VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

AUTONOMNÍ ŘÍZENÍ AUTÍČKA S ADAPTACÍ NA PŘEDEM NEZNÁMÝ TVAR AUTODRÁHY

AUTONOMOUS SLOT-CAR DRIVING WITH ADAPTATION TO UNKNOWN TRACK SHAPE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MAREK NESVADBA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JOSEF STRNADEL, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Nesvadba Marek**
Program: Informační technologie
Název: **Autonomní řízení autíčka s adaptací na předem neznámý tvar autodráhy**
Autonomous Slot-Car Driving with Adaptation to Unknown Track Shape
Kategorie: Vestavěné systémy

Zadání:

1. Seznamte se s vlastnostmi dostupných autodráh. Zvolte autodráhu od konkrétního výrobce, s níž budete dále pracovat.
2. Z ročníků 2009-2012 soutěže Freescale Race Challenge zdokumentujte desky plošných spojů (DPS) navržené pro autonomní řízení autíček pomocí mikrokontrolérů (MCU). Popište principy související s řízením motoru a zpracováním dat z akcelerometru či dalších čidel využitelných pro řízení.
3. Zvolte/upravte/doplňte existující DPS (FRDM, ESP32/8266, Arduino atp.), popř. navrhněte/vyrobte vlastní DPS, vhodnou pro umístění do autíčka a pro jeho autonomní řízení. Danou DPS oživte a demonstrujte její schopnost ovládat pomocí MCU rychlost autíčka a rozpoznat rovinku/zatáčku.
4. Navrhněte datovou reprezentaci tvaru dráhy pro uložení v paměti přístupné z MCU.
5. Navrhněte a v MCU realizujte i) algoritmus autonomního mapování tvaru předem neznámé dráhy, ii) algoritmus schopný využít informaci o tvaru dráhy k projetí dráhy v co nejkratším čase.
6. Funkčnost algoritmů vhodně ověřte, demonstrujte a zhodnoťte.
7. Diskutujte možná vylepšení či rozšíření představeného řešení.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 4 zadání.

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

Vedoucí práce: **Strnadel Josef, Ing., Ph.D.**
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 25. října 2019

Abstrakt

Cílem této práce je návrh a implementace algoritmů pro mapování neznámého tvaru autodráhy a využití informací o tvaru autodráhy pro její projetí v co nejkratším čase. Algoritmus pro mapování rozdělí trať na několik sekcí (rovné úseky, zatáčky, brzdné zóny a výjezdy ze zatáček) a algoritmus pro jízdu následně využívá informaci o ujeté vzdálenosti a informací o tvaru trati pro nastavení rychlosti autíčka. Autíčko je schopno jet po autodráze pouze se znalostí délky dráhy co největší rychlostí bez toho, aby z dráhy vypadlo, což je potvrzeno experimenty.

Abstract

The goal of this bachelor thesis is design and implementation of slot car track mapping algorithm and algorithm that uses information about shape of the track to drive through it in the fastest time. Algorithm for mapping divides the track into sections (straights, corners, braking zones and corner exit zones) and algorithm for driving then uses information about driven distance and about shape of the track to set speed of the car. The car is able to drive around the track with only the information about the track length without crashing, which is confirmed by experiments.

Klíčová slova

autonomní řízení, autodráha, adaptace na neznámý tvar, arduino, akcelerometr, freescale race challenge

Keywords

autonomous driving, slotcar track, adaptation to unknown shape, arduino, accelerometer, freescale race challenge

Citace

NESVADBA, Marek. *Autonomní řízení autíčka s adaptací na předem neznámý tvar autodráhy.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Josef Strnadel, Ph.D.

Autonomní řízení autíčka s adaptací na předem neznámý tvar autodráhy

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Josefa Strnadela Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Marek Nesvadba
18. května 2020

Poděkování

Děkuji Ing. Josefu Strnadelovi Ph.D. za vedení mé bakalářské práce.

Obsah

1 Úvod **3**

2 Rešerše k zadanému problému **4**
2.1 Vlastnosti dostupných autodráh 4
2.2 Soutěž Freescale Race Challenge 6
2.2.1 Ročník 2009 7
2.2.2 Ročník 2010 8
2.2.3 Ročník 2011 8
2.2.4 Ročník 2012 9
2.2.5 Některá řešení použita v soutěži 9
2.3 Principy řízení motoru a zpracování dat z čidel 9
2.3.1 Řízení elektromotoru 11
2.3.2 Akcelerometr 12
2.3.3 Gyroskop 14
2.3.4 Hallův senzor 15
2.3.5 Optický rotační enkodér 16
2.3.6 Senzor rozpoznávající barvy 17
2.3.7 Videokamera 18

3 Rozbor, návrh a řešení problému **19**
3.1 Rozbor problému 19
3.2 Návrh řešení 22
3.2.1 Zvolené komponenty 22
3.2.2 Datová reprezentace tvaru dráhy 24
3.2.3 Algoritmus autonomního mapování tvaru předem neznámé dráhy . . 24
3.2.4 Algoritmus schopný využít informaci o tvaru dráhy k projetí dráhy v co nejkratším čase 25
3.3 Popis realizace 25
3.3.1 Zapojení komponent 25
3.3.2 Implementace 28
3.4 Další možné přístupy k fungování algoritmu 35

4 Experimenty a zhodnocení řešení **37**
4.1 Experimenty 37
4.1.1 Rozdíl mezi zaprášenou a čistou dráhou a pneumatikami 38
4.1.2 Vliv nastavení rychlostí v jednotlivých sekcích na čas 39
4.1.3 Vliv způsobu zmenšování chyby ujeté vzdálenosti na čas 40
4.1.4 Algoritmus rozlišující více prudkostí zatáček 40

4.1.5 Algoritmus využívající aktuální hodnoty akcelerace v zatáčkách . . . 41
4.1.6 Porovnání algoritmu s nejlepším nastavením s ostatními algoritmy . 41
4.1.7 Srovnání časů zajetých na dvou tratích se stejnou délkou ale jiným tvarem 43
4.2 Zhodnocení řešení 44

5 Závěr **45**

Literatura **46**

A Data naměřená v experimentech **49**
A.1 Rozdíl mezi zaprášenou a čistou dráhou a pneumatikami 49
A.2 Vliv nastavení rychlostí v jednotlivých sekcích na čas 50
A.3 Vliv způsobu zmenšování chyby ujeté vzdálenosti na čas 53
A.4 Algoritmus rozlišující více prudkostí zatáček 56
A.5 Algoritmus využívající aktuální hodnoty akcelerace v zatáčkách 59
A.6 Srovnání časů zajetých na dvou tratích se stejnou délkou ale jiným tvarem . 62

B Obsah přiloženého média **63**

Kapitola 1

Úvod

Tato práce se zabývá návrhem algoritmů, které budou autonomně mapovat neznámý tvar autodráhy a následně se snažit dráhu projet v co nejkratším čase s pomocí informací o jejím tvaru. O řízení se stará mikrokontroler a další součástky uvnitř autíčka. Zadání navazuje na soutěž freescale Race Challenge, ve které měli studenti vytvořit obdobné algoritmy. Autíčko si v prvním průjezdu neznámou dráhu pomocí senzorů zmapuje. Je potřeba získat co nejpřesnější data, podle kterých bude algoritmus v dalších kolech řídit rychlost autíčka. Dále se autíčko musí vypořádat i s detekováním ujetého kola.

Cílem této práce je tedy vybrat vhodnou elektroniku pro řízení autíčka a navrhnout její zapojení. Dále navrhnout a implementovat algoritmy pro autonomní mapování tvaru dráhy a použití těchto dat pro jízdu po dráze.

Toto zadání jsem si vybral, protože se týká programování, umělé inteligence a automobilů, a to jsou všechno témata, která mě zajímají.

V kapitole 2 se práce věnuje vlastnostem autodráh, soutěži Freescale Race Challenge a principům řízení motorů a popisu čidel, která lze v autíčku využít. Kapitola 3 se zabývá rozborem problému, zvolením vhodné elektroniky a návrhem algoritmů pro mapování dráhy a řízení. Dále se zabývá realizací návrhu a implementací algoritmů. Kapitola 4 obsahuje experimenty a zhodnocení řešení.

Kapitola 2

Rešerše k zadanému problému

V této části práce jsou popsány vlastnosti autodráh, soutěž Freescale Race Challenge a její jednotlivé ročníky. Dále jsou popsány základní principy funkce a řízení elektromotorů a principy snímačů, které se dají využít při řešení problému autonomní jízdy po autodráze.

2.1 Vlastnosti dostupných autodráh

Autodráhy slouží většinou jako hračky pro 2 a více osob. Jejich základní princip je možnost ovládat elektrické autíčko, které jezdí po dílech sestavených do různých tvarů okruhu. Autíčka zatáčí pomocí vodící drážky a elektřinu sbírají z autodráhy pomocí sběracích kartáčků. Ovládání autíčka probíhá pomocí ovladače, který v nejjednodušším případě pouze mění velikost napětí v dané dráze. Některé autodráhy mají i funkci, která udržuje autíčku v jedné dráze konstantní rychlost a osoba ovládající druhé autíčko s ním může závodit.

Napájení

Starší autodráhy používají pro napájení stejnosměrné napětí. Ovladač reguluje pomocí potenciometru velikost napětí a tím i rychlost autíčka. Hodnoty napájecího napětí těchto autodráh jsou nejčastěji 12 nebo 14V.

Moderní autodráhy používají nejčastěji napětí 14.8V, které je v drahách konstantní. Autíčka jsou digitální a jejich rychlost nezávisí na napájecím napětí. Díky tomuto způsobu napájení může v jedné dráze jet nezávisle na sobě i několik autíček, která se můžou na k tomu určených místech předjíždět nebo měnit dráhy.

K napájení celé autodráhy slouží napájecí díl, do kterého se dají vložit baterie a nebo je možnost k němu připojit transformátor.

Způsob ovládání

Starší autodráhy používají analogové ovládání. Ovladač mění napětí v příslušné dráze a autíčko podle toho mění rychlost. Na takové autodráze může jet v každé dráze pouze jedno autíčko, a možnosti výhybek, přejíždění mezi dráhami a podobně jsou omezeny.

Modernější autodráhy používají digitální ovládání. To funguje tak, že ovladač komunikuje s dekodérem v jemu přiřazenému autíčku pomocí signálů, které jsou přenášeny přes kontakty v dráze a nebo bezdrátově. Díky tomuto způsobu ovládání jsou autíčka nezávislá na trati, po které jedou (a může jich jezdit po jedné trati více nezávisle na sobě), uchovávají si informaci o stavu paliva, opotřebování pneumatik a dá se jim nastavit například i

maximální rychlost. Digitální ovladače mohou mít i tlačítka pro změnu dráhy na nejbližší výhybce nebo například tlačítko turbo, které slouží pro krátkodobé zrychlení autíčka.

Velikost

Autodráhy mají různé měřítka (většinou 1:18, 1:24, 1:32, 1:43) od kterých se odvíjí rozteč drah a jejich celková velikost.

Autíčka mají také různá měřítka a každá velikost je vhodná pro jinou velikost autodráhy. Například autodráha Carrera Evolution má měřítko dílů 1:24 a měřítko autíček 1:32 [9].

Druhy dílů

Za základní díly se dají považovat rovné díly a zatáčky. Dále existují zúžení, křížení, výhybky, klopené zatáčky, lopingy a další speciální díly.

Rovné díly se vyrábí v různých délkách, aby bylo možné sestavit libovolné tvary dráhy. Zatáčky se vyrábí v několika poloměrech (například 1/90, 1/60, 1/45).

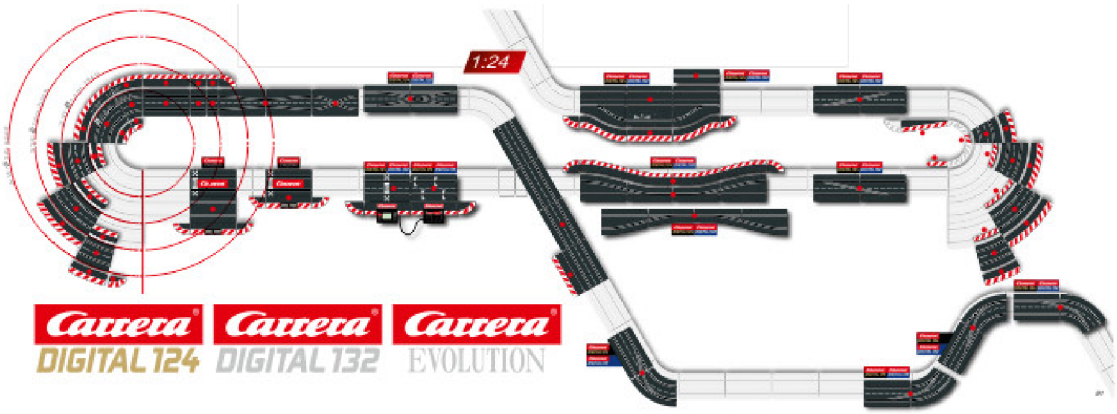
Díly se zúžením nebo křížením lze použít pro sestavení zajímavějšího okruhu, na kterém jde například blokovat ostatní autíčka a je potřeba si dobře načasovat průjezd těmito díly.

Výhybky slouží k přejíždění z dráhy na dráhu. Po stisknutí tlačítka pro změnu dráhy na ovladači začne v autíčku svítit infra červená LED dioda, kterou když zaznamená senzor v autodráze tak přepne výhybku a auto změní dráhu.

Díly autodráhy se většinou spojují mechanickými svorkami nebo mají speciální tvar, který zapadne do dalších dílů a tím je zajištěno, že se nerozpojí. Obsahují i vodivé kontakty pro elektrické propojení jednotlivých drah.

Díly se vyrábí v různých barvách simulujících různé povrchy tratě (asfalt, štěrk, sníh). Existují i různé mantinely, podpěry, rozšířená krajnice atd.

Přehled dílů několika autodráh firmy Carrera je na obrázku 2.1.



Obrázek 2.1: Některé díly autodráh firmy Carrera [10].

Druhy autíček

Autíčka mají různá měřítka (např 1:24) a vzhled. Vyrábí se vše od fiktivních modelů aut, přes závodní speciály (někdy i oficiálně licencované) až po formule a rallye auta. Model autíčka použitý této v práci je na obrázku 2.2.

Principy fungování autíček jsou ale u všech modelů velmi podobné. Jednou z nejdůležitějších částí je systém pro sběr elektřiny z autodráhových dílů. K tomu jsou použity

vodivé kartáčky, které se dotýkají vodivých částí autodráhy. Autíčka potřebují minimálně dva kartáčky, ale moderní mají obvykle 4. Vodivé kartáčky jsou většinou připevněny na otočeném dílu, na kterém je vodící lišta (nebo kolík), která drží autíčko v drážce autodráhy a tím autíčko řídí. Některé autíčka jsou vybaveny magnety, které zvyšují přilnavost autíčka k trati a snižují šanci vypadnutí autíčka z dráhy.

Druhou důležitou částí je elektrický motorek. Pokud je autíčko analogové, tak vedou vodiče od sběracích kartáčků přímo na svorky motorku a nic dalšího se v autíčku většinou nenachází. Digitální autíčka obsahují i řídící elektroniku, která podle signálů z ovladače řídí rychlost motorku. Motorek je spojen mechanickým převodem přímo se zadními koly autíčka.

Obrázek 2.2: Model autíčka, které je použito v této práci [8].

Příslušenství

Moderní autodráhy mají možnost využívat například časomíru i s měřením mezičasů, počítadlo ujetých kol nebo třeba díl, na kterém se provádí zastávka v boxech, která simuluje doplnění paliva v autíčku. Existují také moduly pro propojení autodráhy s počítačem nebo aplikací v chytrém telefonu. Nejmodernější digitální autíčka lze ovládat i z telefonu.

Carrera Evolution

V této bakalářské práci bude využívána autodráha Carrera Evolution, která byla používána i v soutěži Freescale Race Challenge. Je to analogová autodráha, v měřítku 1:24 s autíčky v měřítku 1:32 [9]. Napájecí napětí je 14.8V [11].

2.2 Soutěž Freescale Race Challenge

Informace v této sekci jsou čerpány z původních článků o soutěži [4], [5], [6], [7].

Freescale Race Challenge byla soutěž pořádaná pro studenty několika vysokých škol firmou Freescale. Cílem bylo, aby každý zúčastněný tým vytvořil algoritmus, který bude ovládat rychlost autíčka na autodráze, a snažit se zajet kolo v co nejkratším čase.

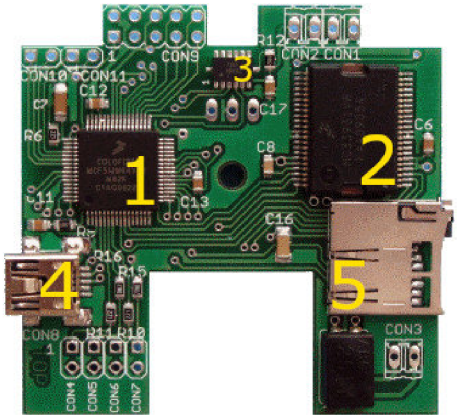
Každý tým dostal stejné autíčko, plošný spoj (verze z roku 2011 na obrázku 2.3), součástky a CD s dokumentací a ukázkovým programem. Univerzity dostaly sadu autodráhových dílů, časomíru, speciální díl s optickými snímači, ukázkové příklady použití a dokumentaci jednotlivých součástek. Plošný spoj obsahoval i expanzní konektor, který studentům umožňuje přidat další prvku. V ročníku 2011 studenti dostali desku již osazenou součástkami.

Po vývojové fázi, která trvala přibližně 4 měsíce bylo na každé univerzitě uspořádáno univerzitní finále. Několik nejlepší týmů z každé univerzity poté postoupilo do velkého finále které se konalo v Rožnově pod Radhoštěm.

Pravidla byla jednoduchá. Tvar dráhy byl předem neznámý, soutěžící pouze věděli, že délka bude v rozsahu 10 až 16 metrů a trať bude poskládána ze specifikovaných dílů.

Autíčka nesměla obsahovat magnety ani dálkové ovládání a nemohla být upravena tak, aby cokoliv přesahovalo vnější rozměry karoserie. Ostatní úpravy byly povoleny.

Před závodem dostaly týmy novou sadu pneumatik a autíčko mělo jeden seznamovací průjezd, při kterém mělo pomocí dat z akcelerometru zjistit tvar dráhy. Poté mělo projet 10 + 10 kol (v pravé a levé dráze) v co nejlepším čase. V ročníku 2010 se navíc ještě konal vyřazovací turnaj, ve kterém závodily vždy 2 autíčka proti sobě na souměrné dráze.



Obrázek 2.3: Osazená deska plošných spojů z ročníku 2011 soutěže Freescale Race Challenge [6]. Součástka 1 je mikrokontroler MCF51JM64VLH, 2 je H-můstek MC33931VW, 3 je akcelerometr MMA7361LR1, 4 je mini-USB konektor a 5 je slot pro mikro SD kartu.

2.2.1 Ročník 2009

V roce 2009 [4] se konal první ročník soutěže. Autodráhu dodala firma FARO, a elektronika byla následující:

Mikrokontroler

Freescale S08JM32 [16] je 8 bitový mikrokontroler pracující na frekvenci 48MHz. Na čipu obsahuje až 60kB Flash paměti a až 4kB RAM paměti.

Byl použit pro provádění algoritmu řízení rychlosti autíčka na základě údajů z akcelerometru.

Akcelerometr

MMA7361 [14] je akcelerometr pracující ve 3 osách s možností nastavení rozsahu, detekce volného pádu, teplotní kompenzací a dalšími funkcemi. Pracuje s napájecím napětím 2.2 - 3.6 V a s proudem 400 μA (3 μA v režimu spánku). Má dva nastavitelné rozsahy akcelerace, 1.5 a 6 g. Výstupní signál akcelerometru je hodnota napětí pro každou osu, která má v

klidovém stavu velikost poloviny napájecího napětí. Při kladném zrychlení se napětí zvyšuje a při záporném zrychlení se snižuje.

Akcelerometr byl používán pro snímání zrychlení působících na autíčko a jako hlavní zdroj informací o jízdě autíčka.

H-můstek

MC33887 [20] je H-můstek pro ovládání DC motorů do maximálního proudu 5A. Pracuje v rozpětí napájecího napětí 5 - 28V. Je kompatibilní s TTL nebo CMOS logickými úrovněmi ovládacího napětí.

H-můstek byl používán pro ovládání elektromotoru autíčka.

Paměť

EEPROM 24AA512 [19] je pamět s kapacitou 512Kb komunikující přes sériovou I2C sběrnici. Maximální hodinová frekvence je 400kHz. Paměť byla použita pro ukládání naměřených dat a následném zobrazení v počítači.

2.2.2 Ročník 2010

V roce 2010 [5] se konal druhý ročník soutěže s mírně upravenými pravidly. Navíc se konal vyřazovací turnaj. Autodráhu nyní dodává firma Carrera kvůli rozšíření soutěže i do Rumunska. Autíčko se nově programuje přes USB a místo EEPROM obsahuje slot na mikro SD kartu.

Mikrokontroler

MCF51JM [15] je 32 bitový mikrokontroler s jádrem V1 ColdFire pracujícím na frekvenci až 50.33 MHz při napájecím napětí 2.7 - 5.5V. Na čipu obsahuje až 128KB Flash paměti a až 16KB RAM paměti.

Akcelerometr

Akcelerometr MMA7361[14] je stejný jako v předchozím ročníku.

H-můstek

MC33931 [21] je h-můstek, který opět slouží k ovládání rychlosti motoru autíčka.

Paměť

Jako paměť pro uložení za jízdy zaznamenaných dat slouží mikro SD karta. Díky tomu je usnadněno čtení dat počítačem a není tolik omezena velikost ukládaných dat.

2.2.3 Ročník 2011

V roce 2011 [6] proběhly minimální změny v řídící elektronice autíček, a dalo se soutěžit i s autíčky z předchozího ročníku. Mezi organizační změny patřil například předem známý tvar dráhy pro vyřazovací turnaj nebo možnost trénovat v prostředí, ve kterém se pojede finální závod.

2.2.4 Ročník 2012

Ročník 2012 [7] byl opět velmi podobný předchozímu ročníku, s několika organizačními změnami. Novinkou bylo zvětšení maximální povolené hmotnosti autíček pokud používaly kameru. V tomto roce už firma Freescale pořádala i celosvětovou soutěž Freescale Cup a byl to poslední ročník Freescale Race Challenge.

2.2.5 Některá řešení použita v soutěži

V následujícím textu jsou stručně popsána některá řešení použitá v soutěží Freescale Race Challenge. Informace jsou čerpány z popisu řešení od některých týmů zveřejněných ve článcích o soutěži [6].

