Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

PLZEŇ, 2015 **PETR BATĚK**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr BATĚK**
Osobní číslo:	**A12B0694P**
Studijní program:	**B3918 Aplikované vědy a informatika**
Studijní obor:	**Kybernetika a řídicí technika**
Název tématu:	**Návrh, modelování a řízení mobilního robotu s Ackermanovým řízením**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování:

1. Vytvořte matematický model mobilního robotu.
2. Navrhněte řídicí algoritmus mobilního robotu.
3. Navrhněte HW řídicího systému mobilního robotu.

Rozsah grafických prací: **dle potřeby**

Rozsah pracovní zprávy: **30-40 stránek A4**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Miroslav Flídr, Ph.D.**
Katedra kybernetiky

Datum zadání bakalářské práce: **1. listopadu 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

Doc. RNDr. Miroslav Lávička, Ph.D.
děkan

L.S.

Prof. Ing. Josef Psutka, CSc.
vedoucí katedry

V Plzni dne 1. listopadu 2014

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 10. 5. 2015 .....

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu bakalářské práce panu Ing. Miroslavu Flídrovi, Ph.D. Jeho rady a připomínky pro mě byly cennými podněty, které významně přispěly ke zdárnému dokončení této práce.

Anotace

Bakalářská práce pojednává o návrhu a vývoji řízení autonomního robotu. Popisovány jsou nejrozšířenější typy podvozků kolových vozidel, jejich matematické modely a zvláštní důraz je kladen na roboty s Ackermanovým řízením. Práce se také věnuje elektronickým akčním členům, řídícím jednotkám a jejich vzájemnému softwarovému propojení. Výsledkem práce je autonomní mobilní robot, který tvoří tzv. embedded systém řízený mikroprocesorem z řady ARM Cortex-M.

Klíčová slova: Autonomní robot, inteligentní auto, sledování čáry, vestavěný systém

Abstract

The bachelor's thesis discusses a design and a development of an autonomous robot control. The most common types of wheeled vehicle chassis are described with their mathematical models. A special emphasis is placed on robots with Ackermann steering. The work deals with electronic actuators, control units and their mutual software interconnection. The result is an autonomous mobile robot, which forms an embedded system cotrolled by a microprocessor ARM Cortex-M.

Keywords: Autonomous robot, intelligent car, line following, embedded system

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Vývoj autonomních robotů a automatizovaných systémů je jeden z hlavních zájmů moderní technologie. Obecnou snahou je přesunout potřebu lidské práce do automatizovaného světa strojů. Nové technologie dnes umožňují řešení komplexních problémů, zvyšují naši bezpečnost a v neposlední řadě výrazným způsobem snižují ekonomické náklady v mnoha oblastech.

Díky stále výkonnějším zařízením je nyní možné regulovat a řídit i složité systémy, o jejichž správnou činnost se dříve museli starat výhradně lidští experti. Jedním z příkladů jsou stále propracovanější systémy autopilotů letadel. Tyto systémy jsou dnes schopny v podstatě zcela samostatně řídit celý průběh letu. Piloti dnes nad těmito systémy především dohlíží a řízení přebírají jen ve výjimečných případech. Je jen otázkou času, než se podobné systémy přemístí i do vozidel. Již dnes fungují v automobilech tempomaty a parkovací asistenty, jedny z prvních automatizovaných systémů pro vozidla.

Výrobou takových řídících systémů se zabývá mnoho technologicky vyspělých firem. Jednou z nich je firma Freescale Semiconductor, Inc. Tato společnost se věnuje vývoji embedded systémů pro automobilové, spotřebitelské a průmyslové trhy. Firma navíc spolupracuje s univerzitami po celém světě a pořádá univerzitní soutěže ve vývoji automatických systémů. Jednou z nich je Freescale Cup. Jedná se o závody autonomně řízených modelů aut. Program nabízí mnoho výukových materiálů z nejlepších světových univerzit jako je například MIT. Je to tedy ideální prostředek k čerpání nových informací.

Cílem této bakalářské práce bylo vyvinout řídící systém pro autonomní vozidlo do soutěže. Závod spočívá v co nejrychlejším průjezdu předem neznámého okruhu. Trať je vymezena černou čarou na bílém povrchu.

Sledování určité dráhy lze pojmout mnoha způsoby. Jedním z nich je klasické sledování čáry. V těchto úlohách se běžně používá robot s diferenciálním řízením. V rámci soutěže Freescale Cup však závodí modely aut, které využívají Ackermanova způsobu řízení. Problém sledování čáry lze rozšířit na úlohu sledování dráhy. Dráha je vymezena dvěma čarami a cílem je využít prostoru mezi nimi (dráhy) k objetí závodního okruhu.

Úloha sledování dráhy je mnohem komplexnější a vyžaduje hlubší analýzu než sledování jedné čáry. V soutěži Freescale Cup se až do roku 2014 závodilo na trati s jednou čarou, letos se však obměnila pravidla právě na sledování dráhy.

Vývoj řídícího systému bylo nutné začít od základních úloh, a proto tato práce pojednává o návrhu systému pro sledování jedné čáry a následné modifikaci pro problém sledování dráhy.

Tato práce je rozdělena do 7 kapitol. První 3 kapitoly jsou zaměřeny na popis řešeného problému a hardwaru robota, na kterém byla celá práce realizována. Další kapitola pojednávává o zpracování signálu z optického senzoru ve formě řádkové kamery. Následně je popsán návrh řídícího algoritmu pro řešené problémy. Předposlední kapitola je věnována průběhu nastavování systému a vše je zakončeno krátkým popisem možností dalšího vylepšování autonomního robota.

Kapitola 2

Konstrukce robota

Důležitou vlastností robota je jeho konstrukce. Od té se dále odvíjejí možnosti pohybu robota a jeho řízení. Kolová vozidla mají jinou realizaci řídícího systému než například humanoidní roboti. Protože se tato práce zabývá návrhem řízení pro autonomní auto, je následující kapitola věnována modelům kolových vozidel.

V této části budou popsány matematické modely různých typů podvozků kolových robotů.

2.1 Modely kolových robotů

Kolová vozidla lze rozdělit do tří základních skupin:

- Všesměrová vozidla
- Vozidla s diferenciálním řízením
- Vozidla s Ackermanovým řízením

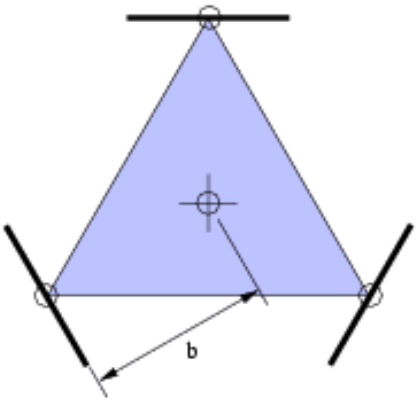
Každý typ podvozku má své výhody, nevýhody a specifické aplikace.

