jedním stisknutím zapínacího tlačítka. Pokud je powerbanka aktivní, tak dvojím stisknutím zapínacího tlačítka dojde k jejímu vypnutí a tím i k vypnutí řídící desky. Po vypnutí řídící desky je nutné, pro úplné vypnutí systému, odpojit akumulátor Dinogy Graphene opětovným rozpojením konektoru XT-60. Powerbanka může poté zůstat připojená k řídící desce, ale doporučeným postupem je rovněž i její fyzické odpojení. Na obrázku B.6 jsou označeny veškeré potřebné konektory a ovládací prvky pro připojení napájení.



Obrázek B.6: Zapojení napájecího systému autíčka.

Po tomto kroku by mělo dojít ke spuštění operačního systému řídící desky a následnému automatickému spuštění řídícího programu, doprovázenému několika pípnutími oznamujícími aktivaci ESC regulátorů zadních elektromotorů. Následně je možné autíčko ovládat těmito klávesami z bezdrátové klávesnice:

E - aktivace zadních elektromotorů, předního servomotoru a zahájení autonomní jízdy po závodní dráze

P - pozastavení jízdy, při kterém dojde k zastavení zadních motorů a natočení předního servomotoru do výchozí pozice

O - opětovné spuštění motorů a jízdy po dráze po zastavení (zastavení může být způsobeno stiskem klávesy **P**, nebo automaticky v případě, kdy autíčko detekuje překážku, na kterou zareaguje zastavením)

S - zastavení zadních elektromotorů, zatímco přední servomotor pokračuje v činnosti (mód určený pro vývoj a testování autíčka)

W - opětovná aktivace zadních elektromotorů po jejich zastavení klávesou **S**

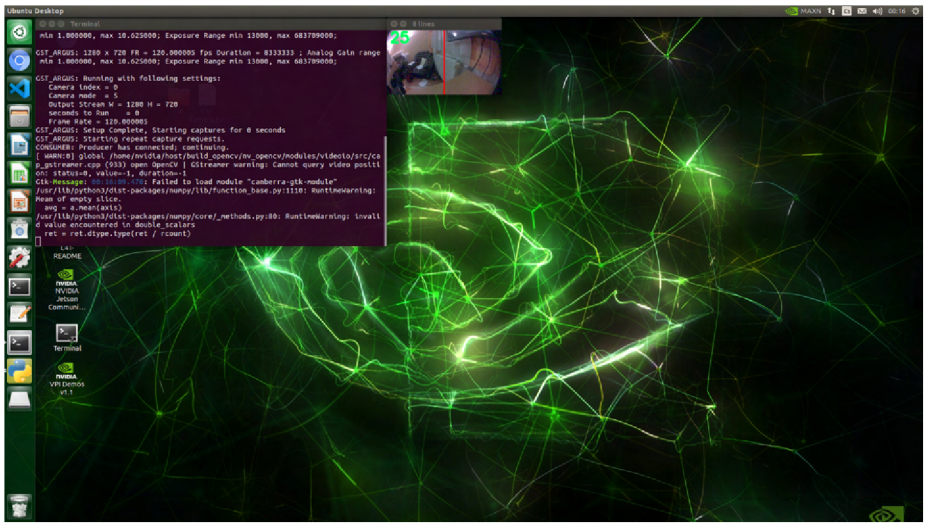
M - aktivace systému detekce překážek, po kterém autíčko začne reagovat na překážky na závodní dráze (stisk klávesy **M** je nutné provézt až poté, co byla stisknuta klávesa **E** pro aktivaci jízdního režimu, jelikož v opačném pořadí dojde k zastavení rotace senzoru, bez možnosti jeho opětovného spuštění, a je nutné restartovat řídící program)

N - držením klávesy **N** až do okamžiku, kdy senzor detekující překážky dojede do jedné ze dvou krajních poloh, dojde k jeho zastavení (následně je možné klávesou **M** provézt jeho opětovné spuštění)

Q - klávesou **Q** dojde k ukončení programu (opětovné spuštění je možné pouze ručně z prostředí operačního systému, nebo vypnutím a zapnutím napájení řídící desky)

Pro zjednodušení práce s autíčkem byly v operačním systému řídící desky provedeny některé změny, které po připojení napájení automaticky přihlásí uživatele do systému a následně spustí terminál, ve kterém dojde ke spuštění řídícího programu autíčka. Občas však nedojde ke zcela správnému provedení těchto operací a řídící systém autíčka nemusí být korektně spuštěn, případně při testování došlo v několika případech k situaci, kdy část programu obstarávající detekci dráhy a jízdu po ní funguje zcela v pořádku, ovšem následné spuštění LIDARu klávesou **M** neproběhne správně, kvůli chybě na sériovém rozhraní, a systém detekce překážek tak nefunguje. V takovém případě je možné vypnout powerbanku a opět ji zapnout, což způsobí restart systému.

Druhou možností, jak tento problém řešit, je připojit k řídící desce monitor přes přítomný HDMI port, jelikož operační systém platformy Jetson Nano disponuje grafickým uživatelským rozhraním (na obrázku B.7), a ukončit aktuálně běžící program klávesou **Q**, nebo v terminálu kombinací kláves **CTRL+C**. Následně je možné pouze otevřít nový terminál, ve kterém opět automaticky dojde ke spuštění řídícího programu autíčka.



Obrázek B.7: Snímek obrazovky uživatelského rozhraní operačního systému platformy Jetson Nano. Na snímku je možné vidět automaticky spuštěný terminál s řídícím programem autíčka a vedle terminálu se nachází malé okno s obrazovým výstupem ukazujícím detekované okrajové čáry (galerie autora).

Poslední možností je řídící program spustit zcela ručně a to otevřením terminálu a spuštěním řídícího programu následujícím příkazem (heslo pro `sudo` by nemělo být vyžadováno, ale je uvedeno v souboru `readme.txt`):

```
sudo python3 Desktop/DP/autonomous-car.py
```

B.2 Instalace systému

Pro případ zprovozňování řídícího programu autíčka na nepřipravené desce je z hardwarového hlediska nutné provézt zapojení celého systému podle schématu 7.2. Implementovaný program v programovacím jazyce Python byl vyvíjen a testován v jeho verzi s číslem 3.6. Program byl realizován na platformě nVidia Jetson Nano 2GB developer kit, na které byl nainstalován operační systém v následující verzi distribuované pro platformu Jetson Nano:

```
Ubuntu 18.04.6 LTS
```

Proces instalace operačního systému na platformu Jetson Nano je popsán v návodu poskytovaném společností nVidia [12]. Pro maximální zjednodušení instalace všech závislostí se ve složce `src` nachází, kromě zdrojového souboru `autonomous-car.py`, soubor s názvem `requirements.txt`, obsahující seznam veškerých knihoven a jejich verzí, potřebných pro správnou funkci programu, vypsaný níže:

```
keyboard==0.13.5
nanocamera==0.1.5
numpy==1.13.3
opencv_python==4.5.5.64
pyserial==3.5
smbus==1.1.post2
smbus_cffi==0.5.1
```

Pro automatické nainstalování těchto knihoven je možné využít následujícího příkazu:

```
pip3 install requirements.txt
```

Po tomto kroku je již možné spustit řídící program, v jeho umístění, pomocí příkazu:

```
sudo python3 autonomous-car.py
```