Týmy většinou do autíčka přidaly inkrementální rotační enkodér (popsán v části 2.3.5). Senzory fungovaly na principu odrážení světelného paprsku o část (nebo několik částí) kola autíčka. Signál ze senzoru byl zpracován mikrokontrolerem. Jeden způsob řešení byl převádět hodnotu proudu na digitální hodnotu a s tou následně pracovat. Jiný způsob zpracování signálu byl takový, že signál vyvolával přerušení. Z těchto hodnot následně algoritmus vypočítal ujetou vzdálenost nebo aktuální rychlost autíčka.

Elektronika v soutěžních autíčkách umožňovala i měření napětí trati. To se dalo využít pro detekci křížení v autodráze. Informaci o křížení týmy využívaly například pro synchronizaci měření ujeté vzdálenosti.

Mapování trati probíhalo většinou tak, že autíčko jelo konstantní rychlostí. Po detekci prvního křížení trati začal algoritmus detekovat zatáčky podle hodnot akcelerace. Datová reprezentace tvaru trati se většinou skládala z pole úseků trati, u kterých byla uložena informace o začátku, konci, někdy i délce a maximální rychlosti úseku.

Po třetím projetí křížení trati se algoritmus přepnul do závodního režimu. V něm využíval informace o tvaru dráhy a podle úseku dráhy na kterém se zrovna nacházel měnil rychlost autíčka. Na rovných úsecích se snažil vyvinout co největší rychlost a podle délky rovného úseku a typu následující zatáčky brzdit tak, aby zatáčku projel bezpečnou rychlostí.

2.3 Principy řízení motoru a zpracování dat z čidel

Některé informace v této části převzaty z [28]. Elektromotor funguje na principu využití silových účinků magnetického pole. Na vodič, ve kterém protéká proud, jenž se nachází v magnetickém poli, působí síla úměrná velikosti magnetického pole a proudu protékajícího vodičem.

Motor je složen ze dvou částí, statoru a rotoru. Rotor se většinou otáčí uvnitř statoru. Magnetické pole může být vytvářeno permanentním magnetem, nebo elektromagnetem.

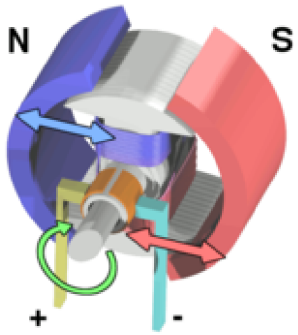
Elektromotory jsou nejčastěji rozdělovány na stejnosměrné, střídavé a krokové motory.

Stejnosměrné elektromotory

Stejnosměrné motory mají obvykle permanentní magnet ve statoru a cívku (elektromagnet) v rotoru. Pro přenos elektrické energie na točící se rotor je obvykle použit komutátor.

Proud protékající cívkou rotoru vytváří magnetické pole, které se pak vzájemně odpuzuje se stejným pólem magnetického pole tvořeného permanentním magnetem ve statoru a tím se motor otáčí (ilustrace na obrázku 2.4).

Otáčky stejnosměrného elektromotoru jsou řízeny velikostí napětí v cívce a směr otáčení lze změnit přepólováním napětí.



Obrázek 2.4: Princip stejnosměrného elektromotoru se statorem z permanentního magnetu a komutátorem [28].

Střídavé elektromotory

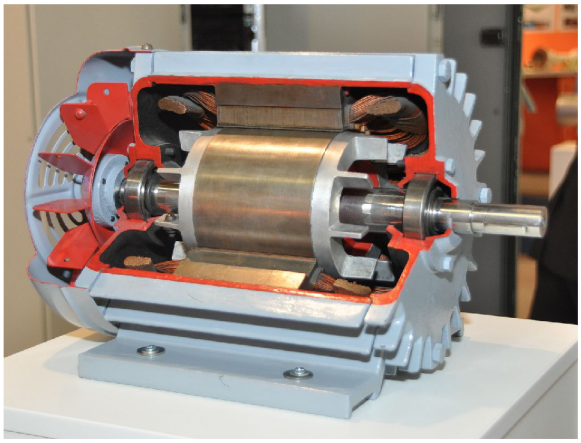
Střídavé motory jsou obvykle třífázové. Mají statorové vinutí, které je složeno ze 3 cívek. Cívky jsou zapojeny do hvězdy nebo trojúhelníku a podle potřeby protéká dvěma ze tří cívek proud, který vytváří magnetické pole (ilustrační obrázek 2.5). Méně výkonné motory mohou být i jednofázové.

Rotor může být vyroben jako kovová klec (kotva nakrátko) nebo jako další 3 cívky. Otáčející se magnetické pole tvořené statorem indukuje do rotoru elektrický proud a na rotor opět působí síla působící na vodič, kterým protéká proud v magnetickém poli.

Střídavé motory jsou většinou rozdělovány na synchronní a asynchronní. U synchronních motorů je frekvence otáček rotoru stejná jako frekvence otáček magnetického pole generovaného statorem. Rotor musí obsahovat permanentní magnet nebo elektromagnet napájený stejnosměrným napětím.

U asynchronních motorů se rotor otáčí pomaleji, a rozdíl mezi otáčkami rotoru a magnetického pole generovaného statorem se nazývá skluz. Velikost skluzu závisí na zatížení motoru. Tyto motory mají jednoduchou konstrukci, protože nepotřebují mechanismus přenosu elektrické energie na otáčející se rotor.

Pro řízení střídavého elektromotoru je většinou potřeba střídač s proměnnou frekvencí, kterou je regulována rychlost otáček motoru.



Obrázek 2.5: Průřez střídavým elektrickým motorem [29].

Krokové elektromotory

Krokové motory jsou výjimečné díky možnosti přesného polohování. Rotor se otáčí nespojitě po určitých úhlech závislých na konstrukci motoru. Krokové motory jsou používány hlavně ve speciálních aplikacích, kde je potřeba přesná informace o poloze (natočení) motoru.

Vlastnosti motorů používaných v autíčkách na autodráhu

V autíčkách na autodráhu jsou používány stejnosměrné elektromotory. Různé typy autodráh používají různé motory. Maximální kroutící momenty různých motorů se pohybují od 0,13 Ncm do 1,7 Ncm u speciálních velice výkonných motorů. Volnoběžné otáčky jednotlivých motorů jsou v rozsahu od 24000 ot./min až do 56000 ot./min. Otáčky během zatížení bývají většinou přibližně poloviční. Proud při maximálním zatížení motoru se pohybuje u jednotlivých typů od 140 mA do 1,7 A.

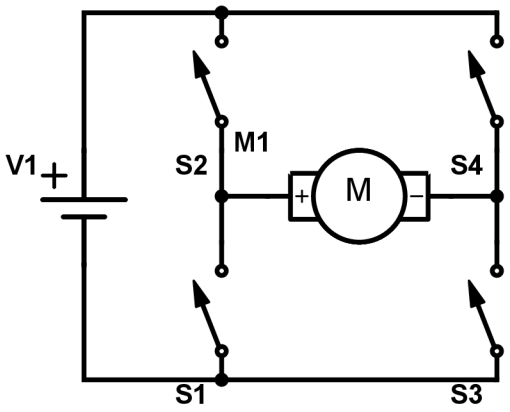
2.3.1 Řízení elektromotoru

V této jsou popsány základní principy řízení stejnosměrných elektromotorů, protože jsou používány v autíčkách na autodráhu.

Otáčky a směr otáčení elektromotoru závisí na velikosti a polaritě napětí. Pro ovládání otáček elektromotoru je tedy potřeba měnit velikost napětí na jeho svorkách. Toho lze docílit několika způsoby.

Nejjednodušší způsob změny napětí je například použití potenciometru zapojeného jako napěťový dělič. Tento způsob ale není v praxi moc použitelný.

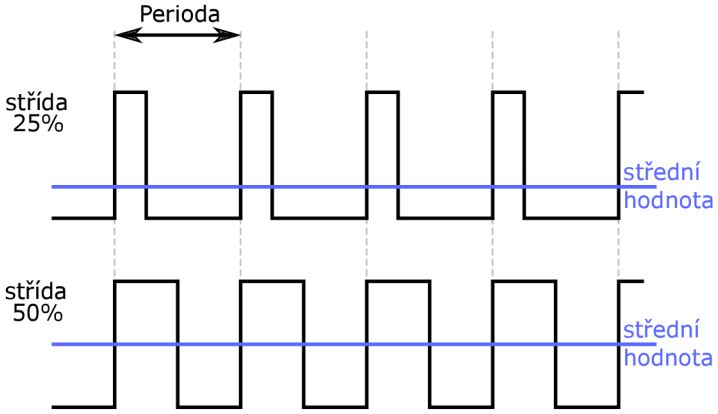
V praxi je často pro používán řízení motoru používán H-můstek. Skládá se ze 2 párů spínacích prvků (např. MOSFET tranzistory nebo relé) zapojených tak, že při sepnutí jednoho páru teče proud jedním směrem, a při sepnutí druhého páru teče proud opačným směrem (obrázek 2.6). Tím lze jednoduše měnit směr otáček motoru. Pokud jsou jako spínací prvky v H-můstku použity tranzistory, tak lze použít k řízení otáček motoru pulzní šířkovou modulaci (PWM). H-můstek lze snadno vyrobit nebo koupit ve formě integrovaného obvodu. Maximální proud můstkem a další parametry jsou různé, a závisí na fyzické konstrukci můstku.



Obrázek 2.6: Zapojení H-můstku pro ovládání motoru. Jako spínací prvky jsou pro znázornění použity vypínače.

Pulzně šířková modulace

Pulzně šířková modulace[1] funguje na principu změn poměru doby mezi napětím v logické 1 a logické 0. Výstupem je obdélníkový signál s různou střídou (poměr časů v napěťových úrovních log. 0 a 1). Střední hodnota takového signálu je pak závislá na střídě (obrázek 2.7). Díky tomu lze generovat různé úrovně napětí i například na digitálním pinu mikrokontroleru, který není vybaven D/A převodníkem. Pulzně šířková modulace se často využívá pro řízení rychlosti otáček elektromotorů, jasu LED diod nebo žárovek a podobně.



Obrázek 2.7: Ukázka dvou PWM signálů s různou střídou.

2.3.2 Akcelerometr

Akcelerometr je zařízení měřící zrychlení. Základní princip akcelerometru lze popsat jako závaží na pružině, které se působením zrychlení vychyluje ze středové polohy a tento pohyb je měřen a převáděn na hodnotu akcelerace. Toto závaží se u akcelerometrů nazývá seismická hmota. Akcelerometry jsou většinou piezoelektrické, piezorezistivní nebo kapacitní, podle způsobu jakým převádí mechanický pohyb závaží na elektrické napětí.

Moderní akcelerometry jsou nejčastěji vyrobeny jako mikro-elektro-mechanické systémy (MEMS). Takto jsou označovány miniaturní elektronická zařízení s mechanickými pohybujícími se částmi (obrázek 2.8).

Piezoelektrické akcelerometry

Vychýlená seismická hmota působí na destičku z piezoelektrického materiálu a tím generuje náboj odpovídající mechanickému namáhání, který odpovídá naměřenému zrychlení.

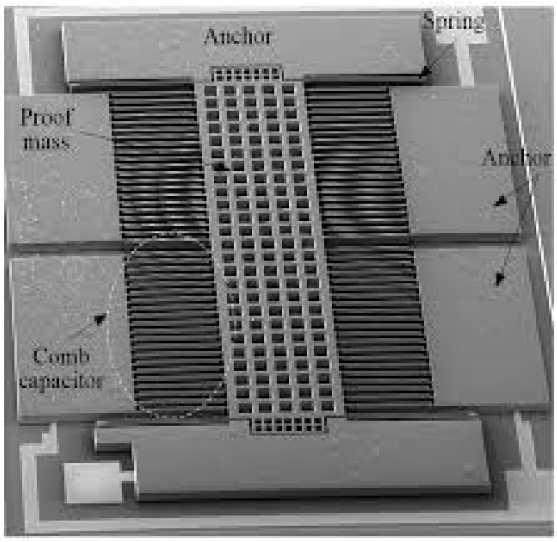
Piezorezistivní akcelerometry

Piezorezistor je integrován do pružiny, na které drží seismická hmota, detekuje její deformaci a tím mění svůj odpor, který je následně měřen a převeden na zrychlení.

Kapacitní akcelerometry

Kapacita mezi pevnými destičkami a destičkami připojenými k seismické hmotě odpovídá výchylce hmoty a tím i zrychlení působící na akcelerometr.

[1] Více na: https://en.wikipedia.org/wiki/Pulse-width_modulation

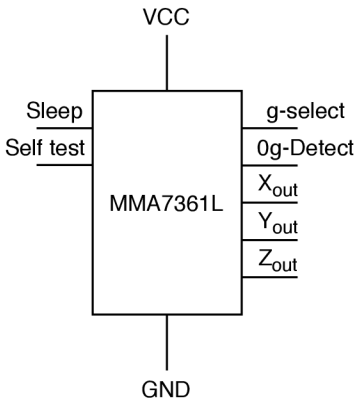


Obrázek 2.8: Struktura akcelerometru vyrobeného technologií MEMS pod mikroskopem [17].

Akcelerometry se vyrábí jako integrované obvody. Většinou měří zrychlení v jedné nebo ve třech osách, a mají různý rozsah (bývá softwarově nastavitelný). Naměřené hodnoty lze získat většinou jako napěťové úrovně, které je následně nutno pomocí A/D převodníku převést na číselnou hodnotu, nebo může mikrokontroler s akcelerometrem komunikovat například po sériové lince (nejčastěji I2C nebo SPI). Rozhraní akcelerometru MMA7361L znázorněno na obrázku 2.9.

Hodnoty získané z akcelerometru mají spoustu využití. V chytrých zařízeních jsou využívány pro zjištění polohy zařízení, v kamerách pro stabilizaci obrazu, v pevných discích pro přepnutí disku do bezpečného režimu při pádu nebo vibracích, a nebo pro měření síly působící na autíčko, které jede po autodráze.

Zpracování dat z akcelerometru bude v případě této práce probíhat následujícím způsobem. Mikrokontrolér bude s akcelerometrem pomocí I2C sběrnice komunikovat a získané data vyfiltruje a uloží do paměti pro další použití.

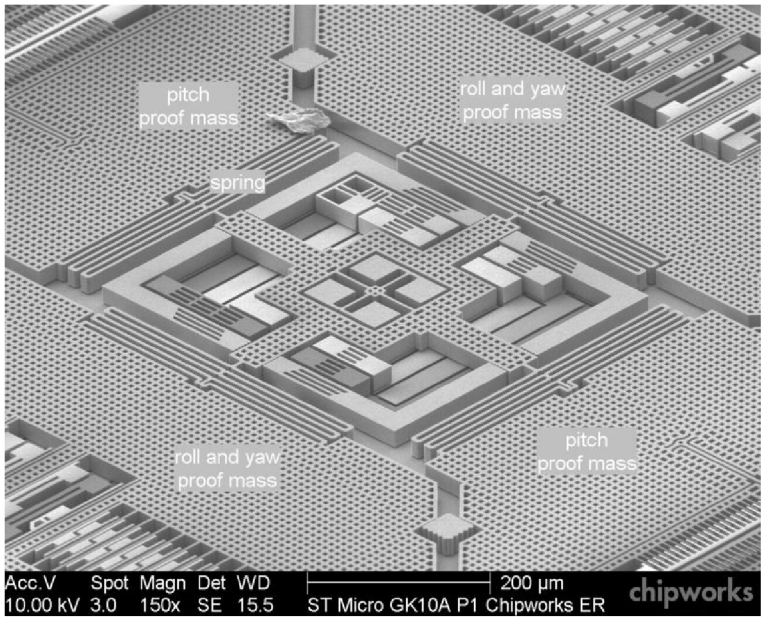


Obrázek 2.9: Schématické zobrazení rozhraní akcelerometru MMA7361L [14].

Při zpracování dat z akcelerometru (ale i ostatních senzorů) je potřeba počítat se zpožděním, které vznikne převodem analogového signálu na digitální v A/D převodníku. Doba převodu závisí na typu převodníku a jeho nastavení. Další zpoždění signálu nastane, pokud jsou data přenášena pomocí sériové komunikace.

2.3.3 Gyroskop

Elektronický gyroskop měří otáčení v rovině. Stejně jako akcelerometry, jsou moderní gyroskopy vyrobeny pomocí technologie MEMS. Tyto gyroskopy fungují opět na princupu seismické hmoty na kterou působí síla (obrázek 2.10). V případě gyroskopu je to většinou Coriolisova síla[2]. Stejně jako akcelerometry většinou měří působení sil v jedné nebo ve třech osách, a často bývají společně s akcelerometrem v jednom pouzdře integrovaného obvodu. Takový obvod pak dokáže měřit dohromady všech šest pohybových os (anglicky Six degrees of freedom).

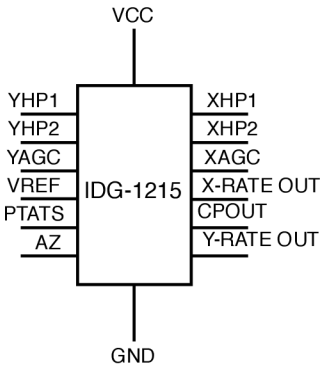


Obrázek 2.10: Struktura gyroskopu vyrobeného technologií MEMS [13].

Hodnoty získané z gyroskopů jsou využívány v chytrých zařízeních,ve vesmírných lodích a družicích, v automobilovém a leteckém průmyslu, v dronech, modelářských helikoptérách, v brýlích pro virtuální realitu a dalších aplikacích.

Komunikace mikrokontroleru a gyroskopu většinou probíhá pomocí sériového rozhraní, nebo jsou hodnoty přenášeny jako napěťové úrovně (obrázek 2.11). Zpracování dat je velmi podobné jako u akcelerometru.

[2]Více na: https://en.wikipedia.org/wiki/Coriolis_force



Obrázek 2.11: Schéma rozhraní dvouosého gyroskopu IDG-1215 [18].

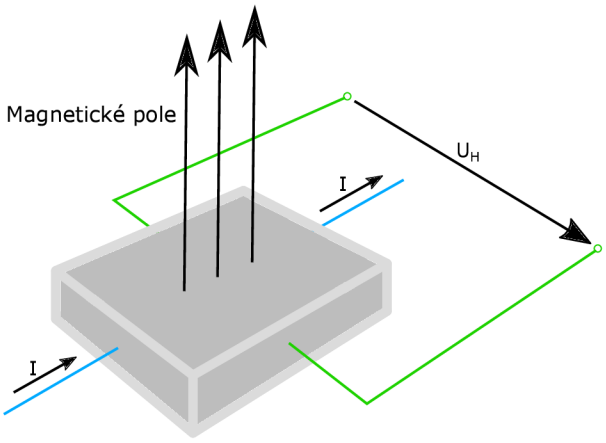
2.3.4 Hallův senzor

Hallův senzor slouží k bezkontaktnímu měření magnetických polí. Využívá tzv. Hallova jevu[3]. Senzor je tvořen polovodičovou destičkou, kterou protéká elektrický proud. Po vložení této destičky do magnetického pole se náboj hromadí na jedné straně destičky a tím vzniká napětí, kolmo na směr protékajícího proudu, které je úměrné magnetickému poli (obrázek 2.12).

Tyto senzory mají široké možnosti využití, jako například měření proudu, detekce stisku tlačítek, prvky zabezpečení budov a měření pozice různých věcí.

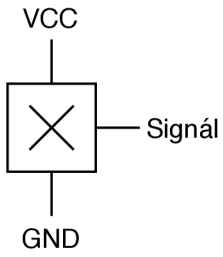
Vyrábí se v různých provedeních od speciálních tvarů pro měření proudu v kabelech až po malé pouzdra určené pro montáž na desky plošných spojů.

Výstupní analogové napětí lze AD převodníkem převést na digitální hodnotu a tu dále využít v mikrokontroleru (schématická znázornění na obrázku 2.13).



Obrázek 2.12: Princip Hallova snímače. I je protékající proud, U_H je hallovo napětí odpovídající velikosti a směru magnetického pole.

[3] Více na: https://en.wikipedia.org/wiki/Hall_effect



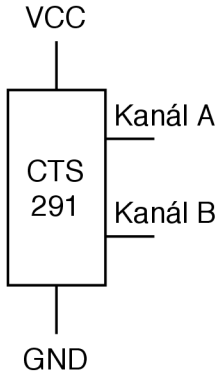
Obrázek 2.13: Schématická značka Hallova senzoru.

2.3.5 Optický rotační enkodér

Rotační enkodéry (schématické znázornění na obrázku 2.14) slouží k měření otáček nebo pozice. Na rozdíl od potenciometrů mají rotační enkodéry neomezený počet otáček a jsou tedy vhodnější pro některé použití. Tato sekce se bude zabývat pouze optickými rotačními enkodéry. Mechanické a lineární enkodéry pracují na podobných principech. Rotační enkodéry se dělí na inkrementální a absolutní.

Rotační enkodéry se skládají z disku, zdroje světla a snímačů světelného paprsku. Disk může být vyroben z neprůsvitného materiálu a obsahovat průsvitná okénka. Druhý typ disku je z průsvitného materiálu, který je potištěn neprůsvitnými tvary. Výstupní signál z enkodéru je připojen k mikrokontroleru, kde je dále zpracovávám. U signálů z inkrementálních senzorů může mikrokontroler pomocí čítače a přerušení například počítat počet náběžných hran, a od toho odvodit rychlost a velikost otáčení. U absolutních enkodérů je přímo vyhodnocena poloha.

Rotační enkodéry mají spoustu využití. Díky nim lze přesně měřit otáčky a natočení hřídele běžných elektromotorů a není potřeba používat krokové motory. Jsou využívány například v tiskárnách, počítačových myších, ovládacích prvcích různých zařízení a dalších.

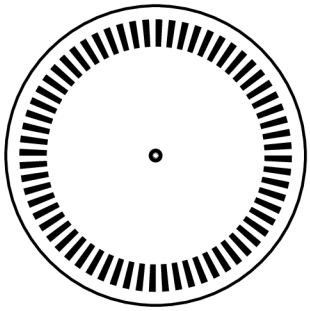


Obrázek 2.14: Schématické znázornění inkrementálního optického rotačního enkodéru CTS 291 [12].

Inkrementální optický rotační enkodér

Tento typ enkodéru slouží pro měření otáček a funguje pouze pokud se disk otáčí. Po obvodu disku jsou umístěny otvory ve stejných rozestupech, a při otáčení disku svítí skrz otvory světlo na snímač, který generuje obdélníkový signál (obrázek 2.15). Podle šířky obdélníků

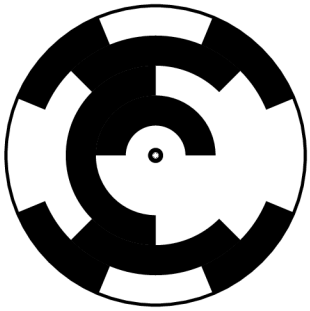
lze určit rychlost otáček. Pokud má senzor podporovat i rozlišování směru otáček, tak jsou na disku umístěny další otvory, posunuté o několik stupňů oproti předchozím. Další snímač generuje opět obdélníkový signál. Pomocí těchto dvou signálů lze určit i směr otáčení disku.