Hlavní důraz je v této kapitole zaměřen na druhou a třetí skupinu vozidel, protože realizační platforma má kromě Ackermanova řízení také nezávisle hnaná kola na zadní nápravě. Této vlastnosti lze využít pro návrh diferenciálního řízení zadní nápravy vozidla.

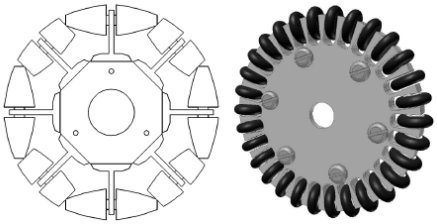
2.1.1 Všesměrová vozidla

Roboti s tímto typem podvozku se dokáží pohybovat všemi směry. Často jsou také označováni pod termínem *omnidrive* [3]. Omnidrive roboti patří do skupiny tzv. holonomních modelů. Slovo *holonomní* označuje skutečnost, že počet stupňů volnosti rychlosti vozidla odpovídá počtu stupňů volnosti pozice. Znamená to, že holonomní vozidlo může měnit svoji rychlost nezávisle ve všech směrech.

Příkladem holonomního vozidla může být například vznášedlo nebo všesměrový podvozek. Tyto vozidla nic neomezuje ani při pohybu do stran. Podvozek omnidrive je zobrazen na obrázku 2.1.



Obrázek 2.1: **Model všesměrového robota** - zdroj: www.robotika.cz



Obrázek 2.2: **Všesměrové kolo** - zdroj: www.robotika.cz

2.1.2 Vozidla s diferenciálním řízením

Změna orientace robota závisí na rozdílu rychlosti levého a pravého kola či pásu [11]. Vozidla s tímto podvozkem jsou schopna otáčet se kolem vlastní osy, ale nemohou se pohybovat do stran. Proto spadají do kategorie neholonomních vozidel. Aby bylo možné zajistit různou rychlost kol, musí být každé z nich nezávisle poháněné.

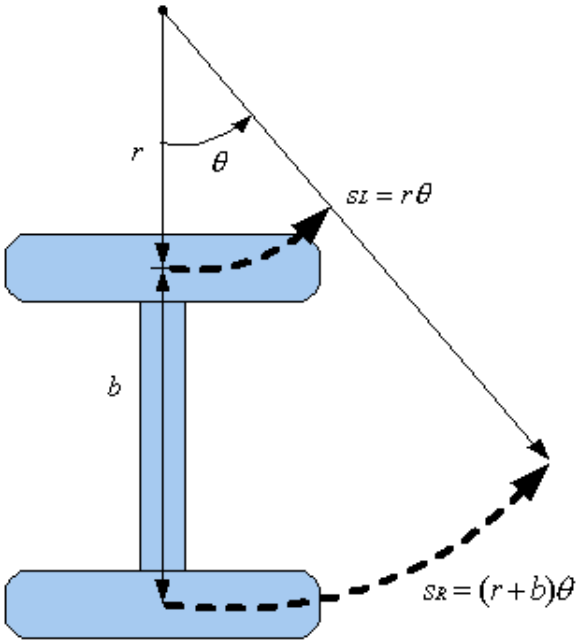
Matematický model

Pokud se robot pohybuje přímým směrem, je ujetá vzdálenost dána jeho rychlostí a pro obě kola stejná. Zajímavější situace nastává při průjezdu zatáčkou. Při zatáčení je každé kolo hnáno jinou rychlostí a ujede jinou vzdálenost, která je určena vztahem:

$$\begin{aligned} s_L &= r \cdot \theta \\ s_R &= (r + b) \cdot \theta \\ s_M &= (r + \frac{b}{2}) \cdot \theta \end{aligned} \quad ,$$

kde parametry jsou zřejmé z obrázku 2.3.

Pro výpočet změny polohy robota v kartézském systému souřadnic s počátkem v bodě [0,0] lze využít převodu na polární souřadnice. V následujícím vztahu je využit úhel nato-



Obrázek 2.3: **Dráha kol při průjezdu zatáčkou** - zdroj: http://rossum.sourceforge.net

čení θ, který udává natočení od osy x proti směru hodinových ručiček:

$$\frac{dx}{dt} = v(t) \cdot cos(\theta(t))$$
$$\frac{dy}{dt} = v(t) \cdot sin(\theta(t)) \quad ,$$

kde $v(t)$ je okamžitá rychlost robota ve směru úhlu θ. Změnu úhlu natočení robota můžeme vypočítat z tohoto vztahu:

$$\frac{d\theta}{dt} = (v_R - v_L)/b \quad ,$$

kde v_L a v_R jsou okamžité rychlosti kol.

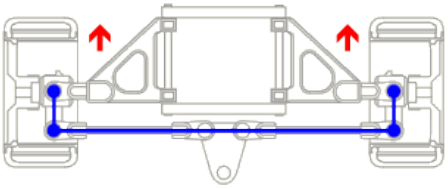
2.1.3 Vozidla s Ackermanovým řízením

Ackermanovo řízení využívají především vozidla se dvěma nápravami. K samotnému zatáčení je využívána přední náprava, jejíž kola se mohou natáčet do směru zatáčky.

Geometrie Ackermanova řízení zajišťuje správný úhel natočení řídících kol při průjezdu zatáčkou[5][9]. Princip tohoto řízení spočívá v rozdílném úhlu natočení vnitřního a vnějšího kola. Při průjezdu zatáčkou opisuje vnitřní strana vozidla kružnici o menším poloměru než strana vnější. Na pevné zadní nápravě je daný problém vyřešen pomocí diferenciálu. Důmysl Ackermanova řízení spočívá především v geometrii přední nápravy auta.

Řízení pomocí paralelní řídící tyče

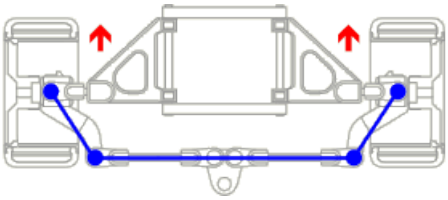
Řídící tyč je umístěna rovnoběžně s řídící nápravou. Při průjezdu zatáčkou se vnitřní i vnější kolo natočí o stejný úhel. Protože ale kola neopisují stejnou kružnici, dochází v tomto případě ke smýkání. To způsobuje další nežádoucí děje jako například nepřesné řízení nebo nadměrné opotřebení pneumatik.



Obrázek 2.4: **Řízení pomocí paralelní řídící tyče** - zdroj: http://rctek.com

Ackermanův princip řízení

Páka řízených kol je skloněná o určitý úhel k šasi. Tím lze dosáhnout, že vnitřní kolo se natočí o větší úhel než kolo vnější a díky tomu obě kola opisují kružnice o různých poloměrech. Průběh rozdílného natočení kol je exponenciální. To znamená skutečnost, že čím více vozidlo zatočí, tím je větší rozdíl v natočení kol.