Obrázek 2.15: Disk inkrementálního optického enkodéru.

Absolutní optický rotační enkodér

Absolutní enkodér slouží pro získání úhlu otočení a funguje i pokud se disk neotáčí. Tento typ enkodéru používá několik světelných snímačů v řadě, a jejich počet udává rozlišení výstupní hodnoty. Na disku jsou natištěny obrazce v binárním nebo Grayově kódu, které udávají jednoznačnou hodnotu pro každé pootočení disku (obrázek 2.16).



Obrázek 2.16: Disk absolutního rotačního enkodéru. Využívá 4 bitový Grayův kód.

Přesnost měření rotačního enkodéru

Přesnost rotačního enkodéru závisí na jeho konstrukci a typu. U inkrementálních enkodérů je přesnost závislá na počtu otvorů nebo odrazných ploch na disku. Čím víc jich je, tím přesnější je měření.

U absolutních enkodérů záleží na počtu bitů kódu, který senzor využívá. Počet bitů určuje rozdělení jedné otáčky na díly. Větší počet bitů tedy znamená přesnější měření. Při použití binárního kódu může navíc nastat stav, kdy při přechodu mezi dvěma kombinacemi hodnot naměří senzor chybnou hodnotu.

2.3.6 Senzor rozpoznávající barvy

Senzory barev jsou obvykle složeny z fotodiod nebo jiných senzorů světla a světelných zdrojů (obrázek 2.17). Světlo ze světelného zdroje se odráží o plochu před senzorem a fotodiody měří jeho intenzitu. Každá fotodioda většinou obsahuje barevný filtr díky kterému měří jen intenzitu světla stejné barvy jako má filtr. Nejčastěji používané filtry jsou červené, zelené a

modré. Z informací o intenzitě světla těchto tří barev lze poté zjistit jakou barvu má plocha která se nachází před senzorem.

Senzory barev jsou většinou používány kontrolu výrobků nebo třeba pro roboty sledující čáru.

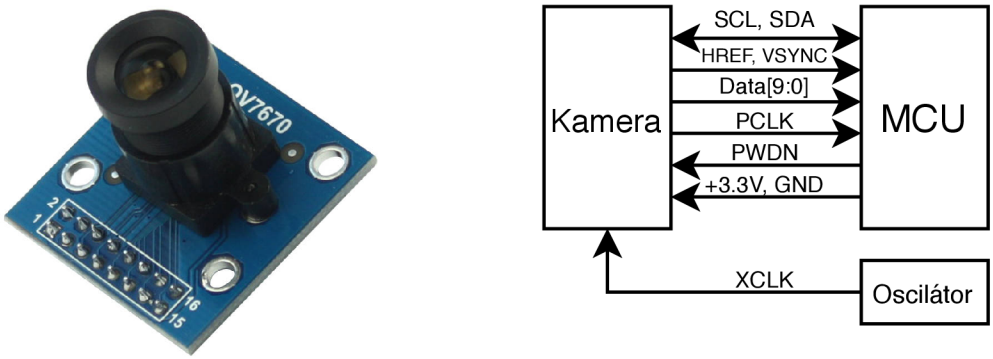
S mikrokontrolerem můžou komunikovat přes sériovou linku (SPI, I2C) nebo informaci předávají pomocí střídy či jako napěťové úrovně. Každá barevná složka je samostatná a mikrokontroler podle jejich velikostí určí výslednou barvu, kterou senzor naměřil.

Obrázek 2.17: Senzor rozpoznávající barvy [23].

2.3.7 Videokamera

Videokamery slouží ke snímání pohyblivého obrazu. Skládají se z citlivého senzoru a objektivu (obrázek 2.18). Světlo prochází objektivem a dopadá na senzor, který ho přemění na elektrický proud. Proud je následně převeden na 2D obraz. V průmyslu a robotice se využívají například pro vizuální kontrolu výrobků, nebo navigaci robotů.

Existují moduly s kamerou určené i pro připojení k mikrokontrolerům včetně Arduina. Tyto kamery mají ovšem na dnešní poměry hodně malé rozlišení (např. 640x480 pixelů) a většinou i malou snímkovou frekvenci. K mikrokontrolerům se dají kamery připojit pomocí sériového rozhraní. Zpracování obrazu je náročné na výkon procesoru, protože obraz z kamery obsahuje velké množství dat.



Obrázek 2.18: Kamera Arducam OV7670 určena k připojení k Arduinu a rozhraní kamery pro komunikaci s mikrokontrolerem [1].

Kapitola 3

Rozbor, návrh a řešení problému

V této kapitole jsou popsány některé problémy které je nutné vyřešit při návrhu algoritmů pro autonomní jízdu po autodráze a následně můj návrh řešení a jeho realizace.

3.1 Rozbor problému

Rozpoznání ujetého kola

Jeden z hlavních problémů je rozpoznání ujetí jednoho celého kola, případně přejetí cílové čáry. Tato informace je důležitá pro to, aby autíčko ve správný okamžik změnilo režim z mapování dráhy na rychlé průjezdy nebo například pro počítání ujetých kol.

Tento problém se dá řešit několika způsoby. Já jsem ho vyřešil následujícím způsobem. Algoritmus zná délku trati a autíčko díky senzoru ujeté vzdálenosti pozná, že už objelo celé kolo. Moje řešení měření ujeté vzdáleností v podobě Hallova senzoru a magnetu je ovšem relativně nepřesné. Senzor měří otáčky předních kol, protože jsem předpokládal, že z nich budou přesnější informace. Zadní kola můžou prokluzovat, ale přední se točí jenom takovou rychlostí jakou jede autíčko. Při testování se tento předpoklad potvrdil, ale i přední kola se někdy nedotýkají dráhy a netočí se pořád. Kvůli tomu je informace o ujeté vzdálenosti nepřesná a chyba narůstá s ujetou vzdáleností.

Dalším možným způsobem měření ujeté vzdálenosti by bylo použití rotačního enkodéru. Takové řešení je ale složitější a náročnější na prostor, kterého v autíčku moc není.

Problém s nepřesným měřením jsem vyřešil tak, že algoritmus při každém projetí celého kola odečte od ujeté vzdálenosti hodnotu, která je podobně velká jako chyba měření za jedno kolo.

Pravděpodobně nejlepší způsob detekce ujetého kola by mohl být senzor rozpoznávající barvy pod autíčkem, který by detekoval bílou barvu startovní čáry. Tento senzor by mohl pracovat samostatně, nebo společně se senzorem ujeté vzdálenosti. Při každém projetí přes start by se synchronizovala ujetá vzdálenost a tím by byla minimalizována chyba měření.

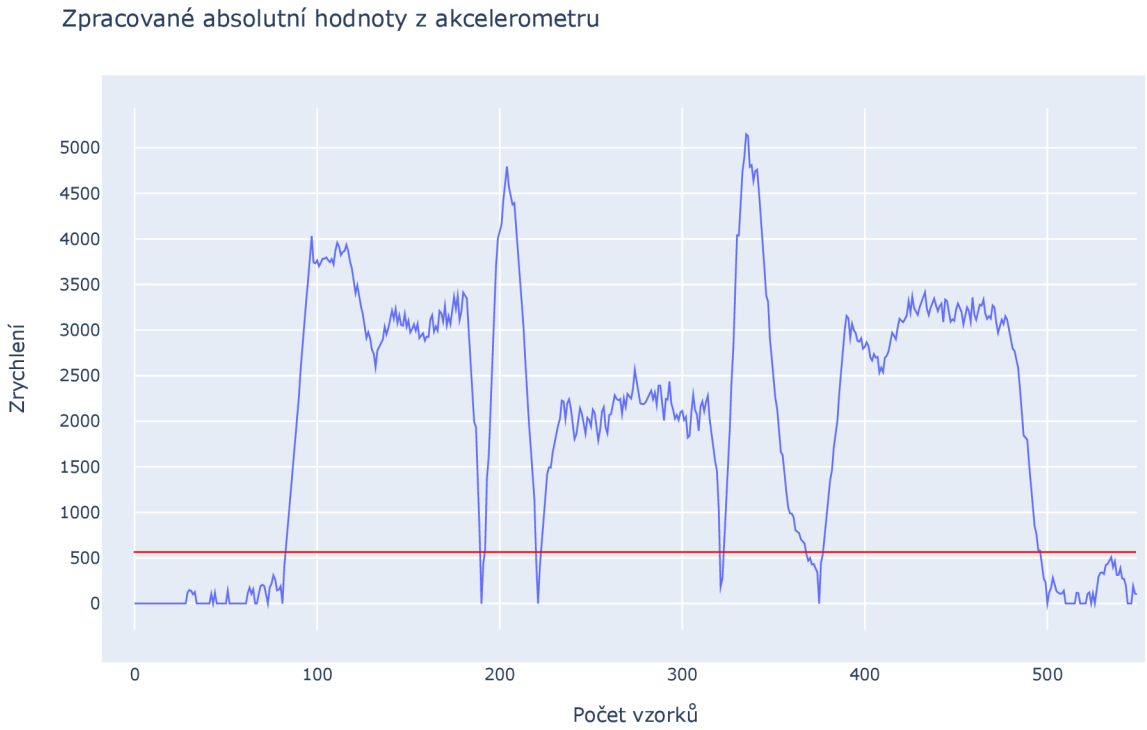
Rozpoznání zatáčky

Dalším problémem je rozpoznání zatáčky na trati. Autíčko potřebuje o zatáčce vědět nejlépe s předstihem, aby stihlo před zatáčkou zpomalit.

Rozpoznání zatáček pomocí akcelerometru funguje na principu změny zrychlení působící na akcelerometr v ose kolmé na pohyb autíčka. Relativně jednoduše se dají rozpoznávat i různě prudké zatáčky. Důležité je vhodně nastavit hodnoty, při kterých algoritmus určí

začátek a konec zatáčky, aby byly informace o zatáčkách co nejpřesnější a následná jízda co nejrychlejší (obrázek 3.1).

Pro rozpoznání zatáčky by byl vhodný i gyroskop. Princip rozpoznání zatáčky pomocí gyroskopu je podobný jako u akcelerometru. Pokud naměřená hodnota překročí určitou hranici, tak algoritmus detekuje průjezd zatáčkou.



Obrázek 3.1: Ilustrace zpracovaných hodnot z akcelerometru. Pro jednodušší rozpoznání zatáčky je použita jejich absolutní hodnota. Možná hodnota prahu je znázorněna červenou čarou.

Jiná možnost rozpoznání zatáčky je měřit úhel natočení vodící lišty autíčka a podle tohoto úhlu detekovat zatáčku. Na toto řešení ale není v autíčku moc místa.

Kontrola trakce

Posledním zde popsaným problémem je prokluzování zadních kol. To nastává pokud se rychlost otáček zadních kol prudce změní a pneumatiky nemají dostatečnou přilnavost k povrchu autodráhy. Dalším faktorem ovlivňujícím přilnavost kol k autodráze je jejich povrch a čistota jak autodráhy tak pneumatik.

Pokud začnou zadní kola prokluzovat na rovince tak to není velký problém, ale pokud se to stane v zatáčce, tak se kvůli tomu autíčko může dostat do smyku a zpomalit nebo i vypadnout z dráhy. Tento problém se dá zmírnit například pomalým zrychlováním při výjezdu ze zatáček.

Dalším možným řešením by bylo umístit do zadní části autíčka senzor barev, který by rozpoznával, jestli je autíčko ve smyku. Senzor by detekoval změnu barvy při přejetí přes

kovové napájecí kontakty v dráze. Pokud by byl senzor mezi kontakty, tak by autíčko jelo správně a pokud by jeden z kontaktů detekoval, tak by autíčko jelo ve smyku. Potom by algoritmus mohl přizpůsobit rychlost tak, aby autíčko opět získalo trakci.

Jiným řešení tohoto problému by bylo měření rychlost otáčení předních a zadních kol. Velký rozdíl těchto rychlostí by opět značil, že autíčko je ve smyku. Na podobném principu by mohlo fungovat i měření akcelerace v přední a zadní části autíčka a opět porovnávání naměřených hodnot.

Dynamika chování autíčka

V této části jsou shrnuty základní fyzikální vztahy pro výpočet rychlosti, zrychlení, hybnosti a sil, které se týkají jízdy autíčka po autodráze. Informace čerpány z [33], [34], [30], [31], [32].

Rychlost je vektorová veličina, která udává velikost a směr změny polohy tělesa v čase (vztah 3.1, kde v je rychlost, s je dráha a t je čas). Změna rychlosti se nazývá zrychlení (akcelerace). Zrychlení je derivace rychlosti podle času (vztah 3.2, kde a je zrychlení).

$$v = \frac{s}{t}[m \cdot s^{-1}] \tag{3.1}$$

$$a = \frac{\mathrm{d}v}{\mathrm{d}t}[m \cdot s^{-2}] \tag{3.2}$$

Součin hmotnosti a rychlosti autíčka se nazývá hybnost. Hybnost je veličina, která udává míru posuvného pohybu (vztah 3.3, kde p je hybnost).

$$p = mv[kg \cdot m \cdot s^{-1}] \tag{3.3}$$

Při pohybu po křivce působí na autíčko kromě dopředné síly i dostředivá a odstředivá síla. Dostředivá síla je kolmá ke trajektorii autíčka. Směr této síly je do středu křivosti trajektorie. Odstředivá síla působí opačným směrem, tedy od středu křivosti trajektorie (vztah 3.4, kde F_d je dostředivá síla, F_o je odstředivá síla a r je poloměr zatáčky).

$$F_d = F_o = \frac{mv^2}{r}[N] \tag{3.4}$$

Při jízdě autíčka rovným úsekem působí na autíčko dopředná síla. Tuto sílu způsobují pneumatiky, kterými otáčí elektromotor v autíčku. Opačným směrem na autíčko působí odpor vzduchu a tření autíčka o dráhu.

Při jízdě autíčka zatáčkou působí na autíčko dostředivá síla, kterou zajišťuje smykové tření mezi pneumatikami a povrchem autodráhy. Autíčko působí odstředivou silou na pneumatiky opačným směrem. Pokud je velikost odstředivé síly větší než velikost dostředivé síly, tak autíčko dostane smyk, což může vést až k vypadnutí z dráhy.

Z uvedených vztahů vyplývá, že důležitá je hmotnost autíčka. Čím má autíčko větší hmotnost, tím větší síla bude potřeba ke změně rychlosti a směru autíčka. Z toho důvodu budou při průjezdu zatáčkou působit na autíčko větší síly. Proto je vhodné, aby byla hmotnost autíčka co nejmenší. Pro výpočet maximální rychlosti v zatáčce je potřeba znát její poloměr, hmotnost autíčka a hlavně činitel smykového tření mezi pneumatikami a povrchem autodráhy.

3.2 Návrh řešení

V této sekci jsou popsány zvolené komponenty a návrh algoritmů pro autonomní mapování předem neznámého tvaru dráhy a algoritmu schopného využít tyto informace k projetí dráhy v co nejkratším čase.

3.2.1 Zvolené komponenty

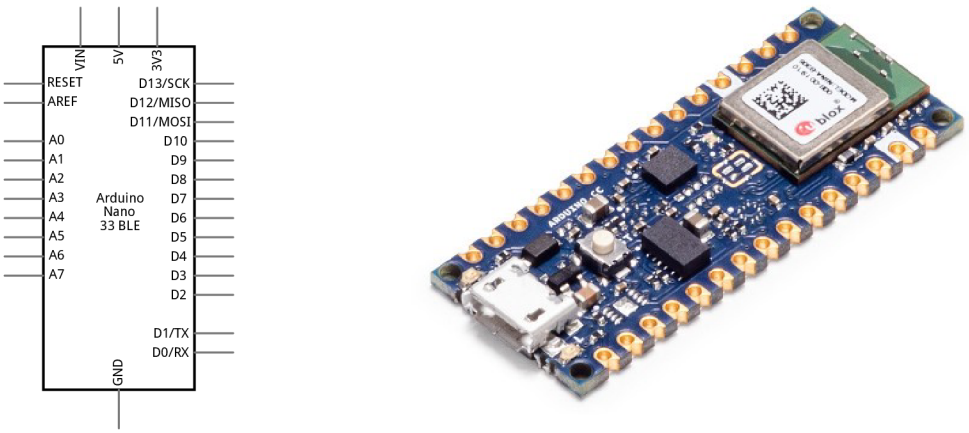
Mikrokontroler

Pro řízení autíčka jsem zvolil vývojovou desku Arduino Nano 33 BLE (obrázek 3.2) [3] [2]. Arduino jsem zvolil, protože s deskami této společnosti mám oproti konkurenčním deskám největší zkušenosti. Na této desce se nachází mikroprocesor nRF52840 s jádrem Cortex^{TM}-M4 taktovaným frekvencí 64MHz. Mikroprocesor obsahuje 1MB Flash paměti a 256kB RAM paměti. Arduino pracuje s logickými úrovněmi napětí 3.3V.

Přímo na Arduinu se nachází i integrovaný obvod LSM9DS1[25], který obsahuje tříosý akcelerometr, magnetometr a gyroskop. Dále se na Arduinu nachází 14 digitálních (6 s možností PWM) a 8 analogových pinů pro připojení senzorů a dalších komponent. Analogové hodnoty převádí na digitální AD převodník s rozlišením 10 až 12 bitů. PWM má rozlišení 8 bitů.

Programování Arduina probíhá přes mikro USB port umístěný na desce. Přes tento port je Arduino i napájeno. Další možností napájení je připojení napětí k pinům *VIN* nebo *5V*.

Arduino lze použít jako USB periferii nebo USB hostující zařízení. Lze využít i vestavěný Bluetooth čip, díky kterému se Arduino může chovat jako Bluetooth host nebo klient.

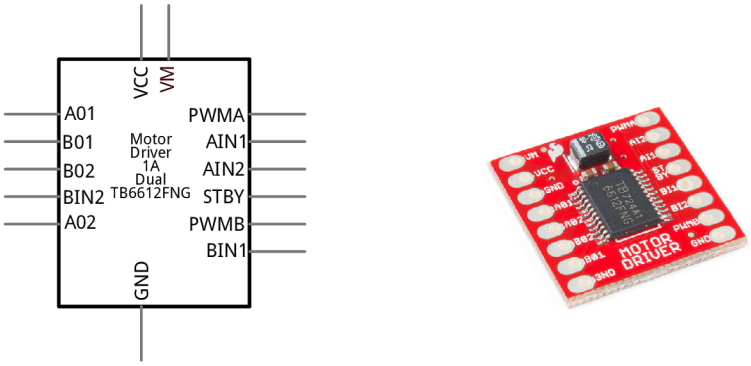


Obrázek 3.2: Schématické zobrazení rozhraní a fotka Arduina 33 BLE [3].

Obvod pro řízení motoru

Pro řízení motoru jsem zvolil desku ROB-14451 (obrázek 3.3), na které se nachází integrovaný obvod pro řízení dvou elektromotorů na bázi H-můstku TB6612FNG [27]. Tento integrovaný obvod obsahuje MOSFET tranzistory pro řízení velikosti otáček motorů pomocí PWM. Oproti možnostem základního H-můstku umožňuje obvod navíc výběr ze 4 módů běhu motoru, obsahuje vestavěnou pojistku proti přehřátí, detekci nízkého napětí a úsporný režim.

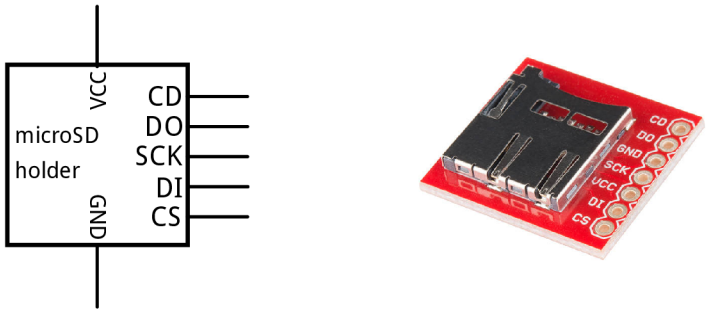
Tento obvod pracuje se stejnou velikostí logických úrovní jako Arduino. Napájení 3.3V bude připojeno přímo k Arduinu a větší napětí pro elektromotor bude získáno se sběracích kartáčků autíčka.



Obrázek 3.3: Schématické zobrazení rozhraní obvodu pro řízení motoru [24].

Ostatní komponenty

Dále jsem vybral slot pro mikro SD kartu na desce BOB-00544 (obrázek 3.4) [22], který bude sloužit pro ukládání ladících informací do souboru na SD kartu. Arduino bude s mikro SD kartou komunikovat pomocí SPI.



Obrázek 3.4: Schématické zobrazení rozhraní slotu pro mikro SD kartu a fotka desky použité v této práci [22].

Stejně jako u obvodu pro řízení motoru využívá deska se slotem pro mikro SD kartu stejné logické úrovně napětí jako Arduino.

Pro měření ujeté vzdálenosti jsem zvolil Hallův senzor DRV5056A3ELPGMQ1 [26], který je kompatibilní napájecím napětím 3.3V s kterým pracuje Arduino a je v malém pouzdru, takže bude snadné ho vestavět do autíčka.

Malé led diody o průměru 2,3mm budou sloužit pro signalizaci brzdění a jiných stavů algoritmu.

Autodráha, kterou budu používat je Carrera Evolution, stejná jako byla použita v soutěži Freescale Race Challenge. Testovací autíčko se také dříve účastnilo soutěže. LED diody ve světlech slouží k signalizaci stavu řídícího algoritmu (detekována zatáčka, brzdění).

3.2.2 Datová reprezentace tvaru dráhy

Předzpracované hodnoty zrychlení v ose kolmé na směr jízdy a počet otáček předních kol budou použity pro vytvoření datové reprezentace dráhy. Ta bude následně využita algoritmem pro projetí dráhy v co nejkratším čase. Data budou uložena v paměti mikroprocesoru jako pole datových struktur. Každá struktura bude představovat jednu zatáčku a hodnoty uložené ve struktuře budou informace o délce a prudkosti zatáčky jak je znázorněno na obrázku 3.5.

Obrázek 3.5: Ilustrace datové reprezentace tvaru dráhy. V jednotlivých buňkách pole se nachází informace o zatáčce.

Následující algoritmy pro svou činnost potřebují znát délku dráhy. Ta lze získat například z programu Carrera track planner[1]. Carrera track planner je program, který umožňuje virtuálně skládat tratě z dostupných dílů a naplánovat si tak tvar autodráhy ještě před její samotnou stavbou.

3.2.3 Algoritmus autonomního mapování tvaru předem neznámé dráhy

Algoritmus pro autonomní mapování neznámého tvaru dráhy bude rozpoznávat a ukládat zatáčky podle hodnot získaných z akcelerometru. Tyto hodnoty budou měřeny během prvního průjezdu dráhou konstantní rychlostí. U každé zatáčky uloží její začátek a konec v otáčkách ujetých od startu získaných z Hallova senzoru, a průměrnou hodnotu akcelerace v zatáčce. Následuje zjednodušený popis algoritmu v pseudokódu. Hodnoty X a Y jsou konstanty, které lze získat experimentování.

```
Nastav aktuální pozici v poli na 0.
Nastav počítadlo průchodů na 0.
Nastav průměrné zrychlení v zatáčce na 0.
Nastav rychlost motoru na konstantní rychlost.

Prováděj dokud je aktuální ujetá vzdálenost menší než délka jednoho kola:
  Pokud je aktuální hodnota zrychlení větší než X:
    Pokud je aktuální pozice v poli prázdná:
      Přidej do pole úseků na aktuální pozici informaci o zatáčce.
      Nastav začátek zatáčky na aktuální ujetou vzdálenost.

    přičti k průměrnému zrychlení v zatáčce aktuální zrychlení.
    přičti k počítadlu průchodů jedničku.

  Pokud je aktuální hodnota zrychlení menší než Y a zároveň je na
```

[1]Ke stažení například zde: https://www.autodraha.cz/track-plannery

```
  aktuální pozici v poli informace o zatáčce bez koncové vzdálenosti:
    Nastav konec zatáčky na aktuální ujetou vzdálenost.
    Vyděl průměrné zrychlení počtem průchodů a tuto hodnotu nastav jako
    průměrné zrychlení v zatáčce.
    Přičti k aktuální pozici v poli jedničku.
    Vynuluj počítadlo průchodů a průměrné zrychlení v zatáčce.
Jinak:
  Trať je zmapována.
```

3.2.4 Algoritmus schopný využít informaci o tvaru dráhy k projetí dráhy v co nejkratším čase

Algoritmus pro projetí dráhy bude měnit rychlost jízdy autíčka podle toho, jestli se autíčko právě nachází v zatáčce nebo ne. Ujetou vzdálenost získá z Hallova senzoru, a podle ní z pole načte zatáčku, ve které se právě nachází. Podle průměrné hodnoty akcelerace zjistí algoritmus jak je zatáčka prudká a podle toho upraví rychlost. Následuje zjednodušený popis algoritmu v pseudokódu. Hodnota A je rychlost na rovince a B je rychlost v zatáčce.

```
Nastav aktuální pozici v poli na 0.

Prováděj pořád dokola:
  Načti informace o zatáčce z aktuální pozice v poli.

  Pokud je ujetá vzdálenost stejná nebo větší než začátek zatáčky a
  zároveň stejná nebo menší než konec zatáčky:
    Nastav rychlost motoru na B.
  Jinak:
    Nastav rychlost motoru na A.

  Pokud je ujetá vzdálenost větší než konec zatáčky:
    Přičti k aktuální pozici v poli jedničku.

  Pokud je ujetá vzdálenost stejná nebo větší než délka tratě:
    Nastav aktuální pozici v poli na 0.
```

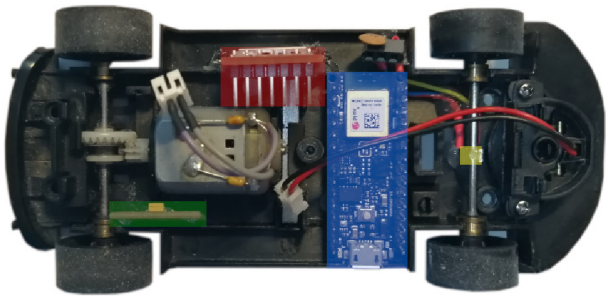
3.3 Popis realizace

Po zvolení komponent a návrhu algoritmů jsem se dostal k praktické části této práce. Začal jsem zapojením komponent na nepájivém poli pro ověření funkčnosti jednotlivých komponent i jako celku. Po úspěšném ověření jsem komponenty rozmístil v autíčku a zapojil je tak aby, se daly v případě problémů rozpojit. Dále jsem implementoval algoritmy pro mapování tvaru dráhy, projetí dráhy a testoval jsem je na různých tvarech trati.

3.3.1 Zapojení komponent

Nejdůležitější komponentou je samotné Arduino s vestavěným akcelerometrem. To je v autíčku umístěno tak, aby byl snadný přístup k USB portu, který slouží pro nahrávání programu do Arduina. Další důležitou věcí ovlivňující pozici Arduina je i to, aby se akcelerometr nacházel pokud možno co nejvíc uprostřed autíčka. Pokud by se nacházel na kraji

tak by mohlo docházet ke zkreslení hodnot. K Arduinu jsou připájeny kolíkové lišty. Na protikusy kolíkových lišt jsou připájeny dráty vedoucí k ostatním komponentám.



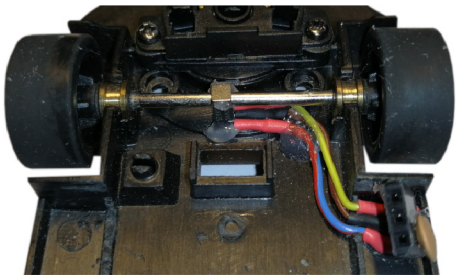
Obrázek 3.6: Umístění Arduina a dalších desek v autíčku. Modrá část je Arduino, červená deska se slotem pro mikro SD kartu, zelená deska s H-můstkem a žlutá Hallův senzor pro měření otáček přední hřídele.

Deska se slotem pro SD kartu je umístěna na kraji autíčka tak, aby ke slotu byl dobrý přístup. K desce je připájena kolíková lišta a je vodiči propojena s Arduinem. Rozmístění komponent znázorňují obrázky 3.6 a 3.10.

Na hřídelce mezi předními koly je přilepen malý magnet (obrázek 3.7). Magnet jsem vybíral z několika kusů, které jsem měl k dispozici tak, aby byl co nejmenší a zároveň jeho magnetické pole dokázal bez problémů detekovat Hallův senzor, který je umístěný pod hřídelkou. K Hallovu senzoru je připojen keramický kondenzátor s kapacitou 100 nF podle doporučení výrobce a senzor je přes konektor připojen k Arduinu.

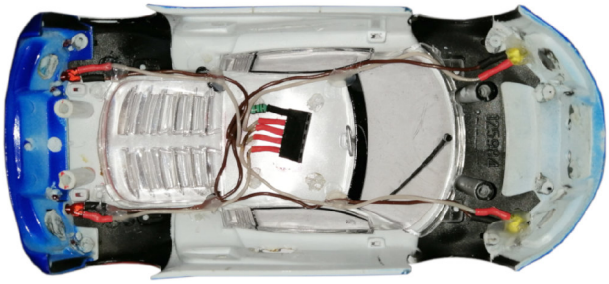
Detekce jedné otáčky magnetu znamená, že autíčko ujelo vzdálenost stejnou jako obvod předních pneumatik. Průměrná chyba ujeté vzdálenosti naměřené tímto senzorem je tedy polovina obvodu přední pneumatiky. Přední kola mohou být na začátku jízdy vždy jinak pootočená a to je jedním ze zdrojů chyby tohoto způsobu měření. Dalším zdrojem chyby je už zmiňované občasné neotáčení předních kol. Chybu způsobenou tímto problémem lze minimalizovat očištěním povrchu trati a pneumatik.

Možným vylepšením senzoru by bylo přidat na hřídelku další magnet, který by byl o 180° posunutý oproti prvnímu magnetu. Detekce magnetu by znamenala ujetí poloviny obvodu přední pneumatiky a díky tomu by měření bylo teoreticky dvakrát přesnější.



Obrázek 3.7: Pohled na řešení senzoru otáček.

Led diody jsou přilepeny ve světlech karoserie autíčka (obrázek 3.8), a jsou s Arduinem spojeny rozpojitelným konektorem na vnitřní straně střechy karoserie.

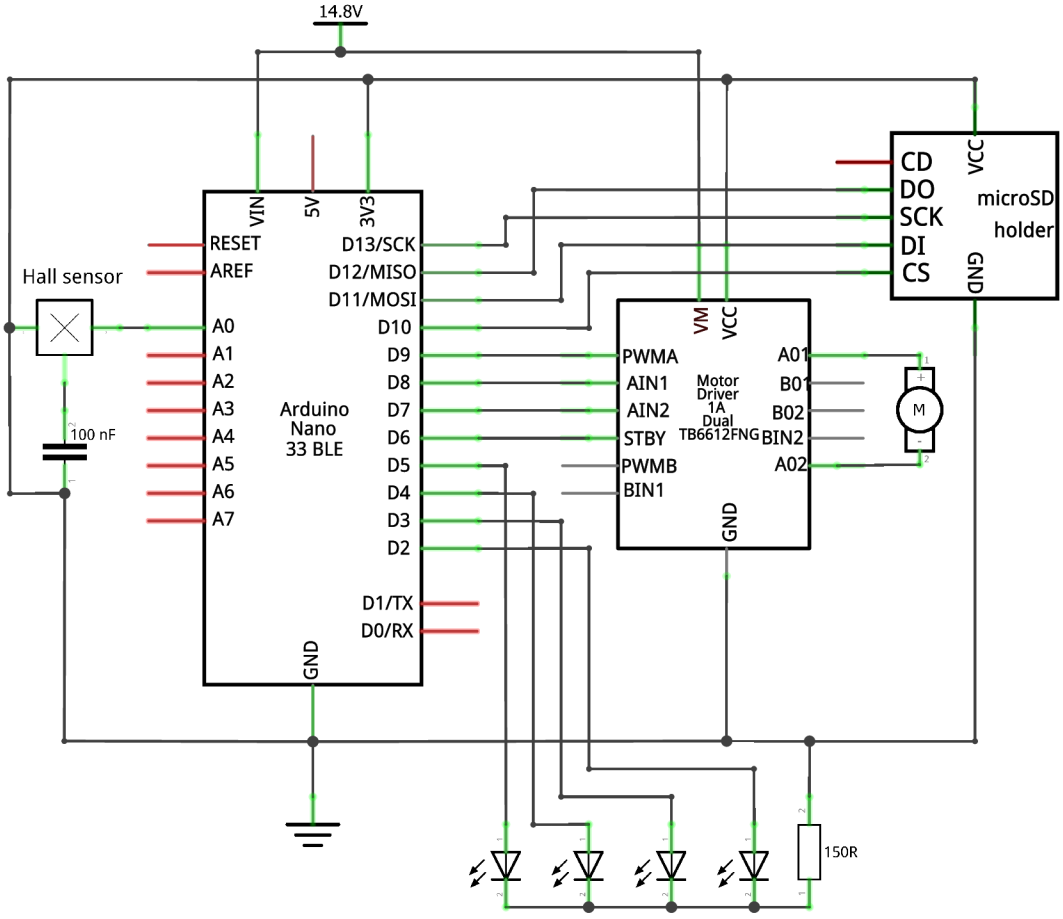


Obrázek 3.8: LED diody na vnitřní straně střechy autíčka.

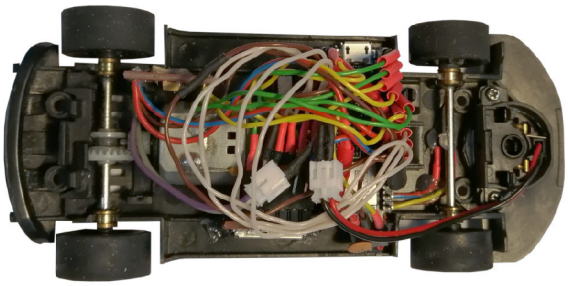
Deska s obvodem pro řízení elektromotoru je umístěna vedle elektromotoru ve volném místě autíčka. Kvůli úspoře místa jsou k ní přímo připájeny dráty pro připojení k Arduinu.

Arduino a obvod pro řízení motoru jsou konektorem spojeny se sběracími kartáčky autíčka, ze kterých jsou napájeny. Na Arduinu se nachází napěťový regulátor, díky kterému samotné Arduino a další komponenty získávají napětí 3,3 V. Schéma zapojení se nachází na obrázku 3.9.

Všechny přilepené součástky jsem lepil tavnou pistolí, tak aby se daly bez problému zase odlepit. Arduino je přišroubováno k podvozku autíčka, tak aby se samo nehýbalo ale šlo v případě potřeby z autíčka vyjmout.



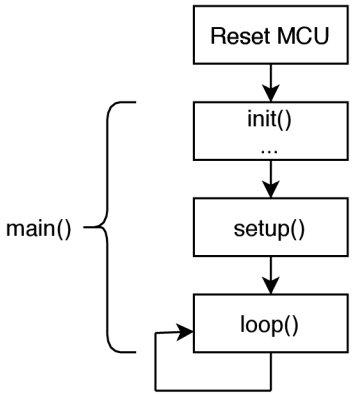
Obrázek 3.9: Schéma zapojení komponent. Vytvořeno programem Fritzing[2].



Obrázek 3.10: Celkový pohled na komponenty uvnitř podvozku autíčka i s propojovacími vodiči.

3.3.2 Implementace

Programy pro Arduino jsou v jazyce C++. Autoři Arduina ale vytvořili několik vlastních knihoven a rozšíření, která přidávají především vyšší míru abstrakce pro jednodušší práci s hardwarem. Zdrojový kód je zpracován preprocesorem a poté přeložen klasickým překladačem pro jazyk C++. Klasická funkce *main()* je v kódu pro Arduino skryta. Je v ní provedena inicializace hardwaru (funkce *init()*) a následně volání funkcí *setup()* a *loop()*, do kterých programátor doplní kód, který má arduino vykonávat. Funkce *setup()* je provedena jednou na začátku běhu programu. Funkce *loop()* je následně prováděna v nekonečné smyčce po celou dobu běhu programu (obrázek 3.11).



Obrázek 3.11: Ilustrace posloupnosti provádění kódu po zapnutí Arduina. Skrytá funkce *main()* obsahuje volání funkcí *init(), setup() a loop()*.

Arduino má svoje vlastní vývojové prostředí Arduino IDE[3], které je ale velmi jednoduché a neposkytuje funkce jako našeptávání názvů a podobně. Proto používám Visual Studio Code[4], které má všechny potřebné a užitečné funkce, a navíc díky pluginu zvládá i všechno co Arduino IDE.

Při implementaci programu jsem zvolil objektově orientovaný přístup, díky kterému je kód přehlednější. Program využívá knihovny SPI[5], SD[6] pro komunikaci s SD kartou a

[2]Informace a odkaz ke stažení: https://fritzing.org
[3]Informace a odkaz ke stažení: https://www.arduino.cc/en/main/software
[4]Ke stažení zde: https://code.visualstudio.com/
[5]Informace na: https://www.arduino.cc/en/reference/SPI
[6]Informace na: https://www.arduino.cc/en/reference/SD

knihovnu LSM9DS1[7] pro komunikaci s Akcelerometrem. Knihovnu LSM9DS1 jsem upravil pro použití v této práci a úpravy jsem popsal v sekci 3.3.2. Upravená verze knihovny LSM9DS1 je k dispozici na přiloženém médiu. Knihovnu určenou pro ovládání motoru jsem nepoužil, protože obsahovala nepotřebnou funkcionalitu. Místo ní motor ovládám pomocí svých jednoduchých funkcí. Tyto knihovny jsem zvolil, protože jsou to oficiální knihovny od autorů Arduina a jsou nimi doporučeny.

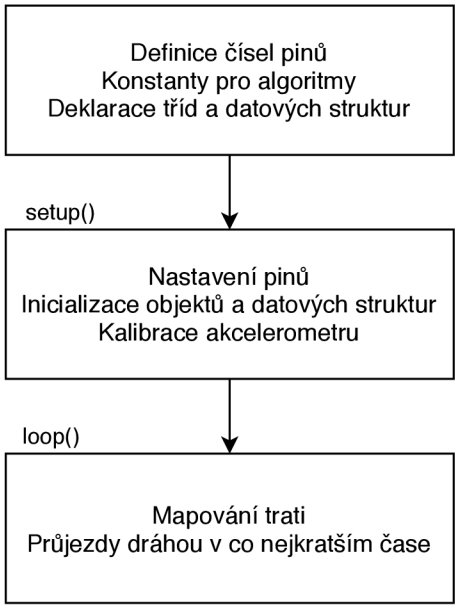
Program jsem implementoval postupně a každou část jsem testoval a upravoval tak, aby algoritmus dosahoval co nejlepších výsledků.

Původní návrh algoritmů popsaný v sekcích 3.2.3 a 3.2.4 se mi při testování zdál nedostatečný, a proto jsem se ho snažil upravit a vylepšit. Oproti návrhu algoritmus pro mapování trati nerozpoznává jenom zatáčky, ale i rovinky, brzdné zóny a výjezdy ze zatáček. Ve výjezdu ze zatáčky autíčko zrychluje pomaleji, než kdyby po zatáčce následovala zrovna rovinka a díky tomu je menší šance, že se dostane do smyku.

Do programu jsem přidal i několik částí, které se netýkají přímo jízdy, ale ulehčují práci a programování. Jedna z těchto částí, je pojistka, která zabrání točení zadních kol, pokud při spuštění Arduina Hallův senzor detekuje magnet. Tato pojistka je velmi užitečná při nahrávání programu do Arduina, protože bez ní by se celou dobu autíčko snažilo jet dopředu. Další užitečnou věcí je stav algoritmu, ve kterém autíčko po projetí prvního kola zastaví a vypíše ladící informace do souboru na SD kartě. Zjednodušený vývojový digram programu se nachází na obrázku 3.13.

Struktura programu

Program lze rozdělit na 3 hlavní části (obrázek 3.12).



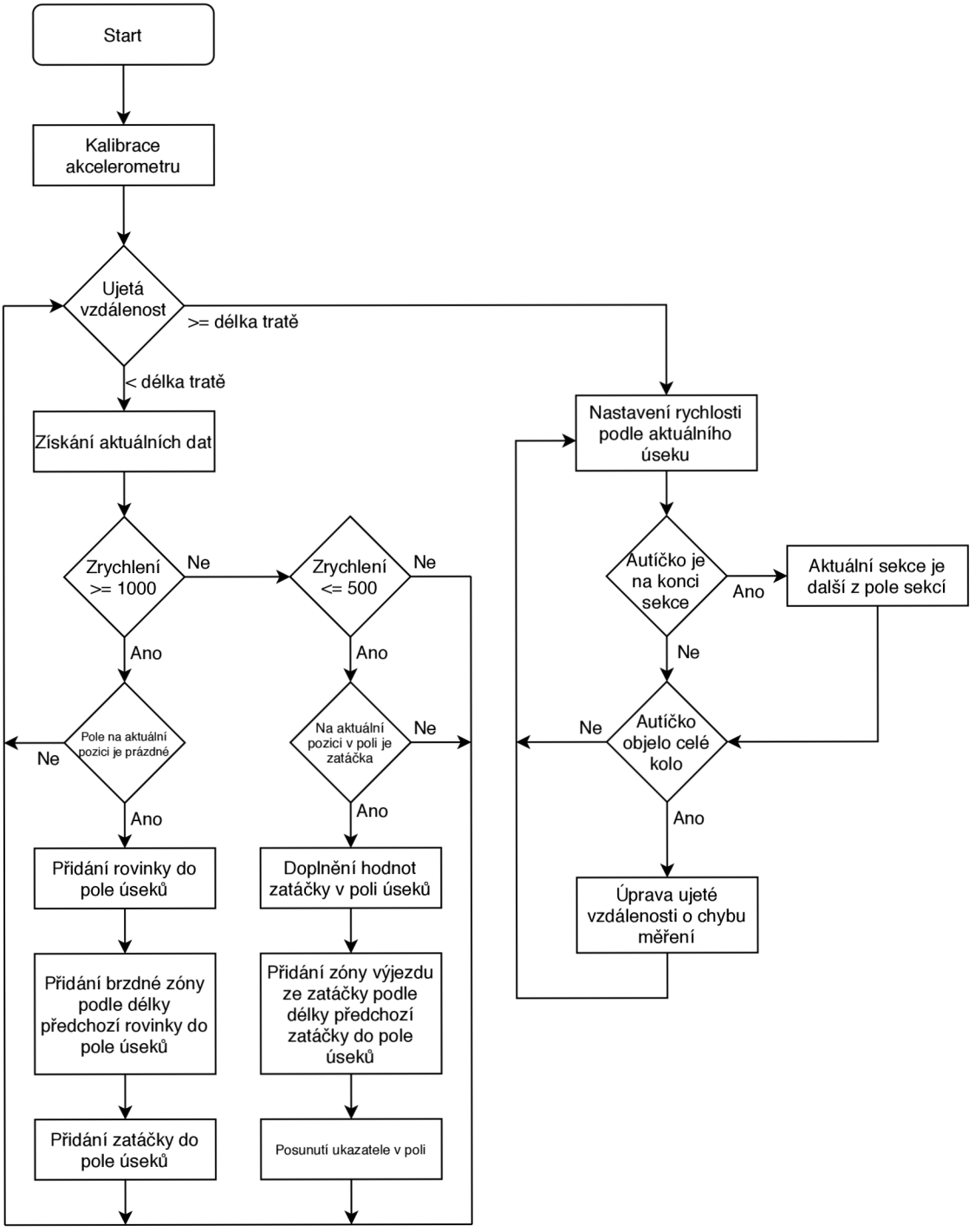
Obrázek 3.12: Ilustrace rozdělení kódu na 3 hlavní části.

V první části se nachází makra pro definici čísel pinů na Arduinu, která odpovídají propojením s dalšími součástkami. Následují konstanty používané algoritmy, deklarace datových struktur a tříd používaných dále v programu.

[7]Informace na: https://www.arduino.cc/en/Reference/ArduinoLSM9DS1

Druhá část je funkce *setup()*. V této funkci je všem používaným pinům nastaveno jestli mají být vstupní nebo výstupní, jsou inicializovány potřebné objekty a datové struktury a je provedena kalibrace akcelerometru.

Poslední částí je funkce *loop()*. Této části se budu podrobněji věnovat v následujících sekcích této práce.



Obrázek 3.13: Zjednodušený vývojový diagram algoritmů pro mapování a průjezd po dráze.