Obrázek 2.5: **Ackermanův princip řízení** - zdroj: http://rctek.com

Modifikace Ackermanova řízení

Princip Ackermanova řízení dává návrháři volnost při nastavování parametrů řízení[12]. Těchto modifikací se využívá především u závodních vozů. Vhodným umístěním čepu, který spojuje řídící tyč s pákou kola, lze docílit sbíhavosti či rozbíhavosti kol. Rozlišují se 3 základní varianty nastavení:

- *Neutrální vytočení kol*
- *Rozbíhavost kol*
- *Sbíhavost kol*

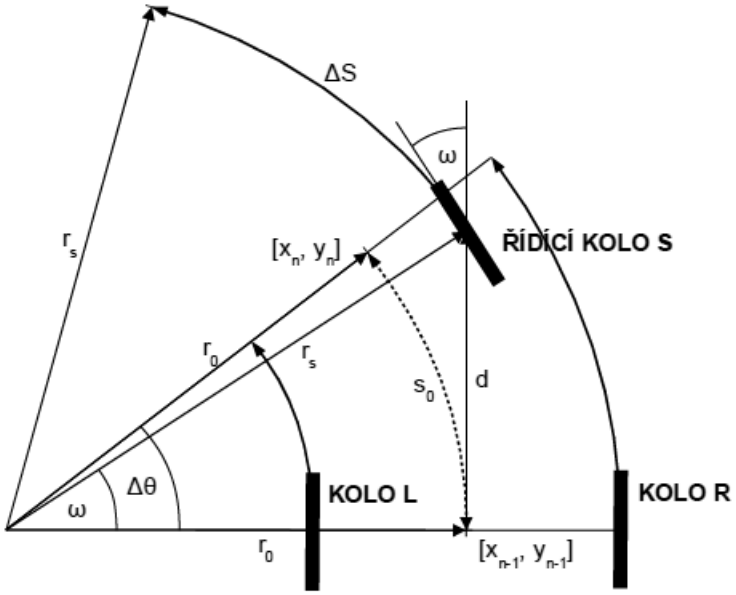
Matematický model

Změna pozice se v globálním systému souřadnic vypočte pomocí stejných vztahů jako pro model s diferenciálním řízením.

$$\begin{aligned} \frac{dx}{dt} &= v(t) \cdot cos(\theta(t)) \\ \frac{dy}{dt} &= v(t) \cdot sin(\theta(t)) \end{aligned} \quad ,$$

kde $v(t)$ je okamžitá rychlost robota ve směru úhlu θ.

Pro výpočet změny orientace robota však platí jiný vztah, který závisí na úhlu natočení řízení. Do vztahu je nutné zahrnout skutečnost, že každé kolo na přední nápravě se natočí o jiný úhel. Výsledný vztah lze odvodit ze zjednodušeného modelu tříkolky (tzv. tripod).



Obrázek 2.6: **Model tříkolky** - zdroj: www.robotika.cz

Pokud je přední kolo natočeno o nenulový úhel, pohybuje se tříkolka po kružnici. Z obrázku 2.6 lze odvodit následující geometrické vztahy.

$$\begin{aligned} r_s &= \frac{d}{sin(\omega)} \\ r_0 &= \frac{d}{tan(\omega)} \end{aligned} \quad ,$$

kde d je rozvor kol a ω je úhel natočení řídícího kola, r_0 je poloměr zatáčky.

Změnu orientace robota lze získat ze vztahu:

$$\Delta\theta = \frac{\Delta S}{r_s} = \frac{\Delta S \cdot cos(\omega)}{r_0} \quad ,$$

kde ΔS je naměřená ujetá vzdálenost řídícího kola.

Vzdálenost ujetou středem zadní nápravy můžeme získat ze vztahu:

$$s_0 = \Delta\theta \cdot r_0 = \Delta S \cdot cos(\omega)$$

Při použití derivace získáme vztah pro spojitou funkci změny úhlu orientace:

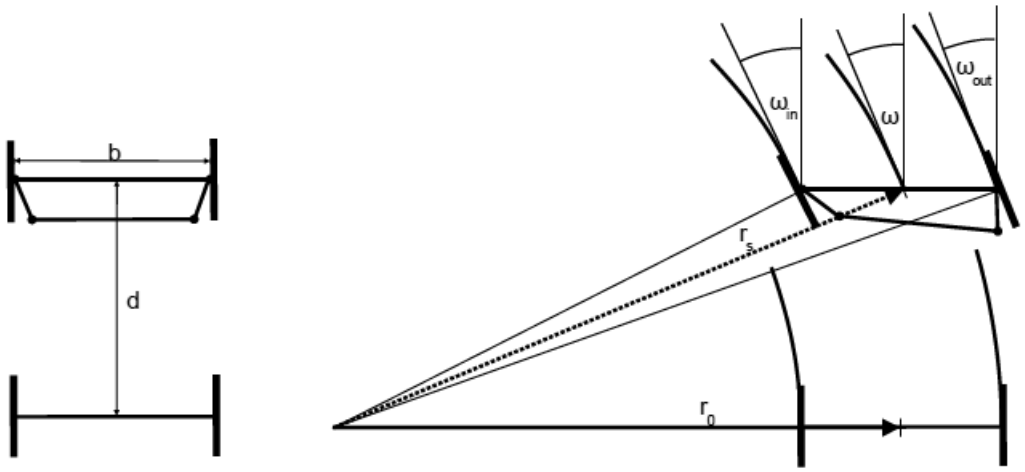
$$\frac{d\theta}{dt} = \frac{dS(t)}{dt} \cdot \frac{cos(\omega(t))}{r_0} = \frac{v(t) \cdot cos(\omega(t))}{r_0} \quad ,$$

kde $v(t)$ je okamžitá rychlost vozidla.

Z modelu tříkolky můžeme přejít na model podvozku klasického auta a přidáme ještě jeden matematický vztah. Je to tzv. podmínka pro Ackermanovo řízení pro natočení vnitřního a vnějšího kola řízené nápravy:

$$cotg(|\omega_{in}|) + \frac{b}{2d} = cotg(|\omega|) = cotg(|\omega_{out}|) - \frac{b}{2d} \quad ,$$

kde parametry jsou zřejmé z obrázku 2.7.