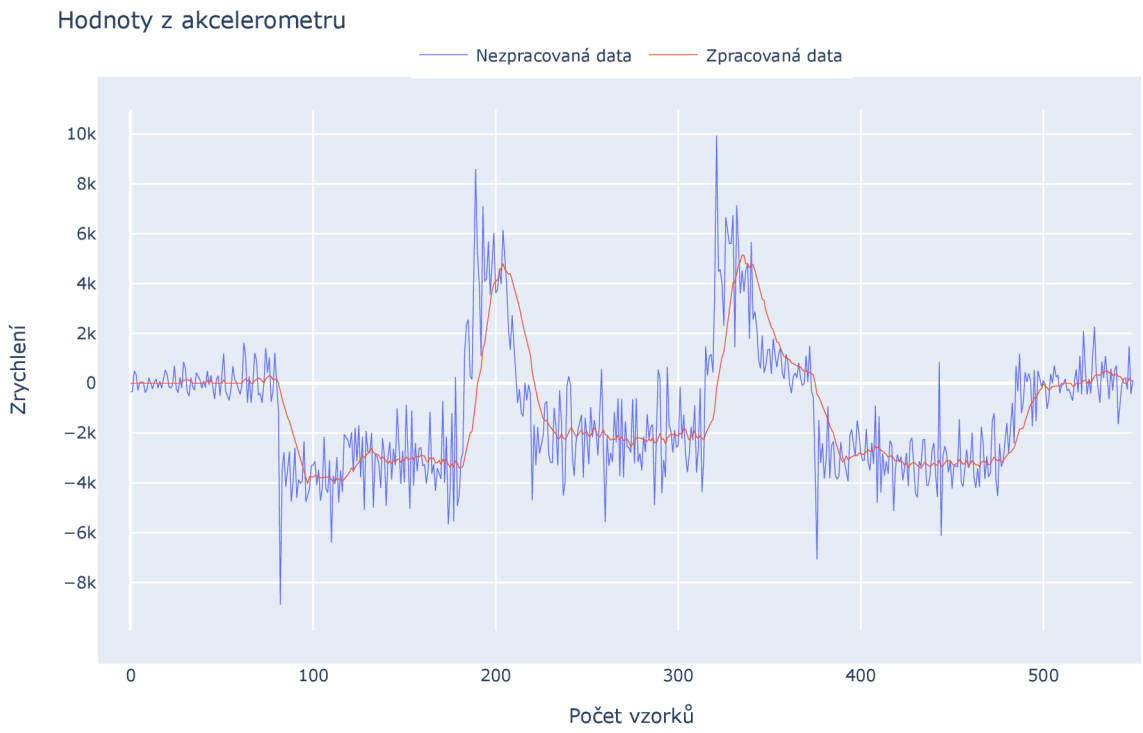
Získávání a zpracování dat

Pro čtení dat z akcelerometru používám vlastní funkci *readAcceleration(x)* z upravené knihovy LSM9DS1, která vrací celočíselnou hodnotu zrychlení v ose kolmé na směr jízdy autíčka. Funkce z původní knihovny vrací hodnoty zrychlení ve všech osách jako desetinná čísla, které jsou násobky tíhového zrychlení (g) v metrech za sekundu. Takové hodnoty nejsou pro použití v tomto případě vhodné, protože arduino rychleji a jednodušeji pracuje s celočíselnými hodnotami a důležitá je hlavně odchylka akcelerace od nuly, v jakých je jednotkách není podstatné.

Získávání dat probíhá pomocí funkce *readRegisters()* z knihovny LSM9DS1, která zahájí sériovou komunikaci s akcelerometrem, a z jeho registrů obsahujících naměřenou hodnotu akcelerace ji načte a uloží do proměnné *data*. Hodnota z proměnné *data* je následně pro každou osu uložena do odpovídající proměnné (například x) k dalšímu použití.

Vzorkovací frekvence akcelerometru je implicitně nastavena na 119 Hz. Tuto hodnotu považuji za dostatečnou pro získávání potřebných informací.

Po spuštění programu je provedena kalibrace akcelerometru (funkce *calibrate()*). Výstup kalibrace je aritmetický průměr N vzorků, který je potom odečítán od každé hodnoty získané z akcelerometru. Kalibrace zajišťuje, že hodnoty v klidovém stavu jsou blízké nule.



Obrázek 3.14: Rozdíl mezi daty přímo z akcelerometru, a předzpracovanými daty, které používá algoritmus. Data jsou z tvaru dráhy 1 (obr. 4.2a).

Hodnota zrychlení je přečtena (pomocí funkce *readAcceleration(x)*) z akcelerometru a je od ní odečtena hodnota získaná kalibrací. Na klouzavý průměr M takových hodnot je poté aplikováno prahování. Výsledná hodnota je pak dále používána algoritmy.

Průměrování hodnot zajišťuje, že data jsou vyhlazená a dá se s nimi lépe pracovat. Prahování slouží ke snížení chyby měření, když na autíčko nepůsobí akcelerace. Prahování je realizováno tak, že pokud je hodnota akcelerace menší než nějaký práh P, tak je změněna na 0, jinak je hodnota ponechána.

Výstup z Hallova senzoru je připojen k analogovému pinu Arduina a napětí je převedeno na digitální hodnotu pomocí 10 bitového AD převodníku (funkce *analogRead()*). Pokud je hodnota větší než daný práh R, je to vyhodnoceno jako jedna otáčka předních kol. Ujetá vzdálenost je pak získána vynásobením otáček a obvodu předních kol.

Hodnoty N, M, P a R jsou konstanty, při kterých program během testování vykazoval nejlepší výsledky. Na obrázku 3.14 je vidět průběh hodnot z akcelerometru před a po zpracování.

Doba od naměření dat do jejich použití v programu se skládá z několika částí. Nová data z akcelerometru jsou dostupná přibližně každých 9,61 ms. Program nová data přečte a zpracuje výše popsaným způsobem. Přenos pomocí I2C sběrnice zabere nějaký čas. Pravděpodobně největší vliv na zpoždění dat při zpracování má průměrování. Vliv průměrování na zpoždění signálu lze vidět na obrázku 3.14. Program tedy pracuje s daty, které mají oproti reálným hodnotám určité zpoždění. Z hodnot naměřených v experimentu 4.1.5 vím, že autíčko mělo na dráze o délce 530cm (vnitřní dráha tvaru 2) průměrný čas 2,66 s. Z těchto hodnot lze vypočítat, že průměrná rychlost je 199,248 cm/s. Za 9,61 ms tedy ujede 1,914 cm.

Průjezd prvním kolem a mapování dráhy

Po každém zavolání funkce *loop()* jsou získány aktuální hodnoty akcelerace a otáček kol autíčka. Proměnná *state* udává, v jakém stavu se právě autíčko nachází, po spuštění má hodnotu 0. Autíčko se rozjede konstantní rychlostí, která je nastavena tak, aby bez problému projelo i nejprudší zatáčku na autodráze. Pokud je detekována zatáčka (akcelerace je větší než nastavený práh) tak je do pole úseků trati přidána rovinka, která začíná na konci předchozího úseku (pokud předchozí úsek neexistuje tak na 0) a pokud je rovinka dostatečně dlouhá tak je na jejím konci vytvořená brzdná zóna (před první zatáčkou je vytvořena vždy). Délka brzdné zóny závisí na délce předešlé rovinky. Potom je do pole přidána zatáčka. Autíčko pokračuje v jízdě dokud hodnoty akcelerace neklesnou pod další hranici. Při té je zatáčce nastavena koncová délka a vypočítán průměr hodnot zrychlení v zatáčce, který udává jak je zatáčka prudká. Pokud byla zatáčka dostatečně dlouhá, tak za ní v poli úseků vytvořen výjezd ze zatáčky. Tento úsek, ve kterém autíčko zrychluje pomaleji než na rovince zamezuje vypadávání autíčka z dráhy. Následuje vytvoření další rovinky a cyklus pokračuje, dokud autíčko neujede vzdálenost stejnou jako je délka trati, která je nastavena v programu. Po projetí celé dráhy je nastaven začátek první rovinky na konec poslední sekce v poli, tak aby jednotlivé sekce dráhy v poli popisovaly celou délku autodráhy.

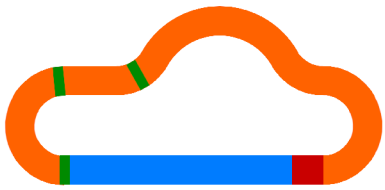
Následuje vynulování ujeté vzdálenosti a dalších hodnot a přepnutí stavu na další průjezdy dráhou, nebo na stav kdy autíčko zastaví a vypíše ladící informace do souboru na SD kartu.

Algoritmus je naprogramován tak, aby se jednotlivé sekce v nepřekrývaly. Sekce nikdy nesmí začínat a končit ve stejné vzdálenosti, musí mít délku minimálně 1 otočku kol. Proto například krátké rovné úseky mezi zatáčkami algoritmus do paměti nezaznamená.

Výpis mapy dráhy na SD kartu a jeho vizualizace (obrázek 3.15)vypadá následovně:

```
76  7   -1  STRAIGHT
8 10 -1 BRAKING
11 45 2915 CORNER
46 47 -1 C_EXIT
48 54 3802 CORNER
55 56 -1 C_EXIT
57 73 3399 CORNER
74 75 -1 C_EXIT
```

První dva sloupce značí začátek a konec sekce v otáčkách kol autíčka. Následuje sloupec s průměrnými hodnotami zrychlení v zatáčkách a nakonec název sekce, pro jednodušší orientaci v informacích. Ukázka je z tvaru dráhy 1 (obr. 4.2a).



Obrázek 3.15: Vizualizace mapy dráhy podle výše uvedených hodnot. Modrá oblast značí rovinku, červená oblast je brzdná zóna, oranžové oblasti značí zatáčky a zelené oblasti výjezdy ze zatáček.

Další průjezdy dráhou

Po průjezdu dráhy konstantní rychlostí a přepnutí do stavu co nejrychlejší jízdy podle informací o tvaru dráhy se algoritmus chová následovně. Na začátku smyčky (funkce *loop()*) autíčko vždy načte aktuální ujetou vzdálenost a podle této vzdálenosti z pole získá úsek dráhy na kterém se právě nachází. Podle tohoto úseku upraví rychlost motoru. Jednotlivé rychlosti jsem zvolil experimentálně, tak aby autíčko jelo co nejrychleji, a nevypadlo z dráhy. Rychlost v zatáčce autíčko volí z několika hodnot podle prudkosti zatáčky.

Kvůli nepřesnostem v měření vzdálenosti musí být hodnota ujeté vzdálenosti jednou za čas upravena, tak aby byla chyba co nejmenší. Pokud chyba naroste tak se zhorší časy autíčka a hrozí i vypadnutí z dráhy. Algoritmus na začátku každého (kromě prvního) kola nastaví hodnotu ujeté vzdálenosti na malou zápornou hodnotu závislou na délce dráhy, tak aby byla minimalizována chyba. Výpočet hodnoty provádí následující kód:

```
error_coeff = -1 * ((TRACK_LENGTH / ERROR_CONSTANT) + (TRACK_LENGTH
% ERROR_CONSTANT != 0))
```

Každé třetí kolo je hodnota *error_coeff* ještě o 2 menší. Konstanta *ERROR_CONSTANT* má hodnotu 300. Tyto hodnoty byly experimentálně zjištěny testováním při implementaci. Funkčnost tohoto řešení je ukázána v experimentu 3 v sekci 4.1.3.

LED diody

LED diody ve světlech autíčka slouží k signalizaci různých stavů a informací během jízdy. Ve finální verzi algoritmu svítí přední světla pouze při průjezdu prvním kolem a jejich zhasnutí tak značí, že se algoritmus přepl do módu rychlé jízdy. Brzdové světla svítí při brzdění.

Výpis ladících informací na SD kartu

Arduino komunikuje s SD kartou přes sériové rozhraní SPI. Využívá k tomu dva datové vodiče, jeden hodinový a jeden volící vodič. Komunikace probíhá pomocí knihovny SD. Po spuštění Arduina je pomocí funkce *begin(CS)* inicializován přístup ke kartě. Parametr CS je číslo pinu, který slouží pro vybrání slave zařízení při komunikaci pomocí SPI, v tomto případě je to SD karta. Návratová hodnota funkce *begin(CS)* značí jestli byla komunikace navázána nebo ne. Navázání komunikace může selhat například když není karta vložena do slotu.

Práce se soubory je stejná jako jakákoliv jiná práce se soubory v C++. Proměnná typu *File*, odkazuje na soubor na kartě, do kterého bude program zapisovat. Pro zápis slouží funkce *writeOnce(string)*, která otevře soubor pro zápis (*file = open("log_file.txt", FILE_WRITE)*), do souboru zapíše řetězec předán v parametru (*file.print(string)*) a soubor uzavře (*file.close()*). Soubor je nutno uzavřít, aby z něho v počítači následně šlo číst. Pro zápis několika řetězců po sobě lze soubor otevřít, zapisovat do něj a následně ho zavřít i ručně za použití funkcí *openFile()*,*write()* a *closeFile()*.

SD kartu je následně možno připojit k počítači a soubor přečíst.

Softwarové ovládání elektromotoru

K ovládání elektromotoru slouží obvod s h-můstkem, jehož princip jsem již popsal v předchozích částech této práce v sekcích 2.3.1 a 3.2.1. Zde bych se chtěl zmínit o ovládání h-můstku a tím i motoru z programu.

Arduino komunikuje s obvodem, který jsem zvolil, pomocí 4 signálů (obrázek 3.3). Nejdůležitější jsou PWMA, AIN1 a AIN2 (čip, který jsem zvolil je určen pro řízení 2 motorů a proto jsou signály označeny jako A. Druhý h-můstek se signály označenými B nevyužívám). Různá kombinace hodnot signálů AIN1 a AIN2 určuje směr otáčení motoru a ovládání brzdění (tabulka 3.1). Signál PWMA slouží k ovládání rychlosti otáček v nastaveném směru. Poslední signál je STBY. Pokud je STBY nastaven na logickou 0 tak je čip v úsporném režimu.

Ovládání v programu je jednoduché. Signály AIN1 a AIN2 jsou binární hodnoty, které stačí nastavit zapsáním na digitální pin Arduina podle požadovaného směru otáčení nebo brzdění. PWMA je v programu hodnota od 0 do 255 která je následně arduinem převedena na PWM signál. K ovládání rychlosti motoru slouží funkce *drive(speed)*, která nastaví rychlost motoru na hodnotu parametru. Při nastavení rychlosti na 0 motor neklade žádný odpor a autíčko ještě nějakou dobu jede setrvačností. Funkce *brake()* přepne obvod pro řízení motoru do režimu brzdění. Tento režim propojí svorky motoru a procházející proud indukovaný otáčejícím se rotorem působí proti otáčkám rotoru a motor tímto aktivně brzdí.

Vstup				Výstup		
IN1	**IN2**	**PWM**	**STBY**	**OUT1**	**OUT2**	**Mód**
H	H	H/L	H	L	L	Brzda
L	H	H	H	L	H	Proti směru hod. ručiček
		L		L	L	Brzda
H	L	H	H	H	L	Po směru hod. ručiček
		L		L	L	Brzda
L	L	H	H	Vypnuto (stav vysoké impedance)		Zastavení
H/L	H/L	H/L	L	Vypnuto (stav vysoké impedance)		Standby režim

Tabulka 3.1: Závislost výstupu obvodu pro ovládání motoru na vstupních signálech [27].

3.4 Další možné přístupy k fungování algoritmu

V následující sekci jsou uvedeny některé další možnosti fungování algoritmů pro mapování a jízdě po neznámém tvaru dráhy.

Periodické ukládání naměřených dat

Nejjednodušším způsobem ukládání dat a tvorby mapy tratě by bylo data periodicky ukládat do paměti bez dalšího zpracování.

Algoritmus by potom při jízdě po trati četl uložené hodnoty s předstihem, tak aby měl informaci o trati nacházející se před autíčkem. Problém u tohoto způsobu je ten, že autíčko musí správně číst uložené hodnoty v závislosti na své poloze a rychlosti. Pokud by autíčko četlo hodnoty moc blízko nebo moc daleko od jeho aktuální pozice tak může ztratit hodně času, nebo i vypadnout z dráhy. Další nevýhodou je relativně velký objem ukládaných dat.

Využití neuronové sítě

Dalším možným způsobem řízení autíčka na neznámém tvaru autodráhy by mohlo být využití neuronových sítí. Neuronová síť by dostávala informace o akceleraci a případně i data z dalších senzorů. Výstupem by byla rychlost autíčka.

Data by mohla být uložena jako v předchozím případě. Natrénovaná neuronová síť by podle následujících hodnot akcelerace určila jakou rychlostí by autíčko mělo jet. Tento způsob řízení autíčka by závisel na správném natrénování neuronové sítě, a mohl by poskytovat lepší výsledky než ručně nastavené hodnoty rychlostí pro rozsahy zrychlení jako u jiných algoritmů.

Trénování by probíhalo na různých tvarech autodráhy nebo pouze z dat naměřených při jízdách. Neuronová síť by například dostala hodnoty akcelerací v různě prudkých zatáčkách a maximální rychlosti, kterými je autíčko schopno dané zatáčky projet.

Modifikace s měřením času

Algoritmus použitý v této práci by se dal modifikovat tak, aby měřil čas, za který autíčko projede jednotlivé úseky trati nebo celé kolo. Změřený čas například průjezdu zatáčkou by potom mohl porovnat s časem ze stejné zatáčky v předchozím kole, a podle toho upravit rychlost v dané zatáčce. Pokud by se čas zlepšoval tak by algoritmus postupně zrychloval,

a když by se čas zhoršil, což by znamenalo, že autíčko jede příliš rychle a dostává smyk, tak by algoritmus hodnotu rychlosti snížil na poslední hodnotu při které ke smyku nedocházelo. Díky tomu by algoritmus postupně mohl zrychlovat až na maximální možnou rychlost na dané trati při aktuálních podmínkách.

Ke správné funkci této modifikace je ale nutné, aby autíčko přesně vědělo kde se právě nachází a aby přesně detekovalo průjezd cílem.

Modifikace s aktuální rychlostí

Další možnou modifikací algoritmu by mohlo být počítání aktuální reálné rychlosti autíčka z hodnot akcelerace a otáček kol. K různě prudkým zatáčkám by byla přiřazena maximální rychlost, kterou je autíčko schopno zatáčku projet. Algoritmus by pak před zatáčkou zpomalil jenom na tuto maximální rychlost a na rovinkách by se snažil jet co nejrychleji.

Kapitola 4

Experimenty a zhodnocení řešení

V této kapitole se nacházejí experimenty znázorňující funkčnost algoritmů a autíčka na několika tvarech dráhy. Nachází se zde také krátké zhodnocení řešení.

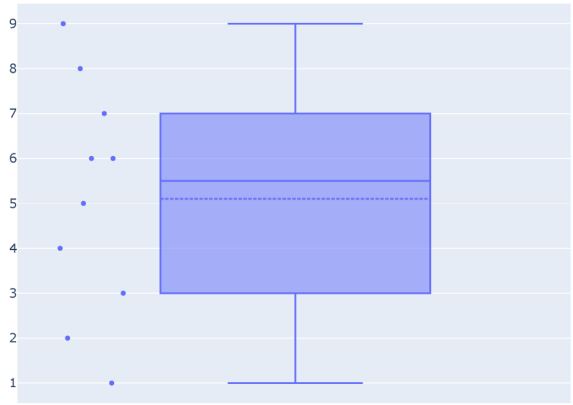
4.1 Experimenty

Doba jízdy autíčka je měřena pomocí Elektronického počítadla kol Carrera 71590. Toto počítadlo kol zobrazuje počet ujetých kol a čas, za který autíčko ujelo poslední kolo pro každou dráhu. Měří s přesností na setiny sekundy, což je pro tyto účely dostačující.

Před každým experimentem jsem otřel dráhu i pneumatiky autíčka navlhčeným kusem látky, abych se zbavil prachu a nečistot a jízdy probíhaly ve srovnatelných podmínkách. Dosažené časy jsem přepsal do souboru, z kterého byly načteny skriptem v jazyce Python, a pomocí knihovny plotly[1] z nich byly vytvořeny grafy.

Boxplot graf

Pro zobrazení naměřených dat v experimentech je použit boxplot (krabicový) graf. Tento graf slouží pro popsání dat pomocí kvartilů. Pomocí boxplot grafu jdou snadno vizualizovat důležité informace o jízdě autíčka jako minimální a maximální čas, průměr a medián času a také rozptyl časů v jednotlivých kolech.



Obrázek 4.1: Ukázka boxplot grafu.

[1] Informace na https://plotly.com/python/

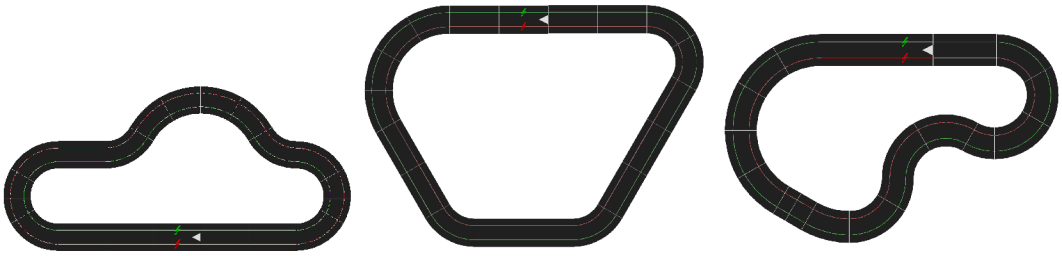
Pro vytvoření ukázkového grafu (obrázek 4.1) jsem použil hodnoty 1, 2, 3, 4, 5, 6, 6, 7, 8 a 9. Tyto hodnoty jsou v grafu vyznačeny jako tečky nalevo od samotného boxplot grafu. Čáry ve tvaru T vystupující z prostředního obdélníku označují minimální a maximální hodnotu. Plná čára uvnitř obdélníku představuje průměrnou hodnotu a přerušovaná čára značí medián. Horní a dolní okraj obdélníku představují společně s mediánem 3 kvartily, které rozdělují data na 4 stejně velké části. Toto rozdělení ale není pro tuto práci podstatné.

Algoritmy používané v experimentech

- Jízda konstantní rychlostí
- Základní algoritmus popsaný v této práci
- Základní algoritmus bez snižování chyby ujeté vzdálenosti
- Modifikace základního algoritmu pro rozpoznávání více prudkostí zatáček
- Modifikace základního algoritmu pro změnu rychlosti v zatáčkách podle aktuální akcelerace
- Algoritmus se snižováním chyby, rozpoznáním více prudkostí zatáček a nejlepšími hodnotami získanými v experimentech

Tvary tratí

Na obrázku 4.2 jsou znázorněny tvary tratí použité v experimentech.



(a) Tvar 1 (b) Tvar 2 (c) Tvar 3

Obrázek 4.2:
Tvar číslo 1: Délka vnitřní dráhy je 530cm a délka vnější dráhy je 592cm.
Tvar číslo 2: Délka vnitřní dráhy je 578cm a délka vnější dráhy je 640cm.
Tvar číslo 3: Délka vnitřní dráhy je 467cm a délka vnější dráhy je 529cm.

Tyto tvary dráhy jsem sestavil, protože obsahují různý poměr a délky rovných úseků a zatáček. Myslím tedy, že jsou dostatečně rozmanité a nachází se na nich rozdílné úseky vhodné pro testování algoritmů.

4.1.1 Rozdíl mezi zaprášenou a čistou dráhou a pneumatikami

V tomto experimentu chci zjistit jaký vliv má zaprášená trať a pneumatiky autíčka na čas projetí kola.

Budu měřit dvě jízdy po 10 kolech ve vnitřní a vnější dráze. Autíčko pojede konstantní rychlostí a jediný rozdíl bude v zaprášené dráze a pneumatikách v prvních dvou jízdách,

a čisté dráze a pneumatikách v druhých dvou jízdách. Rychlost bude nastavena tak, aby autíčko bylo schopné objet dráhu bez vypadnutí.