Obrázek 2.7: **Ackermanova podmínka**

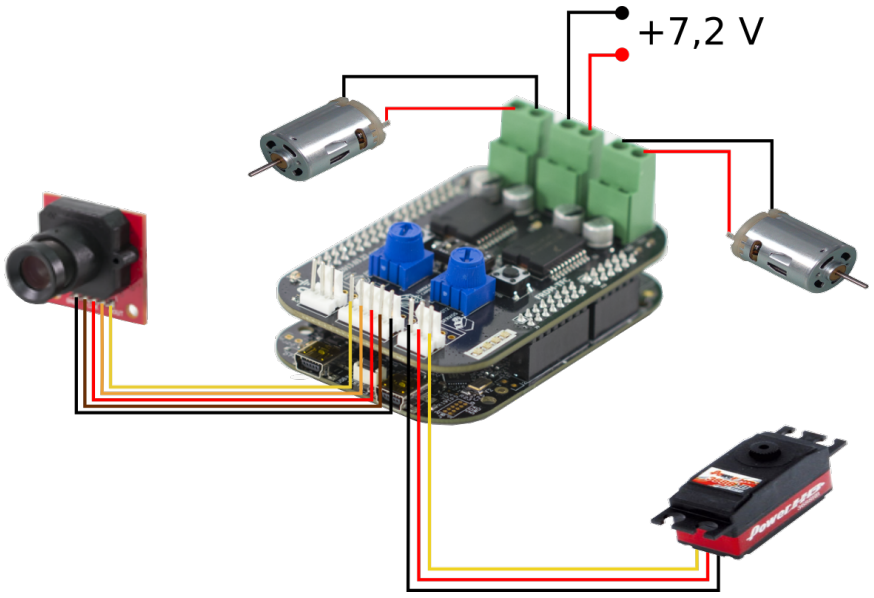
V této kapitole byla rozebrána konstrukce šasi vozidla. V následující kapitole bude popsána elektronika řídícího systému.

Kapitola 3

Realizační platforma - elektronika

Klíčovou úlohu pro funkčnost robota mají jeho senzory, akční členy a řídící jednotka. Jednotlivé komponenty spolu musejí při řízení vhodně spolupracovat. Je rovněž vhodné, aby jednotlivé části systému byly podobné kvality a výkonu. Nemá smysl realizovat závodní model se silnými motory, ale nepřesnými senzory. Pro soutěž Freescalecup je stanovena jednotná realizační platforma. Tato kapitola je věnována popisu jednotlivých komponent.

Na následujícím obrázku je znázorněno propojení elektroniky.



Obrázek 3.1: **Schéma platformy**

3.1 Řízení DC motorů

Motor

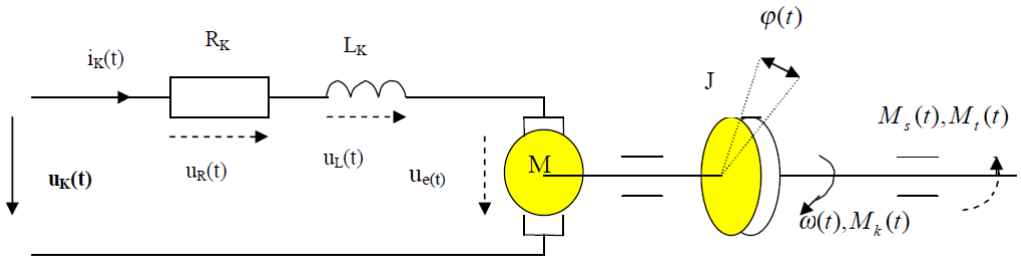
Stejnosměrný elektromotor převádí elektrickou energii na mechanickou ve formě rotací. Pohyb je výsledkem elektromagnetických sil. Motory mají v sobě cívky, které vytvářejí mag-

netické pole. Cívky jsou namotané na rotoru a jejich počet může být různý dle kontrukce motoru. Většina modelářských stejnosměrných motorů se vyrábí se třemi cívkami. Správná změna jejich magnetických pólů je zajištěna komutátorem. Rotor se točí mezi permanentními magnety, které jsou připevněny ke statoru.

Stejnosměrné motory již byly ve výkonu a efektivitě překonány střídavými, neboli bezuhlíkovými (brushless) motory, ale stále se hojně používají.

Matematický model

Následující matematický model DC motoru je odvozen za zjednodušujících, nicméně většinou akceptovatelných, předpokladů. [6]



Obrázek 3.2: **Model DC motoru**

Popis paramertrů:

- R_K, L_K - odpor a indukčnost vinutí kotvy (konstantní parametry)
- $u_K(t)$ - napětí přiváděné do kotvy motory
- $i_K(t)$ - proud kotvy
- $u_e(t) = k_e\omega(t)$ - napětí vzniklé v důsledku rotace kotvy
- $\omega(t), \varphi(t)$ - úhlová rychlost otáčení hřídele motoru, úhel natočení hřídele motoru
- J - moment setrvačnosti rotoru
- $M_k(t) = k_M i_K(t)$ - kroutící moment motoru (předpokládáme lineární závislost na proudu kotvy; k_M je konst.)
- $M_s(t) = J\frac{d\omega(t)}{dt}$ - setrvačný moment
- $M_t(t) = b\omega(t)$ - třecí moment (předpokládáme lineární závislost na proudu kotvy; k_M je konst.)

Diferenciální rovnice

Matematický model systému lze rozdělit na dvě části[10]:

- Elektrická část
- Mechanická část

Rovnice pro elektrickou část vyjadřuje závislost proudu na vstupním napětí kotvy $u_K(t)$, odporu a indukčnosti cívky R_K, L_K a napětí $u_e(t)$, které vzniklo v důsledku rotace kotvy v magnetickém poli.

$$L_K \frac{di_K(t)}{dt} + R_K i_K(t) = u_K(t) - k_e \omega(t)$$

Rovnice popisující mechanickou část vyjadřuje závislost úhlové rychlosti rotace kotvy $\omega(t)$ na proudu $i_K(t)$ protékajícím kotvou, koeficientem viskózního tření b, momentem setrvačnosti kotvy J a kroutícím momentu motoru $k_M i_K(t)$.

$$J \frac{d\omega(t)}{dt} + b\omega(t) = k_M i_K(t)$$

Úhlovou rychlost $\omega(t)$ lze dále převést na úhel natočení hřídele $\varphi(t)$.

$$\frac{d\varphi(t)}{dt} = \omega(t)$$

Stavový model

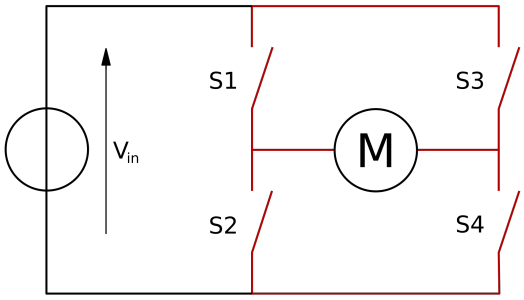
Zavedením stavových proměnných $x_1(t) = i_K(t)$, $x_2(t) = \omega(t)$ lze získat stavový model motoru, jehož výstupem je úhlová rychlost $y(t) = \omega(t)$

$$S(A, b, c^T) : \begin{bmatrix} \dot{x_1}(t) \\ \dot{x_2}(t) \end{bmatrix} = \begin{bmatrix} -\frac{R_K}{L_K} & -\frac{k_e}{L_K} \\ \frac{k_M}{J} & \frac{-b}{J} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_K} \\ 0 \end{bmatrix} u_K(t)$$
$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

H-můstek

Na obrázku 3.3 je schéma můstku. Touto komponentou lze kontrolovat směr otáčení motoru.