Naměřené hodnoty

Grafy naměřených hodnot se nachází v příloze v sekci A.1.

Závěr experimentu

Z grafů naměřených hodnot lze vidět vliv čistoty tratě a pneumatik na čas projetí kola. Na vnitřní dráze jsou časy při průjezdu po čisté dráze zřetelně rychlejší a rozptyl časů v jednotlivých kolech je menší. Na vnější dráze jsou časy na očištěné dráze také rychlejší, ale už ne o tolik jako na vnitřní dráze. Tato skutečnost je dána tím, že na vnitřní dráze mají zatáčky menší poloměr než na vnější dráze a autíčko je na špinavé dráze více náchylné k tomu aby dostalo smyk a tím se zpomalil čas průjezdu. Na vnější dráze autíčko při průjezdu zatáčkou po špinavé dráze díky jejímu většímu poloměru nemá takovou tendenci dostat smyk a proto se časy o tolik neliší. Na grafech je také vidět, že i při konstantní rychlosti autíčka a stejných podmínkách se časy jednotlivých jízd liší. V následujících experimentech se snažím aby byly podmínky v jednotlivých jízdách co nejpodobnější a vliv na čas průjezdu měl hlavně použitý algoritmus.

4.1.2 Vliv nastavení rychlostí v jednotlivých sekcích na čas

V tomto experimentu chci získat nejlepší hodnoty rychlostí pro jednotlivé sekce. Autíčko musí zvládnout těmito rychlostmi projet různé tvary tratí bez vypadnutí nebo velkých smyků, které autíčko zpomalují.

Na každém tvaru dráhy autíčko pojede 10 kol ve vnitřní a 10 kol ve vnější dráze při rychlostech, které jsem experimentálně nastavil při implementaci algoritmu. Dále autíčko pojede stejný počet kol při maximálních rychlostech pro daný tvar dráhy, při kterých z dráhy nebude vypadávat nebo dostávat velké smyky.

V tomto a následujících experimentech budu měřit stejně jako v prvním experimentu časy vždy pro dvě jízdy při stejném nastavení na stejné dráze. V prvním experimentu jsou tyto hodnoty vyneseny do grafu každá zvlášť pouze pro ilustraci, že i ve stejných podmínkách jezdí autíčko pokaždé trochu jinak rychle. Od tohoto experimentu budou tyto hodnoty zprůměrovány a do spojnicového grafu vyneseny jako jedna jízda. V box-plot grafu budou všechny časy jednotlivých kol také zahrnuty do jedné jízdy.

Naměřené hodnoty

Grafy naměřených hodnot se nachází v příloze v sekci A.2.

Závěr experimentu

Dosažené časy ukazují, že na všech tvarech tratí byly původně nastavené hodnoty nízké, a autíčko dokáže jezdit ještě rychleji. Při experimentálním nastavování hodnot při vývoji jsem dráhu a pneumatiky autíčka nečistil od prachu a na autíčku byly nazuty staré pneumatiky. Proto byly maximální možné rychlosti nižší. Tyto rychlejší hodnoty zkombinuji tak, aby autíčko zvládlo projet všechny tratě na stejné nastavení.

4.1.3 Vliv způsobu zmenšování chyby ujeté vzdálenosti na čas

Jak už jsem zmínil v sekci o rozpoznání ujetého kola neměří senzor ujeté vzdálenosti úplně přesně. Této chyby snažím zbavit odečítáním hodnoty, která odpovídá chybě, od ujeté vzdálenosti na začátku každého kola.

V tomto experimentu chci zjistit, jak se liší časy autíčka bez jakéhokoliv odstraňování chyby, a s použitím mého přístupu odečítání hodnot.

Hodnoty budu opět měřit na stejných tvarech dráhy a stejným způsobem jako u předchozího experimentu.

Naměřené hodnoty

Grafy naměřených hodnot se nachází v příloze v sekci A.3.

Závěr experimentu

Z naměřených hodnot je vidět, že způsob odstraňování chyby, který je v experimentu použit funguje. Až na jednu výjimku jsou časy v jízdách bez odstraňování chyby pomalejší a mezi časy v jednotlivých kolech jsou větší rozdíly. Kvůli zvětšující se chybě autíčko brzdilo a zrychlovalo na špatných místech a kvůli tomu jelo zbytečně pomalu nebo naopak projelo zatáčkou moc rychle a dostalo smyk, který ho zpomalil. Několikrát se autíčko i úplně zastavilo, protože před brzděním jelo moc pomalu, takže jsem musel provést další měření.

Měření časů bez odstraňování chyby na tvaru dráhy 3 (obr. 4.2c) probíhalo v jiný den než měření časů s odstraňováním chyby, a pravděpodobně vlivem různých podmínek autíčko jelo bez odstraňování chyby rychleji.

Velikost hodnoty, kterou odečítám od ujeté vzdálenosti jsem získal experimentálně. Testováním během implementace jsem zjistil, že hodnota, kterou nyní algoritmus využívá je vhodná pro různé tvary a délky okruhů autodráhy. Tuto hodnotu budu tedy nadále v algoritmu používat.

4.1.4 Algoritmus rozlišující více prudkostí zatáček

V tomto experimentu je algoritmus upraven tak, aby reagoval na více prudkostí zatáček, a podle toho upravil rychlost. Původní algoritmus reaguje pouze na dvě prudkosti zatáček, ve kterých autíčko jede příslušnou rychlostí. Po analýze ladících informací o tvaru tratí jsem zvolil 4 rozsahy prudkosti zatáček. Testováním jsem získal vhodné rychlosti v těchto zatáčkách.

Naměřené hodnoty

Grafy naměřených hodnot se nachází v příloze v sekci A.4.

Závěr experimentu

Naměřené časy ukazují, že detekce více prudkostí zatáček zlepšila čas za kolo. Autíčko zvládá tratě projet v kratším čase. Hodnoty rychlostí na rovinkách a při výjezdech ze zatáčky byly u obou měření stejné, aby se daly tyto dva způsoby jízdy porovnat. Výhodou rozpoznávání více prudkostí zatáček jsou i větší možnosti nastavení rychlostí pro algoritmus. Nadále tedy budu používat tuto modifikovanou verzi algoritmu.

4.1.5 Algoritmus využívající aktuální hodnoty akcelerace v zatáčkách

V tomto experimentu zkouším jiný přístup k jízdě po dráze. Autíčko v zatáčkách pojede podle aktuální akcelerace, a tím by mělo jet v každé zatáčce největší rychlostí, kterou zatáčka dovoluje. Hodnota rychlosti na rovinkách a při výjezdech ze zatáčky zůstane stejná aby bylo možno porovnat časy. Oproti algoritmu, který by měnil rychlost na základně aktuální akcelerace na celé trati má toto řešení výhodu ve znalosti tvaru, a díky tomu schopnosti brzdit před zatáčkami. Mělo by tedy jet rychleji.

Naměřené hodnoty

Grafy naměřených hodnot se nachází v příloze v sekci A.5.

Závěr experimentu

Závislost rychlosti na aktuální akceleraci jsem se snažil nastavit tak, aby autíčko jelo co nejrychleji a zároveň nevypadávalo z dráhy. Rychlost je ale oproti nemodifikované verzi algoritmu nižší, protože autíčko mělo tendenci častěji z dráhy vypadávat. Algoritmus se chová méně předvídatelně než předchozí přístup k změně rychlosti v zatáčkách.

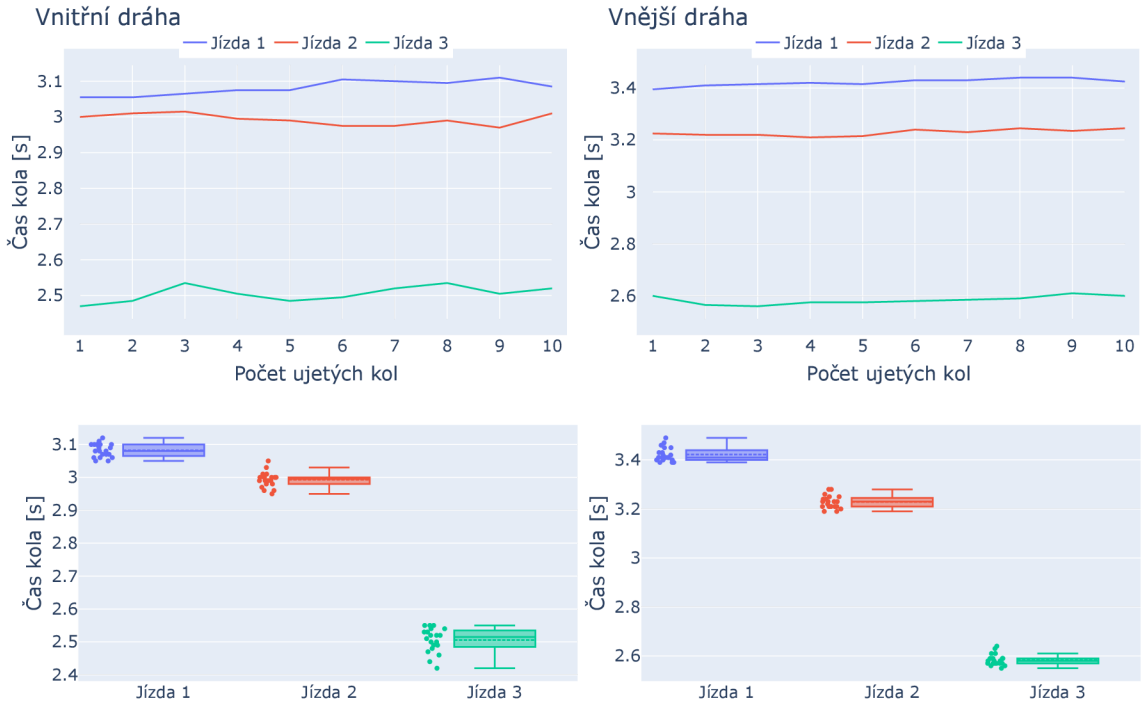
Na tvarech trati 1 a 3 je algoritmus změny rychlosti v zatáčkách podle aktuální akcelerace pomalejší než při požití algoritmu který byl použit v dřívějších experimentech. Pouze na tvaru 2 je autíčko rychlejší než kdyby ho řídil předchozí algoritmus. To je dáno pravděpodobně tím, že trať obsahuje málo krátkých zatáček a více rovinek, takže autíčko neztrácí v zatáčkách tolik času jako u jiných tvarů trati.

Tato modifikace tedy není vhodným vylepšením stávajícího algoritmu a nebudu ji nadále používat.

4.1.6 Porovnání algoritmu s nejlepším nastavením s ostatními algoritmy

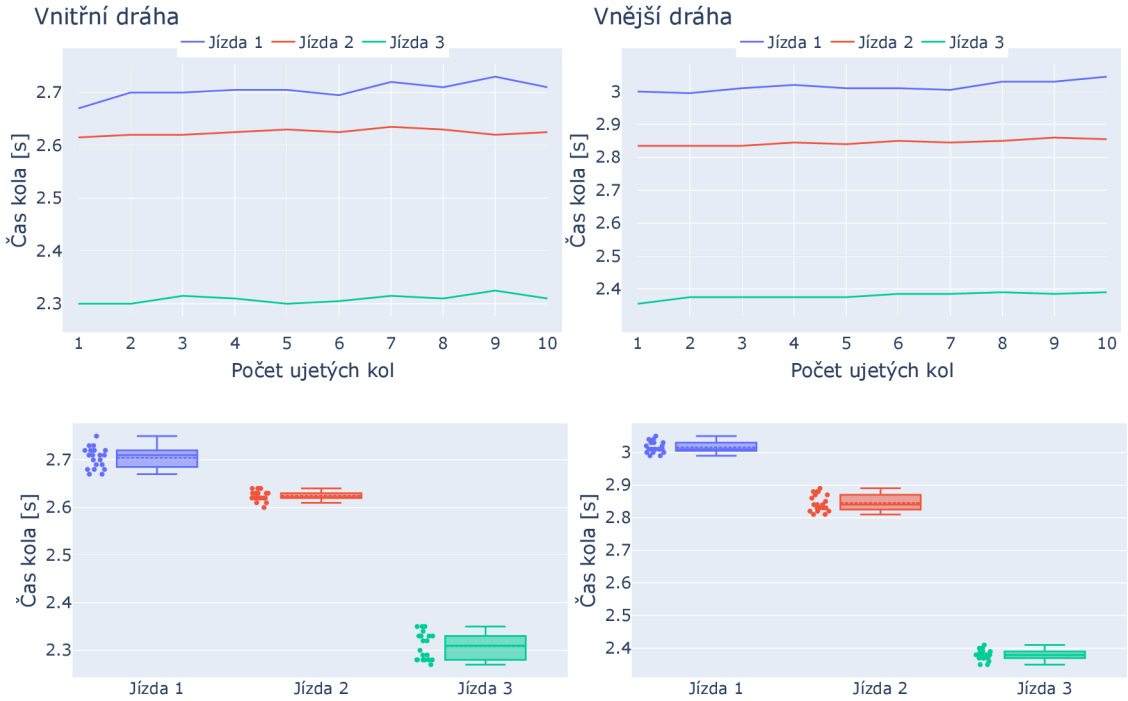
Tento experiment má prokázat, že algoritmus s nejlepším nastavením, získaným v předchozích experimentech, detekcí více prudkostí zatáček a snižováním chyby měření ujeté vzdálenosti zvládá jízdu po různých tvarech dráhy co největší rychlostí. Pro porovnání budou uvedeny časy při jízdě konstantní rychlostí a časy dosažené základním algoritmem uvedeným v této práci.

Naměřené hodnoty



Obrázek 4.3: Hodnoty z tvaru dráhy 1 (obr. 4.2a). Jízda 1 je průměr dvou jízd s konstantní rychlostí. Jízda 2 je průměr dvou jízd s použitím základního algoritmu a jízda 3 je průměr dvou jízd s použitím algoritmu s nejlepšími hodnotami získáními v experimentech.

Obrázek 4.4: Hodnoty z tvaru dráhy 2 (obr. 4.2b). Jízda 1 je průměr dvou jízd s konstantní rychlostí. Jízda 2 je průměr dvou jízd s použitím základního algoritmu a jízda 3 je průměr dvou jízd s použitím algoritmu s nejlepšími hodnotami získáními v experimentech.



Obrázek 4.5: Hodnoty z tvaru dráhy 3 (obr. 4.2c). Jízda 1 je průměr dvou jízd s konstantní rychlostí. Jízda 2 je průměr dvou jízd s použitím základního algoritmu a jízda 3 je průměr dvou jízd s použitím algoritmu s nejlepšími hodnotami získáními v experimentech.

Závěr experimentu

Algoritmus s nejlepším nastavením a modifikací pro rozpoznávání více prudkostí zatáček byl ve všech měřeních nejrychlejší (obrázky 4.3, 4.4, 4.5). Tento experiment tedy potvrzuje, že jsem při nastavování a zlepšování algoritmu postupoval správným směrem. Lepších časů by tento algoritmus mohl dosáhnout pouze s nějakou modifikací nebo rozšířením, to je popsáno v následující kapitole jako součást zhodnocení řešení.

4.1.7 Srovnání časů zajetých na dvou tratích se stejnou délkou ale jiným tvarem

Vnitřní dráha tvaru 1 (obr. 4.2a) a vnější dráha tvaru 3 (obr. 4.2c) mají rozdíl délky 1cm. V tomto experimentu chci zjistit jaký má vliv tvaru dráhy na čas autíčka.

Naměřené hodnoty

Grafy naměřených hodnot se nachází v příloze v sekci A.6.

Závěr experimentu

Autíčko má rychlejší časy na tvaru dráhy 3 (obr. 4.2c), na které se nenachází skoro žádné rovinky a skládá se hlavně ze zatáček. Zatáčky ve vnější dráze jsou méně prudké, než zatáčky na vnitřní dráze tvaru dráhy 1 (obr. 4.2a) a autíčko díky tomu může jet rychleji. Další důvod je pravděpodobně i to, že autíčko nemusí tolik brzdit na konci dlouhé rovinky.

4.2 Zhodnocení řešení

Z naměřených hodnot v experimentech je zřejmé, že autíčko řízené algoritmem navrženým v této práci dokáže projet různé tvary autodráhy velkou rychlostí bez toho, aniž by z dráhy vypadlo. Výsledný algoritmus v experimentu 4.1.5 zajel na všech tvarech dráhy lepší časy než když jelo autíčko konstantní rychlostí nebo používalo nevyladěný základní algoritmus. Z rozptylu časů v jednotlivých kolech je vidět, že si algoritmus vede o něco lépe na vnějších drahách, protože je na nich menší rozptyl časů dvou měřených jízd a také rychlejší čas oproti ostatním způsobům řízení autíčka než ve vnitřních drahách. Důležité je aby autíčko jelo po čisté dráze bez prachu a mělo očištěné i pneumatiky. Jen tak může dosahovat nejlepších výsledků.

Hlavním nedostatkem algoritmu je chyba při měření ujeté vzdálenosti a z toho vyplývající neschopnost přesné detekce ujetého kola. Toto by se dalo vyřešit například optickým snímačem přejetí cílové čáry.

Kapitola 5

Závěr

Cílem této práce bylo navrhnout a implementovat algoritmy pro autonomní mapování neznámého tvaru autodráhy a následnou snahu o dosažení co nejlepšího času s využitím těchto informací.

Po seznámení se s vlastnostmi dostupných autodráh a s jednotlivými ročníky soutěže Freescale Race Challenge byly popsány principy související s řízením motoru a zpracování dat z čidel využitelných pro mapování neznámého tvaru autodráhy a jízdě autonomní jízdu. Dále bylo zvoleno Arduino jako vhodný mikrokontroler pro autonomní řízení a byly vybrány další potřebné komponenty, které byly následně zapojeny a vyzkoušeny. Také byla navrhnuta datová reprezentace dráhy a algoritmy pro mapování neznámého tvaru dráhy a autonomní jízdu využívající informací o tvaru dráhy pro její projetí v co nejkratším čase.

Dráha je reprezentována pomocí pole sekcí trati. Sekce se dělí na rovný úsek, brzdnou zónu, zatáčku a výjezd ze zatáčky. U každé sekce je uložena vzdálenost jejího začátku a konce od startu. U zatáček navíc i jejich prudkost. Algoritmus pro autonomní jízdu následně podle senzoru ujeté vzdálenosti určí v které sekci se autíčko právě nachází a podle toho nastaví rychlost.

Tyto algoritmy byly následně implementovány, a otestovány pomocí několika experimentů. Experimenty ukázaly, že autíčko je schopné rychlé jízdy bez vypadávání na různých tvarech dráhy se znalostí pouze její délky. Dále také experimenty ukázaly rozdíly dosažených časů autíčka při různých podmínkách a různým modifikacích algoritmu. Cíl práce byl tedy splněn.

Díky této práci jsem si vyzkoušel vývoj pokročilejšího programu pro moderní Arduino se zajímavým tématem autonomního řízení.

V práci by bylo možno pokračovat přidáním senzoru detekujícího cílovou čáru, díky kterému by bylo měření ujeté vzdálenosti přesnější než nyní. S přidaným senzorem by bylo možné vyzkoušet některou z modifikací algoritmu uvedených v sekci 3.4, které by mohly časy autíčka ještě snížit. Dále by se také daly vyzkoušet jiné přístupy k fungování algoritmů jako například použití neuronových sítí a strojového učení.

Literatura

[1] Arducam. *Arducam OV7670 0.3 Megapixel Camera Module for Arduino Boards* [online]. Arducam, 2019 [cit. 2020-27-04]. Dostupné z: https://www.arducam.com/product/arducam-ov7670-0-3-megapixel-camera-module-for-arduino-boards/.

[2] Arduino. *Getting started with the Arduino NANO 33 BLE* [online]. Arduino, 2019 [cit. 2020-06-1]. Dostupné z: https://www.arduino.cc/en/Guide/NANO33BLE.

[3] Arduino. *Arduino Nano 33 BLE* [online]. Arduino, 2020 [cit. 2020-06-1]. Dostupné z: https://store.arduino.cc/arduino-nano-33-ble.

[4] Brejl, M. *Freescale Race Challenge 2009 - Soutěž samořídících autíček na autodráhu* [online]. HW server s.r.o., 2008 [cit. 2019-13-12]. Dostupné z: https://vyvoj.hw.cz/teorie-a-praxe/freescale-race-challenge-2009-soutez-samoridicich-auticek-na-autodrahu.html.

[5] Brejl, M. *Freescale Race Challenge 2010 - Nový ročník soutěže samořídících autíček na autodráhu* [online]. HW server s.r.o., 2009 [cit. 2019-18-12]. Dostupné z: https://vyvoj.hw.cz/procesor/freescale-race-challenge-2010-novy-rocnik-souteze-samoridicich-auticek-na-autodrahu.html.

[6] Brejl, M. *Freescale Race Challenge 2011 - Další ročník soutěže samořídících autíček na autodráhu* [online]. HW server s.r.o., 2010 [cit. 2019-18-12]. Dostupné z: https://vyvoj.hw.cz/teorie-a-praxe/freescale-race-challenge-2011-dalsi-rocnik-souteze-samoridicich-auticek-na-autodrahu.

[7] Brejl, M. *Freescale Race Challenge 2012: Soutěž samořídících autíček na autodráhu pokračuje* [online]. HW server s.r.o., 2011 [cit. 2019-18-12]. Dostupné z: https://vyvoj.hw.cz/teorie-a-praxe/mimochodem/freescale-race-challenge-2012-soutez-samoridicich-auticek-na-autodrahu.

[8] Carrera. *Audi R8 GT LMS United Autosports No.23* [online]. Carrera, 2020 [cit. 2020-27-04]. Dostupné z: https://www.carrera.cz/katalog/vyrobci/carrera/produkt/audi-r8-gt-lms-united-autosports-no.23_1970.

[9] Carrera. *Carrera EVOLUTION* [online]. Carrera, 2020 [cit. 2020-29-04]. Dostupné z: https://www.carrera-toys.com/en/1492/evolution.

[10] Carrera. *Díly dráhy pro autodráhu Carrera Evolution* [online]. Carrera, 2020 [cit. 2020-27-04]. Dostupné z: https://www.carrera.cz/m/katalog/87_carrera-evolution-dily-drahy.

[11] Carrera. *Evolution Transformer EU* [online]. Carrera, 2020 [cit. 2020-29-04]. Dostupné z: https://www.carrera-toys.com/en/product/20026710-evolution-transformer-eu.

[12] CTS® Corporation. *Series 291 Precision, Long-life 12mm Optical Encoder*. D. červenec 2017.

[13] Dixon Warren, D. *Motion sensing in the iPhone 4: MEMS gyroscope* [online]. MEMS Journal, 2011 [cit. 2020-27-04]. Dostupné z: https://www.memsjournal.com/2011/01/motion-sensing-in-the-iphone-4-mems-gyroscope.html.