Rychlost otáčení je určována pulsně šířkovou modulací tzv. PWM. Tato metoda spočívá v sepínání tranzistorů H-můstku tak, aby byl omezen čas průchodu proudu. Toho je docíleno



Obrázek 3.3: **H-můstek**

pomocí digitálního signálu ovlivněného střídou (ang. duty cycle). Protože se pojem duty cycle používá častěji, bude dále používán pouze tento pojem.

Duty cycle znamená poměr logické úrovně 1 a 0 v konstantní frekvenci digitálního signálu. Tím je docíleno, že je proud do motoru „dávkován“. Protéká jen tehdy, kdy jsou tranzistory sepnuté, respektive na digitálním signálu je logická 1.

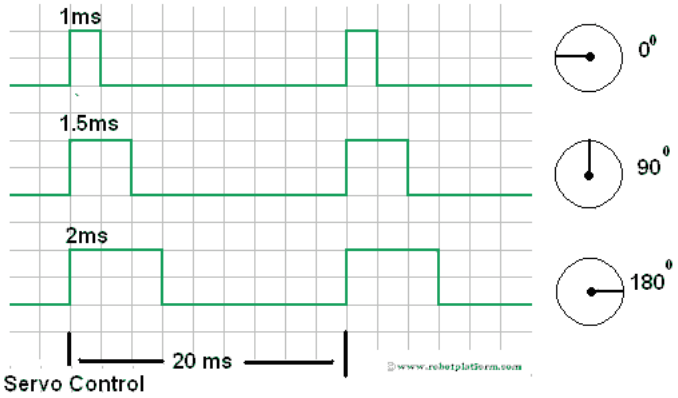
Na závěr této sekce je přiložena tabulka 3.1 možných řídících signálů a jejich vlivu, který mají na motor [14].

Tabulka 3.1: Řídící signály H-můstku

S1	S2	S2	S4	Výsledek
1	0	0	1	Otáčení doprava/dopředu
0	1	1	0	Otáčení doleva/dozadu
0	0	0	0	Volné otáčení motoru
0	1	0	1	Brzdění motoru
1	0	1	0	Brzdění motoru
1	1	0	0	Zkratování zdroje
0	0	1	1	Zkratování zdroje
1	1	1	1	Zkratování zdroje

3.2 Ovládání serv

Servo je akčním členem pro zatáčení robota. Je to speciální zařízení složené z DC motoru, převodů, potenciometru a řídícího obvodu. Speciální funkcí serva je, že je schopné zůstat v natočené pozici libovolnou dobu. Serva jsou velice často používána v robotice. Na rozdíl od DC motoru má servo ještě třetí vodič, pomocí kterého lze ovládat jeho natočení. Tímto vodičem je do serva předáván PWM signál, který svým duty cyclem udává, o jaký úhel se má páka serva natočit. Perioda tohoto signálu je 20ms a puls, udávající natočení serva, se pohybuje mezi 1 ms a 2 ms. Obrázek 3.4 ilustruje řídící signály servomotoru.



Obrázek 3.4: **Servo - duty cycle** - zdroj: http://www.robotplatform.com

3.3 Kamera

Hlavním senzorem robota je kamera. Snímá obraz o rozlišení 1x128 pixelů. Kamera je nazývána Linescan-kamera a snímá jednodimenzionální obraz. Tento senzor je ideální k použití na sledování čáry. Díky malému rozlišení je zpracování snímku výkonově a paměťově nenáročné.

Senzor se skládá ze 128 fotodiod [1]. Světlo dopadající na fotodiodu generuje proud, který je následně integrován a převeden na hodnotu mezi 0 a 4096. Pomocí softwarových driverů lze nastavit čas integrování a tedy i velikost hodnot vracených kamerou. Při vysoké intenzitě světla a dlouhém integračním času se fotodiody dostávají do saturace. Naopak při nedostatečném osvětlení a krátkém času jsou naměřeny slabé rozdíly mezi odrazem světla od bílého a černého povrchu.

3.4 Vývojová platforma

Vývojovou platformou je deska Freescale FRDM-KL25Z. Výhodou této platformy jsou nízké pořizovací náklady a široké možnosti uplatnění. Jádrem platformy je processor ARM Cortex M0+. K jeho programování je používán jazyk C a k vývoji software lze využít řadu IDE. Dva z nejznámějších zástupců jsou MBED online coder [2] vyvíjený společností ARM a IDE Codewarrior [8] od společnosti Freescale, které je založeno na prostředí Eclipse.

V této kapitole byla popsána elektronika autonomního auta. Nyní je zřejmé jaké senzory a akční členy byly při projektu k dispozici.

Aby bylo možné navrhnout řídící algoritmus, bylo nejprve nezbytné vytvořit systém pro zpracování obrazu z kamery.

Kapitola 4

Zpracování obrazu

Klíčovou úlohou pro správnou funkčnost robota je rozpoznávání obrazu z kamery. Při nespolehlivém rozpoznávání nemůže sebelepší regulátor fungovat správně. Aby bylo možné vytvořit spolehlivý algoritmus na rozpoznávání, bylo potřeba provést řadu měření. Při tom bylo žádoucí vyzkoušet různé intenzity osvětlení, integrační časy a pozorovací úhly kamery.

Nejlepších výsledků při rozpoznávání je docíleno manuálním nastavením těchto parametrů v závislosti na podmínkách prostředí, ve kterém je robot provozován. Proces je však zdlouhavý, což vedlo ke snaze vytvořit alespoň částečně automatický algoritmus.

Vizualizační program

Pro usnadnění úlohy nastavování parametrů byl vytvořen vizualizační program. Implementace byla provedena v jazyce Python [7] z důvodu poměrně snadné a rychlé realizace. Robot posílá přes sériový port naměřená a spočtená data, která jsou následně zobrazena pomocí vytvořené aplikace.

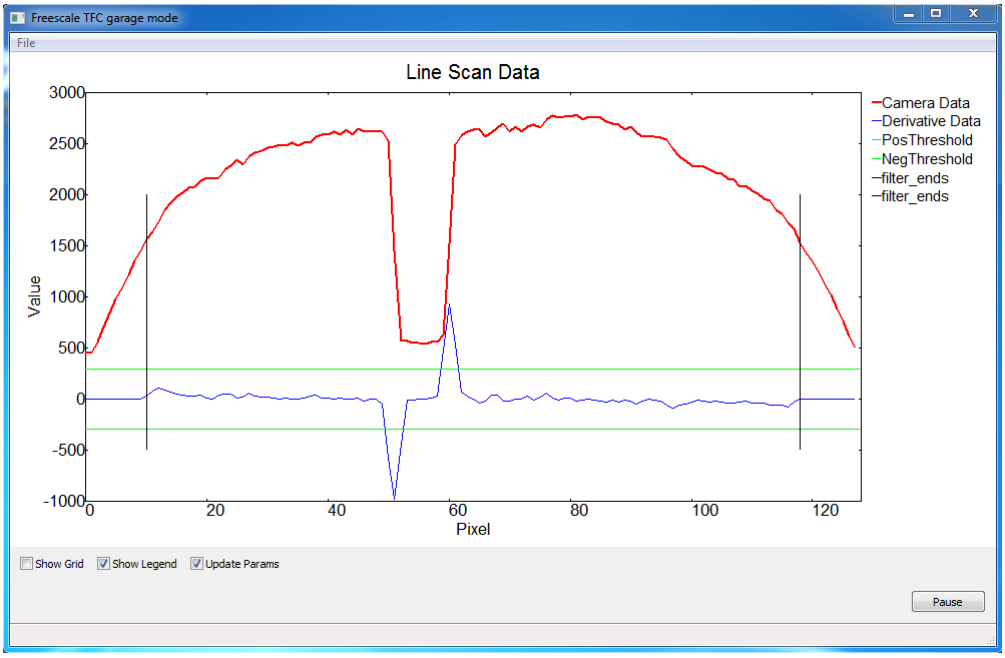
Na obrázku 4.1 je jeden snímek z kamery. Na vodorovné ose jsou zobrazeny pixely a na svislé naměřené hodnoty světla a jejich diference. Propad v naměřených hodnotách světla je právě obraz černé čáry, která odráží méně světla.

Jedna z možností, jak rozpoznat čáru, je zavedení určitého prahu. Naměřené hodnoty nad tímto prahem jsou považovány za bílou a ty pod ním za černou barvu. Tento způsob má však značné nevýhody. Intenzita světla může být na různých částech tratě jiná a tudíž by jeden univerzální práh nestačil. Navíc, jak je zřejmé z obrázku 4.1, kamera snímá nižší hodnoty na okrajích snímku.

Mnohem lepších výsledků bylo dosaženo při použití metody založené na diferenci hodnot intenzity světla. Místo černé čáry tato detekce hledá ostré přechody mezi černou a bílou barvou. Přechod je reprezentován vysokou, nebo nízkou hodnotou diference. Na obrázku 4.1 je zobrazena i diference snímku (modrá čára). Zavedením prahu - thresholdu (zelená čára) lze z obrazu vyfiltrovat ostré přechody v barvě povrchu.

V tabulce 4.1 lze nalézt význam hodnot diferencí.

Tato metoda je již univerzální a není tolik citlivá na intenzitu osvětlení. Stále však hodně záleží na správném zaostření kamery a nastavení prahů detekce.



Obrázek 4.1: **Vizualizační program**

Tabulka 4.1: Význam hodnot diference

Hodnota diference	Význam
$diference > threshold$	Přechod barvy povrchu z černé na bílou
$diference < -threshold$	Přechod barvy povrchu z bílé na černou
$\|diference\| < \|threshold\|$	Jednotný povrch

Složitější situace nastává při rozpoznávání dráhy, kdy je nutné detekovat čáry na okrajích snímku.

Na obrázku 4.2 je zobrazen snímek, kde jsou na krajích zachyceny černé čáry a mezi nimi bílá trať. Po porovnání s obrázkem 4.1 je vidět, že bylo nutné nastavit nižší práh pro detekci přechodů. Oba snímky byly měřeny za dobrého rovnoměrného osvětlení. V praxi je však takového osvětlení dosaženo jen vzácně, a proto detekce obou postranních čar představuje složitější problém.

Obrázek 4.2: **Detekce 2 čar**

Diference naměřených hodnot

Diference je velmi podobná derivaci. Na rozdíl od derivace, která je definována na spojitých, hladkých funkcích, však lze diferenci využít i na diskrétních posloupnostech čísel. Snímek kamery se skládá právě z diskrétních hodnot, a proto je pro účely rozpoznávání použita tato metoda.

Pro signál:

$$x[k]..., k = \langle 0, 127 \rangle$$

lze získat hodnotu diference takto:

$$\dot{x}[k] = \frac{x[k+1] - x[k-1]}{2}$$

a v krajních bodech obrazu je to:

$$\dot{x}[0] = \frac{x[1] - x[0]}{2}$$
$$\dot{x}[127] = \frac{x[127] - x[126]}{2}$$

Rozpoznávací algoritmus detekuje počet čar, respektive hran a jejich polohu. Na základě těchto informací a vzdálenosti hran lze určit chybu pozice robota.

Po vytvoření programové části získávající informace z obrazu bylo možné začít s vývojem softwaru pro řízení.

Kapitola 5

Řízení

V této kapitole se dostáváme k nejdůležitější části práce, návrhu řídícího algoritmu. Na následujících řádcích je popsán vývoj systému. První část kapitoly se věnuje vývoji algoritmu vytvořenému pro sledování čáry a ve druhé se dostaneme ke sledování dráhy.

5.1 Sledování čáry

5.1.1 Výpočet chyby

Vstupem regulátoru je odchylka od požadované hodnoty referenční veličiny. Referenční veličinou v této úloze je poloha čáry. Její požadovaná pozice je uprostřed snímku z kamery, tedy kolem 64. pixelu.

Pro určení chyby pozice robota je nutno vycházet z hodnot získaných při detekci hran. K dispozici jsou pozice přechodových hran, jejich počet a vzdálenosti. V tabulce 5.1 jsou rozepsány individuální případy pro určení pozice čáry.

Tabulka 5.1: Určení pozice čáry

`#PEdg`	`#NEdg`	SupplementaryCondition	`LineStatus`	`CurrentLinePosition`
1	1	`p[0] - n[0] < LineWidth`	`LineFound`	$\frac{\texttt{n[0] + p[0]}}{2}$
1	0	`p[0] < LineWidth`	`LineOnTheLeft`	$\frac{\texttt{p[0] - LineWidth}}{2}$
0	1	`n[0] > 128-LineWidth`	`LineOnTheRight`	$\frac{\texttt{n[0] + LineWidth}}{2}$

Popis parametrů

- `#PEdg, #NEdg` - počet detekovaných pozitivních, resp. negativních hran v obrazu.
- SupplementaryCondition - další podmiňující podmínka pro detekci stavu

- `n[0,...]`, `p[0,...]` - pole nalezených negativních, resp. pozitivních hran
- `LineWidth` - šířka čáry v obrazu (počet pixelů)
- `LineStatus` - informace o tom, zda byla detekována čára
- `LineFound` - detekována jedna pozitivní a jedna negativní hrana, které jsou od sebe vzdálené o hodnotu menší nebo rovnu LineWidth - algoritmus rozpoznávání detekoval čáru.
- `LineOnTheLeft` - byla detekována pouze jedna pozitivní hrana na okraji snímku. Čára je na levém okraji snímku.
- `LineOnTheRight` - byla detekována pouze jedna negativní hrana na okraji snímku. Čára je na pravém okraji snímku.
- `CurrentLinePosition` - pozice čáry. Podle detekovaného stavu se určí pozice čáry.