[14] Freescale. *+-1.5g, +-6g Three Axis Low-g Micromachined Accelerometer*. 1. vyd. Duben 2008.

[15] Freescale. *MCF51JM128 ColdFire Microcontroller*. 4. vyd. Květen 2012.

[16] Freescale. *MC9S08JM60 Series Data Sheet*. 5. vyd. říjen 2014.

[17] He, J., Zhou, W., He, X. a Peng, P. Structural Designing of a MEMS Capacitive Accelerometer for Low Temperature Coefficient and High Linearity. *Sensors*. Únor 2018, sv. 18, s. 643. DOI: 10.3390/s18020643.

[18] InvenSense, Inc.. *Integrated Dual-Axis Gyro IDG-1215*. 2008.

[19] Microchip technology. *24AA512/24LC512/24FC512 512K I2C^{TM} Serial EEPROM*. 1. vyd. Leden 2010.

[20] NXP. *MC33887 5.0 A H-bridge with load current feedback*. 17. vyd. Září 2018.

[21] NXP. *MC33931 5.0 A throttle control H-bridge*. 5. vyd. Září 2018.

[22] Sparkfun. *SparkFun microSD Transflash Breakout* [online]. Sparkfun, 2010 [cit. 2020-06-1]. Dostupné z: https://www.sparkfun.com/products/544.

[23] Sparkfun. *SparkFun RGB Light Sensor - ISL29125* [online]. Sparkfun, 2014 [cit. 2020-27-04]. Dostupné z: https://www.sparkfun.com/products/12829.

[24] Sparkfun. *SparkFun Motor Driver - Dual TB6612FNG (1A)* [online]. Sparkfun, 2018 [cit. 2020-27-04]. Dostupné z: https://www.sparkfun.com/products/14451.

[25] STMicroelectronics. *LSM9DS1 iNEMO inertial module:3D accelerometer, 3D gyroscope, 3D magnetometer*. 2. vyd. Listopad 2014.

[26] Texas Instruments. *DRV5056-Q1 Automotive Unipolar Ratiometric Linear Hall Effect Sensor*. 2. vyd. Srpen 2018.

[27] Toshiba. *Toshiba Bi-CD Integrated Circuit Silicon Monolithic TB6612FNG Driver IC for Dual DC motor*. 1. vyd. Leden 2014.

[28] Wikipedia contributors. *Electric motor* [online]. Wikipedia, The Free Encyclopedia., 2020. 27 April 2020 22:59 [cit. 2020-27-04]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Electric_motor&oldid=953422825.

[29] Wikipedia contributors. *Induction motor* [online]. Wikipedia, The Free Encyclopedia., 2020 [cit. 2020-27-04]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Induction_motor&oldid=952479207.

[30] Wikipedie. *Hybnost — Wikipedie: Otevřená encyklopedie*. 2018. [Online; navštíveno 6. 05. 2020]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Hybnost&oldid=16689172.

[31] Wikipedie. *Odstředivá síla — Wikipedie: Otevřená encyklopedie*. 2019. [Online; navštíveno 6. 05. 2020]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Odst%C5%99ediv%C3%A1_s%C3%ADla&oldid=16982066.

[32] Wikipedie. *Tření — Wikipedie: Otevřená encyklopedie*. 2019. [Online; navštíveno 6. 05. 2020]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=T%C5%99en%C3%AD&oldid=17865978.

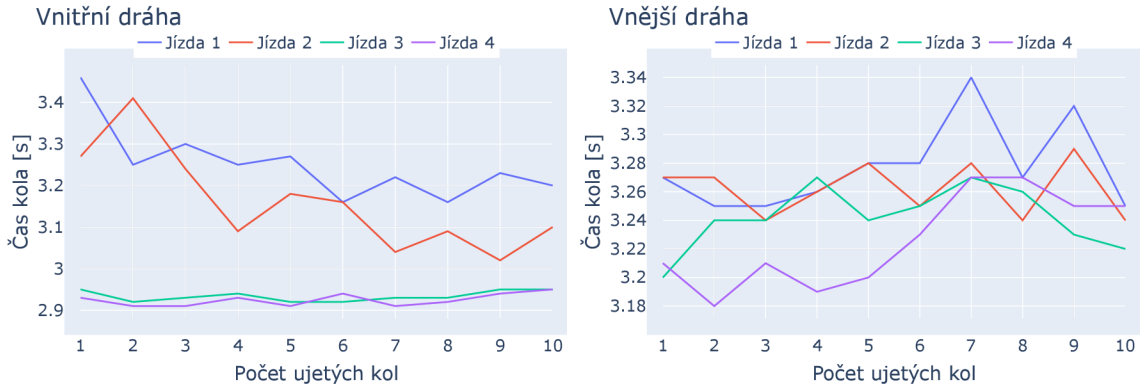
[33] Wikipedie. *Rychlost — Wikipedie: Otevřená encyklopedie*. 2020. [Online; navštíveno 6. 05. 2020]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Rychlost&oldid=18309605.

[34] Wikipedie. *Zrychlení — Wikipedie: Otevřená encyklopedie*. 2020. [Online; navštíveno 6. 05. 2020]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Zrychlen%C3%AD&oldid=18458109.

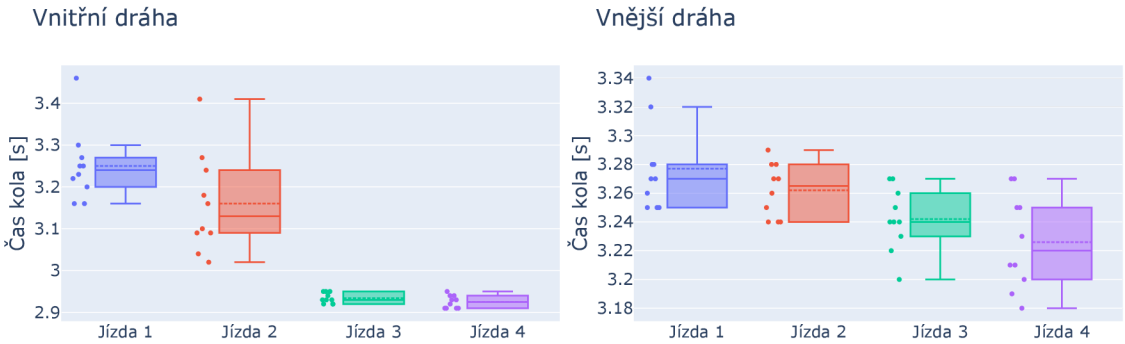
Příloha A

Data naměřená v experimentech

A.1 Rozdíl mezi zaprášenou a čistou dráhou a pneumatikami

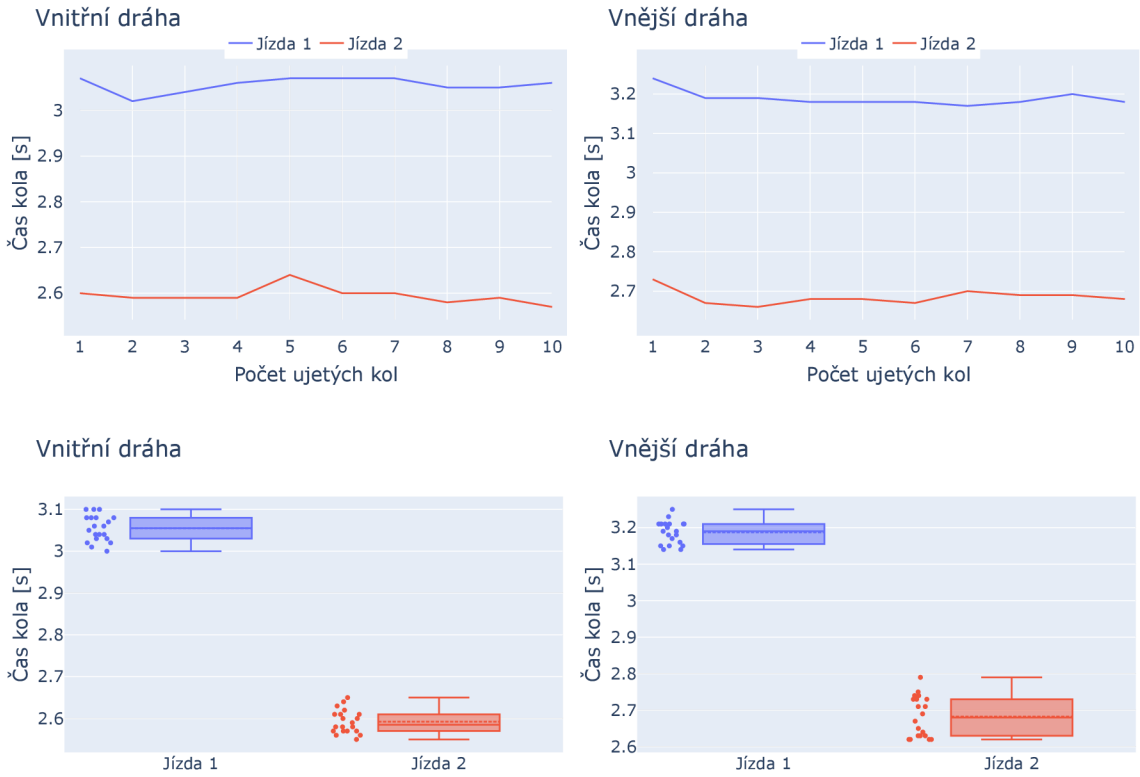


Obrázek A.1: Trend doby průjezdu v závislosti na počtu ujetých kol. Jízdy 1 a 2 probíhaly na zaprášené trati, jízdy 3 a 4 probíhaly na očištěné trati.

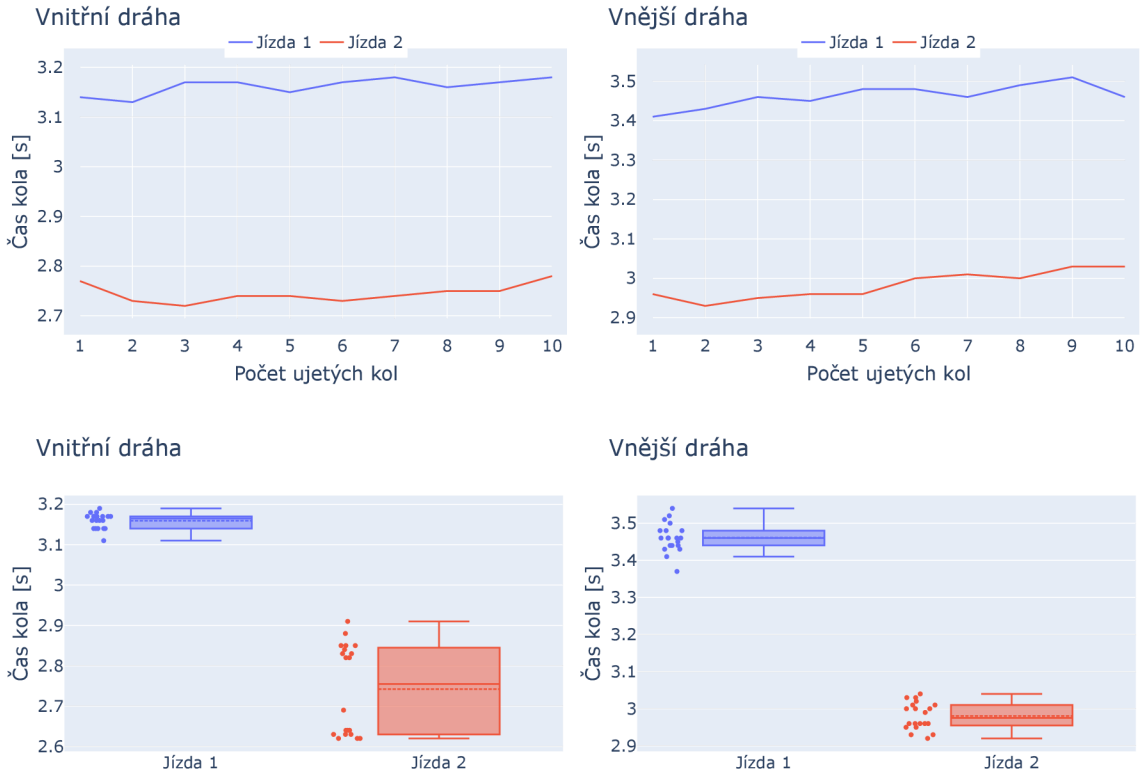


Obrázek A.2: Statistická reprezentace časů v jednotlivých jízdách. Jízdy 1 a 2 probíhaly na zaprášené trati, jízdy 3 a 4 probíhaly na očištěné trati.

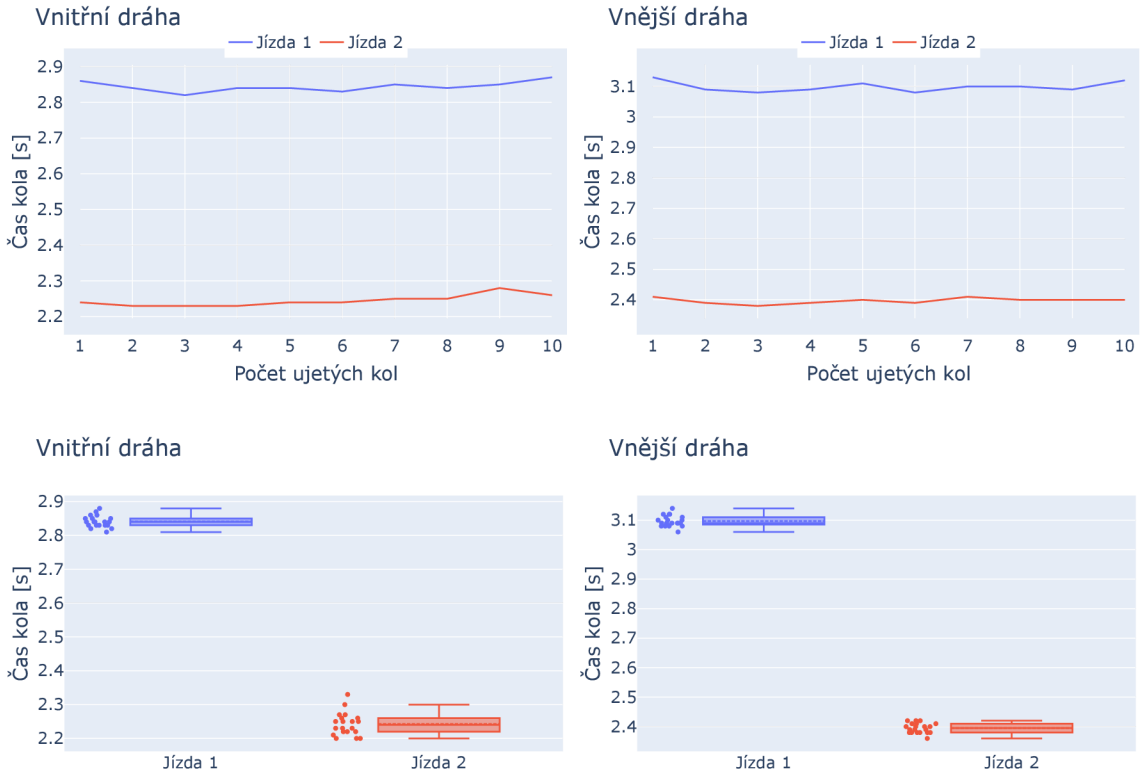
A.2 Vliv nastavení rychlostí v jednotlivých sekcích na čas



Obrázek A.3: Hodnoty z tvaru dráhy 1 (obr. 4.2a). Jízda 1 je průměr dvou jízd při základních hodnotách rychlostí v sekcích, jízda 2 je průměr dvou jízd při maximálních rychlostech v sekcích.

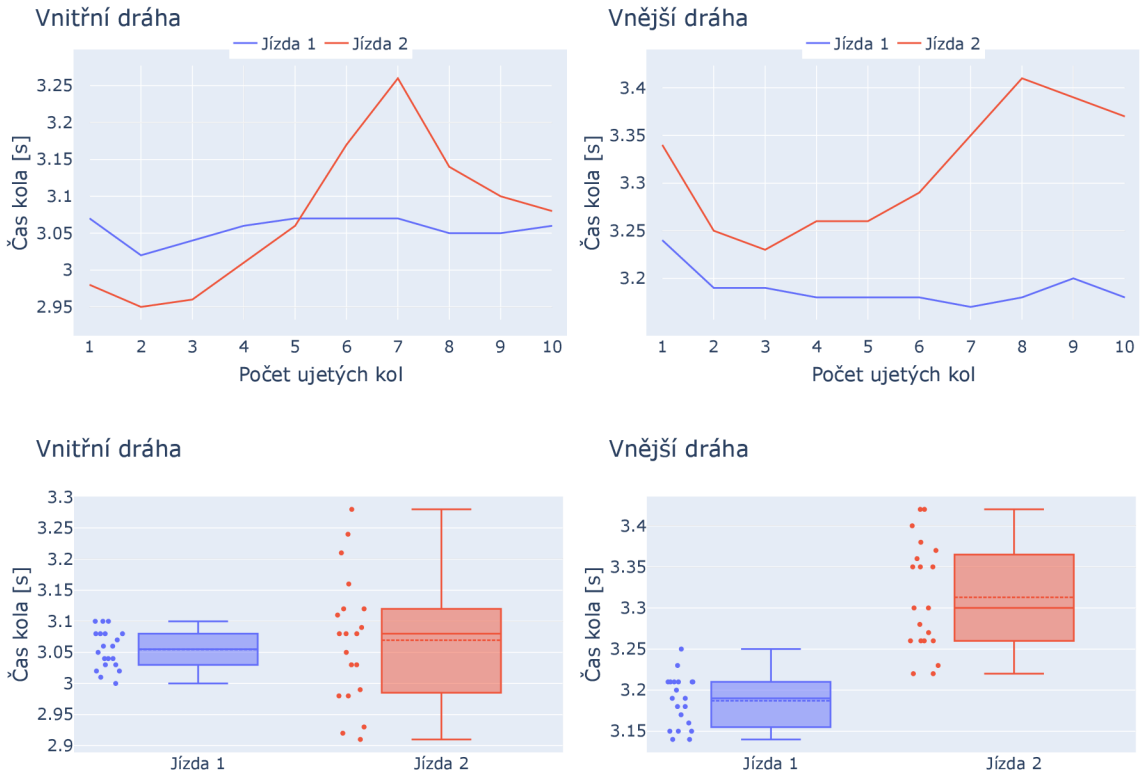


Obrázek A.4: Hodnoty z tvaru dráhy 2 (obr. 4.2b). Jízda 1 je průměr dvou jízd při základních hodnotách rychlostí v sekcích, jízda 2 je průměr dvou jízd při maximálních rychlostech v sekcích.

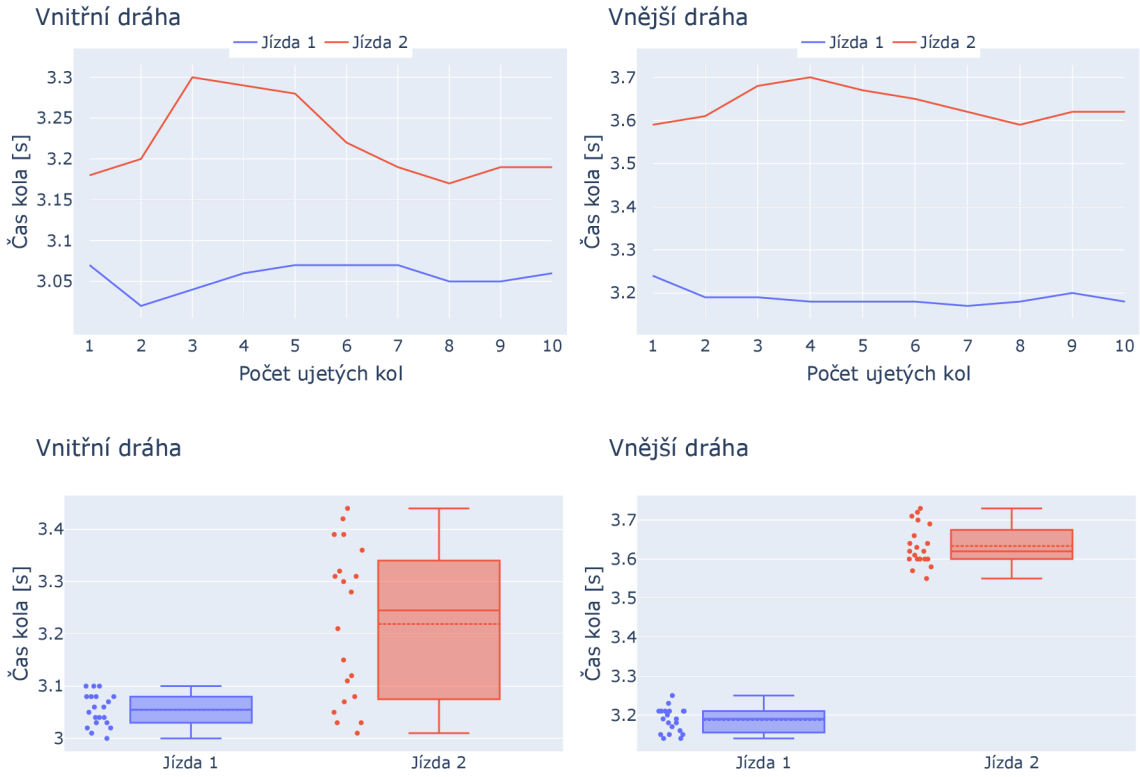


Obrázek A.5: Hodnoty z tvaru dráhy 3 (obr. 4.2c). Jízda 1 je průměr dvou jízd při základních hodnotách rychlostí v sekcích, jízda 2 je průměr dvou jízd při maximálních rychlostech v sekcích.