Pokud byly detekovány jiné počty hran, je snímek označen za neplatný a není použit k řízení.

Po získání pozice čáry lze určit okamžitou chybu z následujícího vztahu:

$$\texttt{CurrentLinePosError = CurrentLinePosition - TargetPosition} \quad ,$$

kde `TargetPosition` = 63.5.

Tato odchylka plně postačuje k proporcionálnímu řízení zatáčení.

V následující části této kapitoly se podíváme, jaké jsou možnosti regulace[15].

5.1.2 Typy regulátorů

P - regulátor

Jedná se o základní regulátor. Velikost chyby je pouze násobena konstantou a přenášena na výstup systému prostřednictvím akčního členu. Proporcionální regulátor je prostý zesilovač. Mezi akční veličinou a regulační odchylkou je přímá úměrnost.

Matematický model:

$$u(t) = r_0 e(t) \quad ,$$

kde e(t) je regulační odchylka a u(t) akční veličina.

Přenos:

$$F_R(s) = \frac{U(s)}{E(s)} = r_0 = K_R \quad ,$$

kde K_R je konstanta zesílení.

Při návrhu tohoto regulátoru je žádoucí normalizovat zesílení tak, aby při maximální chybě `CurrentLinePosError` $= \pm 63.5$ byla na servo posílána hodnota pro maximální zatáčení. Koeficient K_R lze spočítat z následujícího vztahu:

$$K_R = \frac{\texttt{MAX\_STEER\_RIGHT - MAX\_STEER\_LEFT}}{\texttt{NUM\_LINE\_SCAN}} \quad ,$$

kde `MAX_STEER_RIGHT` a `MAX_STEER_LEFT` je hodnota pro maximální natočení serva doprava, resp. doleva a `NUM_LINE_SCAN` je počet pixelů kamery.

Regulátor lze rozšířit dalšími členy, aby byla dosažena rychlejší a kvalitnější regulace.

I - regulátor

Akční veličina tohoto regulátoru je přímo úměrná integrálu regulační odchylky. Jedná se tedy o nekonečně jemné vysčítání regulačních odchylek v čase (0,t). Čím déle je regulační odchylka nenapravena, tím větší akční veličinu poskytne I - regulátor k její korekci.

Matematický model:

$$u(t) = r_i \int_0^t e(\tau)d\tau + u(0)$$

Přenos:

$$F_R(s) = \frac{U(s)}{E(s)} = \frac{r_i}{s} = \frac{1}{T_i s} \quad ,$$

kde r_i je zesílení integračního členu a T_i je časová konstanta.

Regulační děj I regulátoru je oproti proporcionálnímu pomalejší a může způsobit horší stabilitu soustavy. Vlivem integrace může docházet k překmitům.

Realizace I - členu

Matematický model s integrálem je opět zaveden pro spojité veličiny. Pro použití v digitálních systémech je potřeba přejít k diskrétním odchylkám. Integrál se proto převádí na sumu:

$$\texttt{SumLinePosError} = \sum_0^t \texttt{CurrentLinePosError[t]}$$

D - regulátor

D - regulátor převádí derivaci regulační odchylky na akční veličinu. Protože „čistá“ derivace není technicky možná, zavádí se pojem ideální D regulátor.

Matematický model:

$$u(t) = r_d \frac{de(t)}{dt}$$

Přenos:

$$F_R(s) = \frac{U(s)}{E(s)} = r_d s \quad ,$$

kde r_d je zesílení derivační složky.

Derivační regulátor se používá pro zrychlení regulačního děje. Nevýhodou je, že zesiluje šum. Tato vlastnost je příčinou, že ho v mnoha případech nelze použít. Samostatně se nikdy nevyskytuje. Je součástí PD nebo PID regulátorů.

Realizace D - členu

I v tomto případě je potřeba přejít ze spojitých funkcí na diskrétní. Derivace je nahrazena rozdílem:

```
DerivError = CurrentLinePosError[t] - CurrentLinePosError[t-1]
```

Všechny tři členy je možné spojit do jednoho regulátoru. Tím se vytvoří PID regulátor.

PID regulátor

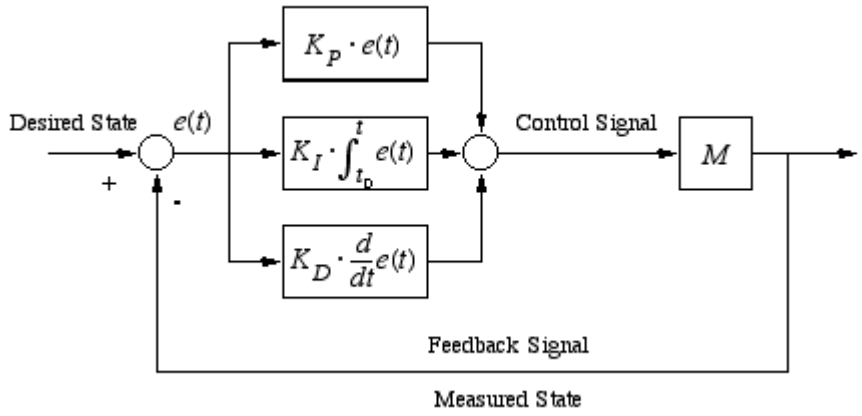
Matematický model:

$$u(t) = r_0 e(t) + r_d \frac{de(t)}{dt} + r_i \int_0^t e(\tau) d\tau + x(0)$$

Přenos:

$$F_R(s) = \frac{U(s)}{E(s)} = K_R(1 + T_D s + \frac{1}{T_I s}) = K_R \frac{(T_1 s + 1)(T_2 s + 1)}{s}$$

Vhodnou kombinací jednotlivých členů je možné značně vylepšit chování regulované soustavy. Velice však záleží na vhodném nastavení zesílení jednotlivých složek. V případě regulování systému, u kterého není k dispozici matematický model, nemusí být volba parametrů snadná.



Obrázek 5.1: **PID regulátor** - zdroj: http://cs.brown.edu

5.1.3 Implementace regulátorů

Pro řízení autonomního vozidla byly naimplementovány dva regulátory. První z nich je určen pro řízení zatáčení 5.1, druhý pro regulaci rychlosti motorů 5.3.

Listing 5.1: PID zatáčení - pseudokód

```
LastLinePosError = 0
SumLinePosError = 0
start:
  CurrentLinePosError = CurrentLinePosition - TargetPosition
  SumLinePosError += CurrentLinePosError
  DerivLinePosError = CurrentLinePosError - LastLinePosError
  Pout = Kp * CurrentLinePosError
  Iout = Ki * SumLinePosError
  Dout = Kd * DerivLinePosError
  CurrentSteerSetting = Pout + Iout + Dout + MID_POSITION
  LastLinePosError = CurrentLinePosError
  wait(dt)
  goto start
```

Regulátor pro zatáčení generuje řízení `CurrentSteerSetting`. Tento signál je otestován, zda je v intervalu ⟨`MAX_STEER_LEFT`, `MAX_STEER_RIGHT`⟩ a pokud ne, je uveden do saturace 5.2.