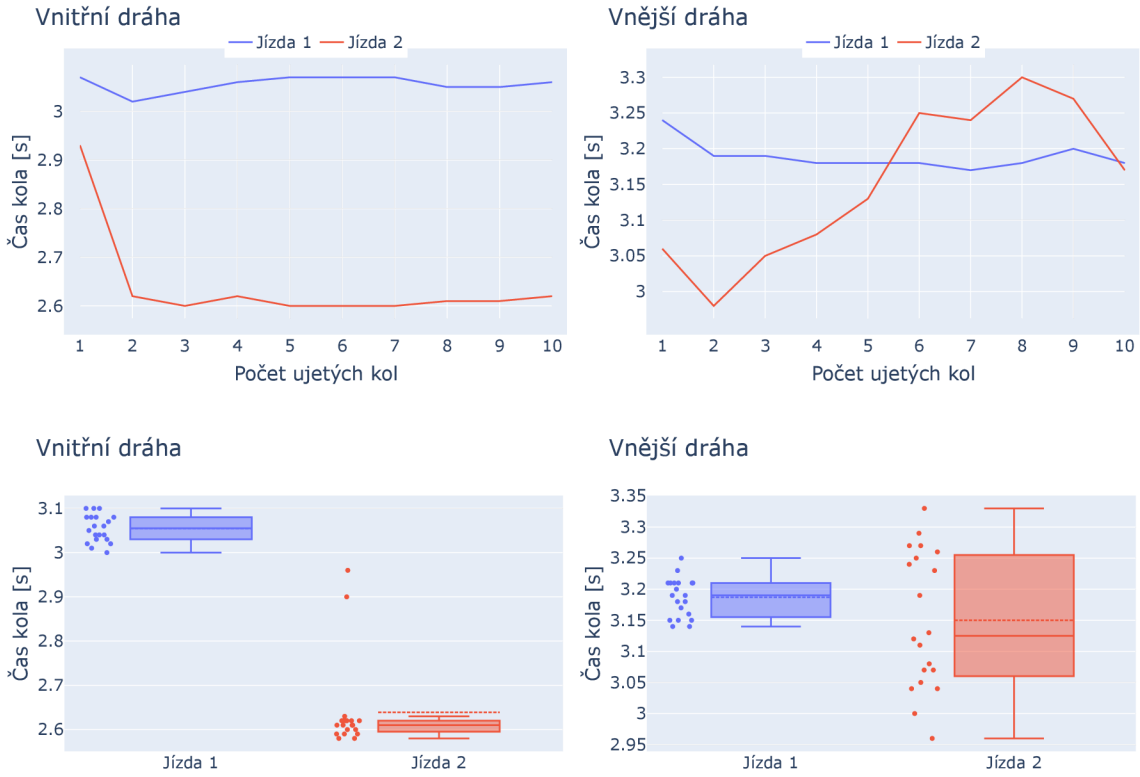
A.3 Vliv způsobu zmenšování chyby ujeté vzdálenosti na čas



Obrázek A.6: Hodnoty z tvaru dráhy 1 (obr. 4.2a). Jízda 1 je průměr dvou jízd s odstraňováním chyby, jízda 2 je průměr dvou jízd bez odstraňování chyby.

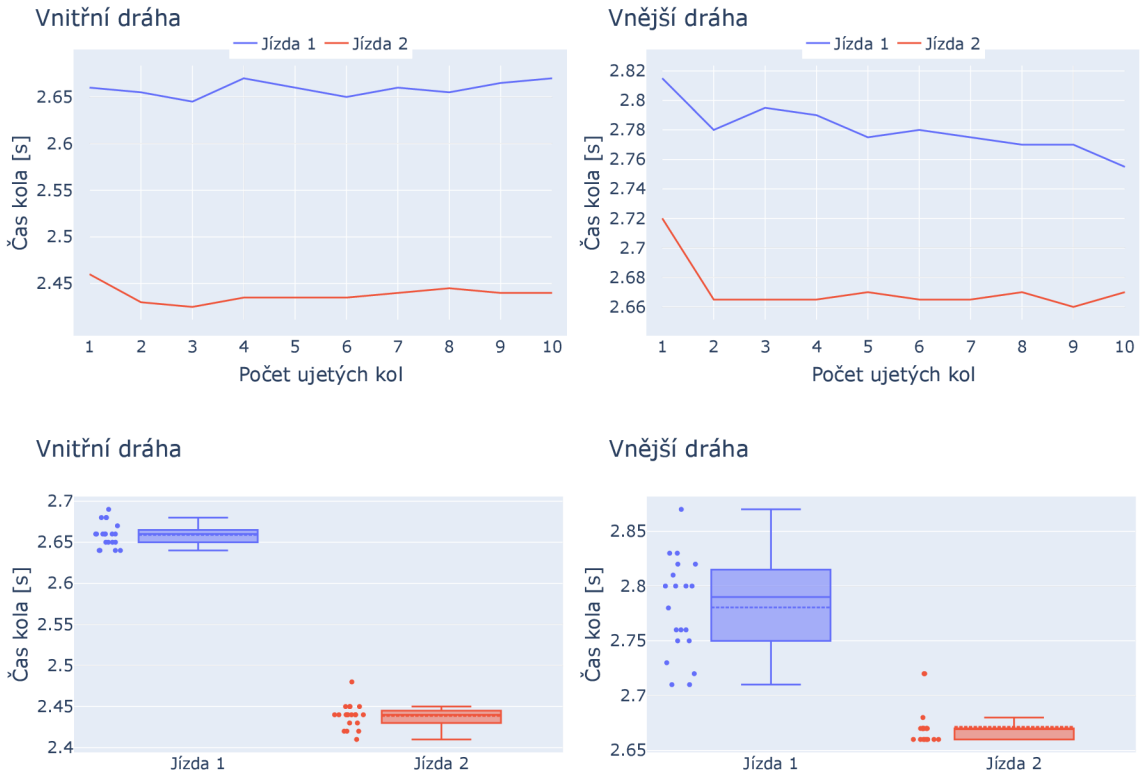


Obrázek A.7: Hodnoty z tvaru dráhy 2 (obr. 4.2b). Jízda 1 je průměr dvou jízd s odstraňováním chyby, jízda 2 je průměr dvou jízd bez odstraňování chyby.

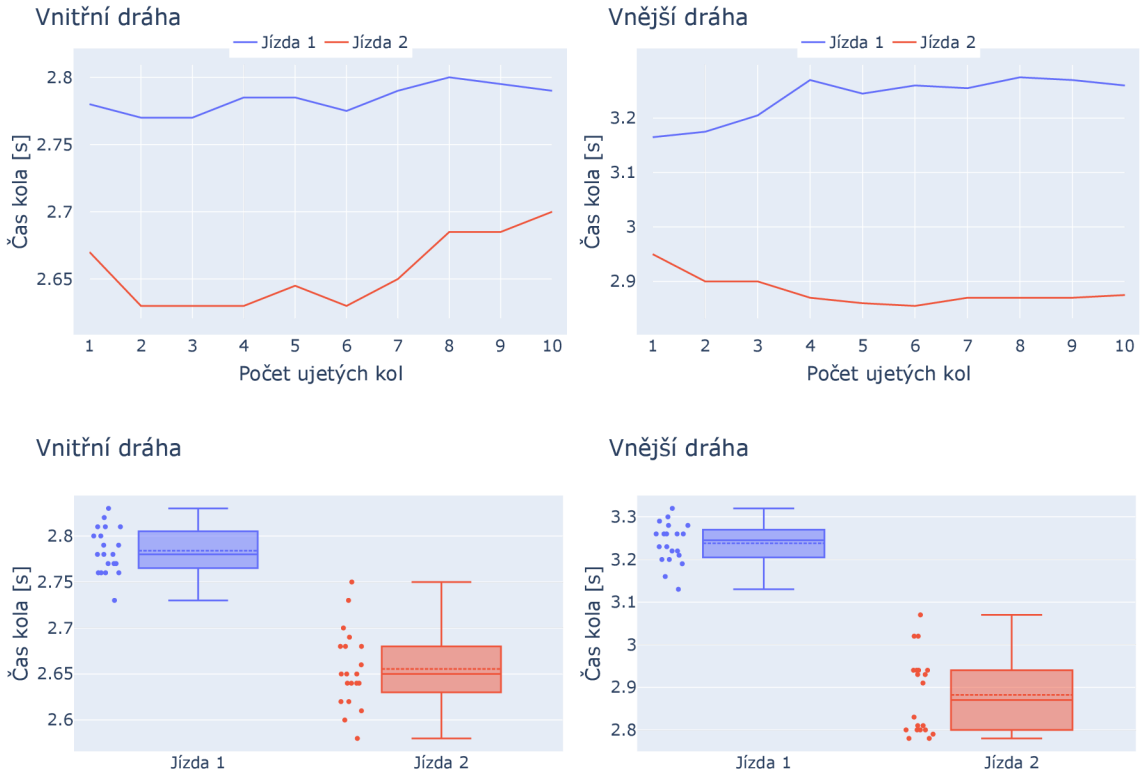


Obrázek A.8: Hodnoty z tvaru dráhy 3 (obr. 4.2c). Jízda 1 je průměr dvou jízd s odstraňováním chyby, jízda 2 je průměr dvou jízd bez odstraňování chyby.

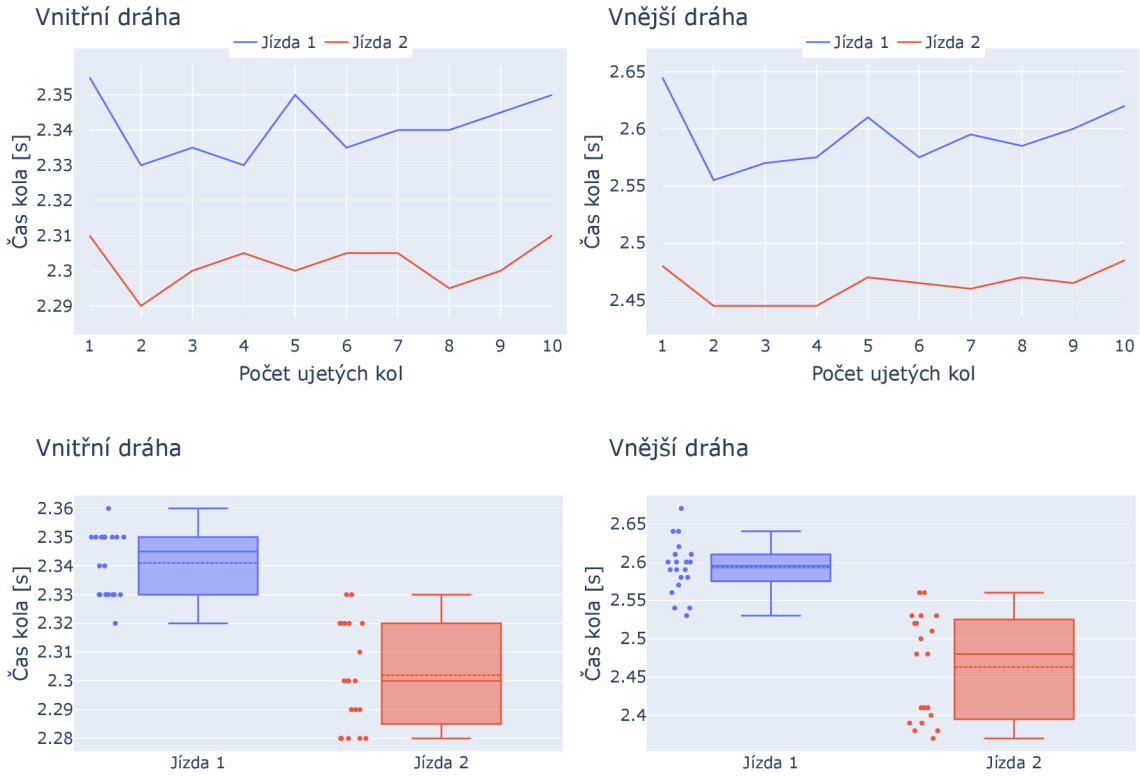
A.4 Algoritmus rozlišující více prudkostí zatáček



Obrázek A.9: Hodnoty z tvaru dráhy 1 (obr. 4.2a). Jízda 1 je průměr dvou jízd s rozlišováním 2 prudkostí zatáček, jízda 2 je průměr dvou jízd s rozlišováním 4 prudkostí zatáček.

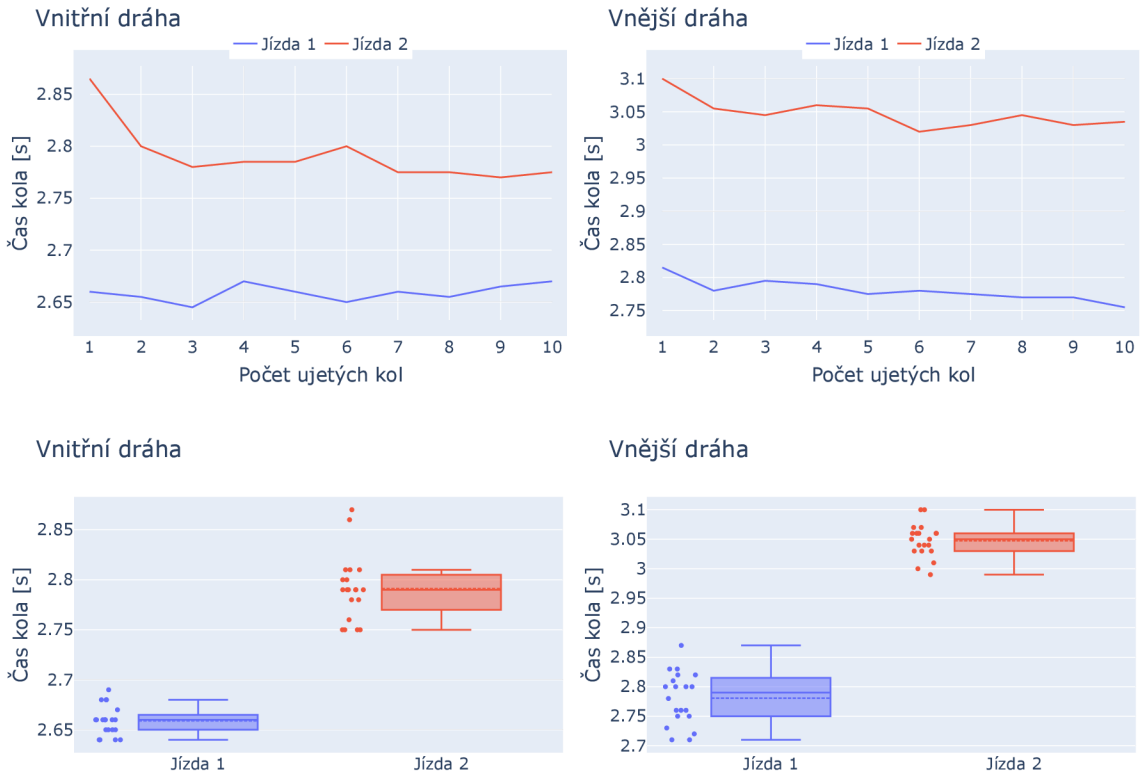


Obrázek A.10: Hodnoty z tvaru dráhy 2 (obr. 4.2b). Jízda 1 je průměr dvou jízd s rozlišováním dvou prudkostí zatáček, jízda 2 je průměr dvou jízd s rozlišováním čtyř prudkostí zatáček.

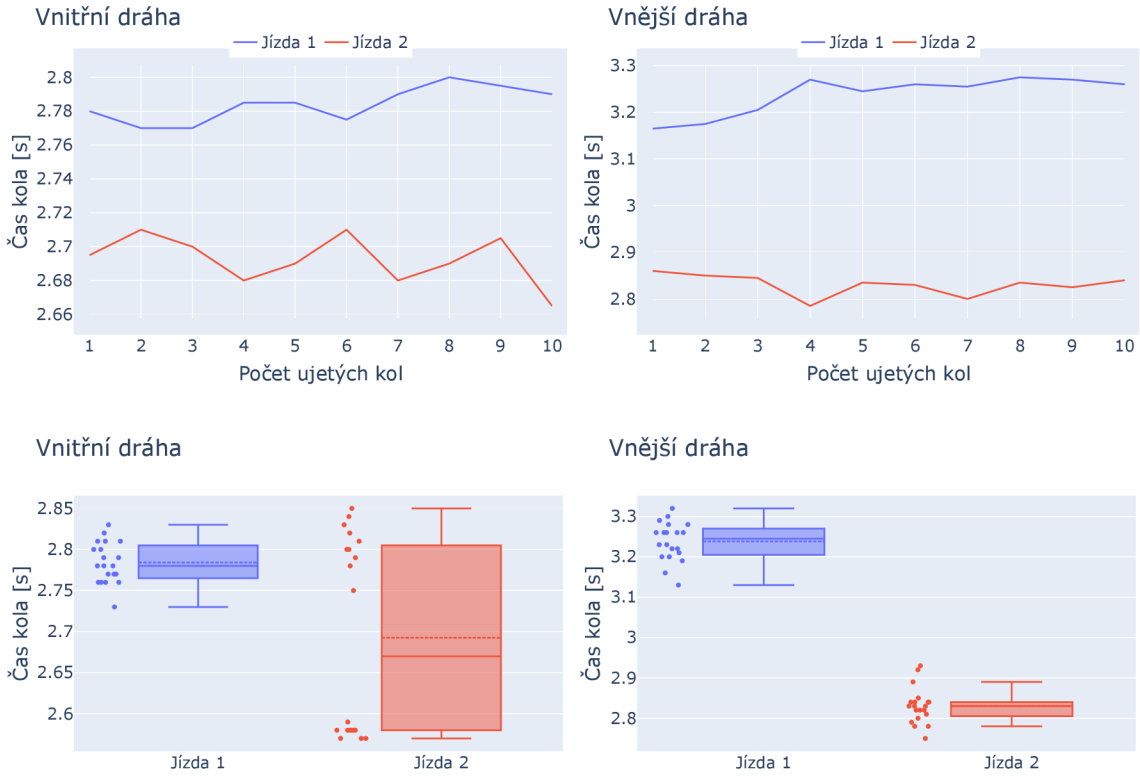


Obrázek A.11: Hodnoty z tvaru dráhy 3 (obr. 4.2c). Jízda 1 je průměr dvou jízd s rozlišováním 2 prudkostí zatáček, jízda 2 je průměr dvou jízd s rozlišováním 4 prudkostí zatáček.

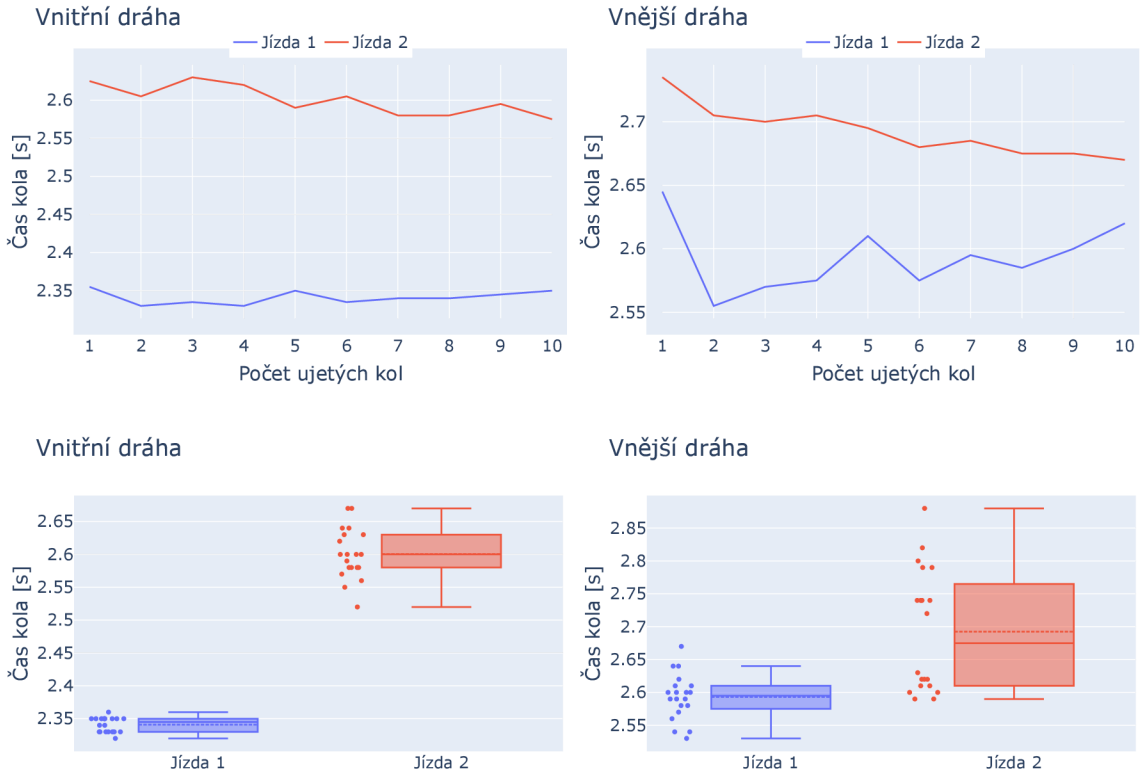
A.5 Algoritmus využívající aktuální hodnoty akcelerace v zatáčkách



Obrázek A.12: Hodnoty z tvaru dráhy 1 (obr. 4.2a). Jízda 1 je průměr dvou jízd s rozlišováním 2 prudkostí zatáček, jízda 2 je průměr dvou jízd s rychlostí v zatáčkách závislou na aktuální akceleraci.

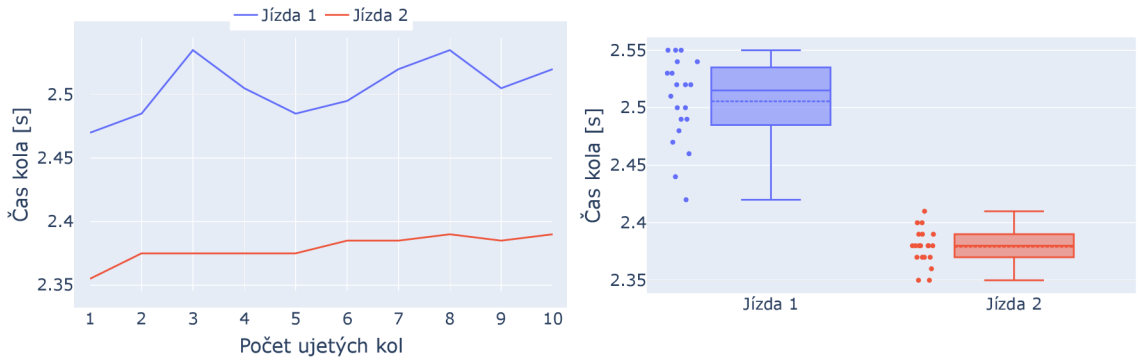


Obrázek A.13: Hodnoty z tvaru dráhy 2 (obr. 4.2b). Jízda 1 je průměr dvou jízd s rozlišováním 2 prudkostí zatáček, jízda 2 je průměr dvou jízd s rychlostí v zatáčkách závislou na aktuální akceleraci.



Obrázek A.14: Hodnoty z tvaru dráhy 3 (obr. 4.2c). Jízda 1 je průměr dvou jízd s rozlišováním 2 prudkostí zatáček, jízda 2 je průměr dvou jízd s rychlostí v zatáčkách závislou na aktuální akceleraci.

A.6 Srovnání časů zajetých na dvou tratích se stejnou délkou ale jiným tvarem



Obrázek A.15: Jízda 1 je průměr dvou jízd na vnitřní dráze tvaru dráhy 1 (obr. 4.2a) a jízda 2 je průměr dvou jízd na vnější dráze tvaru dráhy 3 (obr. 4.2c). Tyto dvě dráhy mají skoro stejnou délku.

Příloha B

Obsah přiloženého média

Na přiloženém datovém médiu se nachází:

- soubor BP_xnesva06.pdf - elektronická podoba této práce
- složka latex - zdrojový kód této práce v jazyku LaTeX
- složka src - zdrojový kód programu
- složka experiments - naměřené hodnoty v experimentech
- soubor README.txt - popis a další informace o datech na médiu