Listing 5.2: Saturace `CurrentSteerSetting` - pseudokód

```
if (CurrentSteerSetting > MAX_STEER_RIGHT)
{
    CurrentSteerSetting = MAX_STEER_RIGHT
}
else if (CurrentSteerSetting < MAX_STEER_LEFT)
{
    CurrentSteerSetting = MAX_STEER_LEFT
}
```

Konstanta `MID_POSITION` je zde pro korekci středové pozice serva. Po každé montáži serva je nutné konstanty `MAX_STEER_LEFT, MAX_STEER_RIGHT` a `MID_POSITION` správně nastavit. Parametry `Kp, Ki, Kd` jsou zesílení jednotlivých složek, které je nezbytné vhodně zvolit.

Funkcí regulátoru pro řízení motorů je brzdit vozidlo do zatáček a regulovat rychlost při jejich průjezdu. Pro tento účel byl naimplementován PD regulátor.

Listing 5.3: PD řízení motorů - pseudokód

```
LastLinePosError = 0
start:
    CurrentLinePosError = CurrentLinePosition - TargetPosition
    DerivLinePosError = CurrentLinePosError - LastLinePosError

    if (CurrentLinePosError < 0)
        AbsLineError = -1 * CurrentLinePosError
    else
        AbsLineError = CurrentLinePosError

    if (DerivLinePosError < 0)
        AbsDerLineError = -1 * DerivLinePosError
    else
        AbsDerLineError = DerivLinePosError

    Pout = ThKp * AbsLineError
    Dout = ThKd * AbsDerLineError

    CurrentMotorChange = Pout + Dout
    CurrentSpeedSetting = maxSpeed - CurrentMotorChange

    LastLinePosError = CurrentLinePosError
    wait(dt)
    goto start
```

Řízení motorů je odlišné hlavně v použití absolutních chybových hodnot `AbsLineError` a `AbsDerLineError`. Při regulaci rychlosti je důležitá absolutní hodnota chyby. Ta je převedena na signál brzdění `CurrentMotorChange`, který je následně odečten od maximální rychlosti `maxSpeed`. `CurrentSpeedSetting` je řídící signál. Parametry `ThKp` a `ThKd` je opět nezbytné vhodně nastavit.

Výše popsanou regulaci lze již použít v úloze sledování jedné čáry. Druhá úloha je však značně odlišná, protože se při ní musí počítat s přechody mezi několika stavy systému.

5.2 Sledování dráhy

Sledování dráhy je složitějším problémem a liší se zejména ve výpočtu chyby. Zde je nutné nejprve určit, na kterou stranu je potřeba zatáčet, a poté jak prudce.

Okraje dráhy jsou od sebe vzdáleny 50 cm a kamera má zorný úhel 45 °. Při návrhu systému je možné experimentovat s natočením kamery, a tedy tím, jak široký úsek trati bude senzor snímat. Podstatný rozdíl je ve stavu, kdy se robot nalézá uprostřed trati. Kameru je možné naklonit tak, že snímá buď obě, nebo ani jednu čáru.

V této práci je využit druhý případ, kdy při nulové chybě není v zorném poli kamery ani jedna čára.

Pro výpočet chyby byl naimplementován automat s konečným počtem stavů (Finite State Machine).

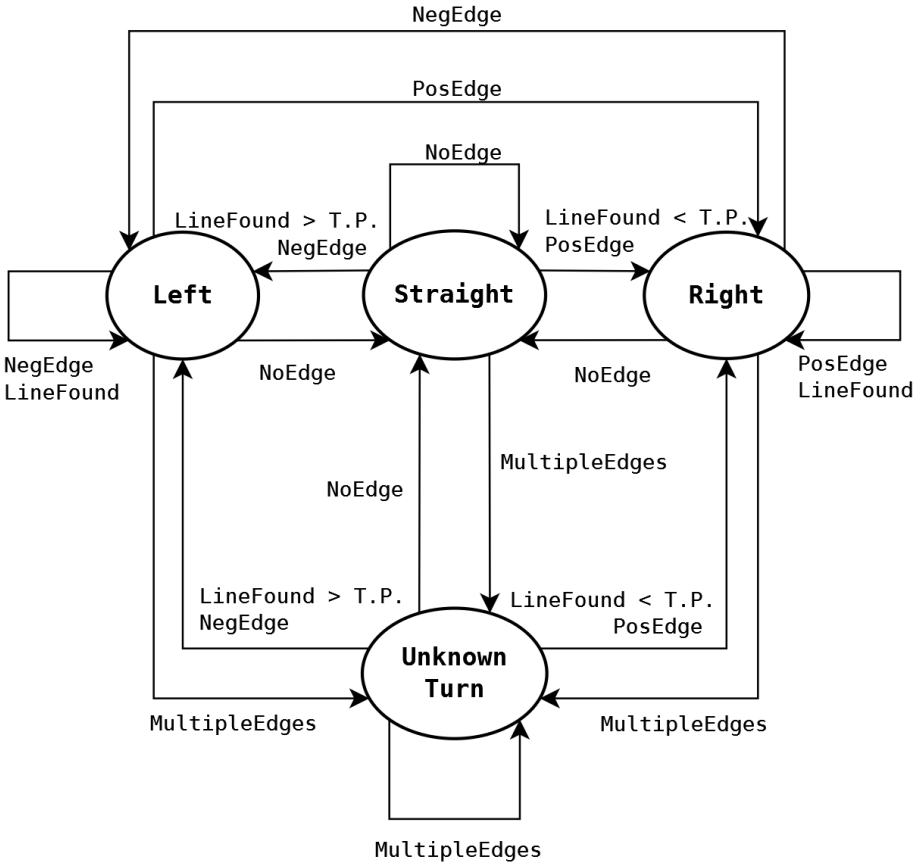
5.2.1 Finite State Machine

Finite State Machine je model systému, který se skládá z konečného počtu možných stavů [4]. Systém se vždy nalézá právě v jednom stavu. Ten je nazýván aktuálním stavem. Při vzniku spouštěcí události nebo vyhodnocení podmínky přejde systém do některého ze svých dalších stavů. Přechody mezi stavy jsou striktně definovány. Každý konečný automat je definován seznamem stavů a spouštěcími událostmi, které definují přechody.

Stavy konečného automatu v této práci byly charakterizovány proměnnou `CurrentTurn`, která může nabývat následujících hodnot:

- `Left`
- `Right`
- `Straight`
- `UnknownTurn`

Na obrázku 5.2 je zobrazen stavový diagram FSM a v tabulce 5.2 jsou shrnuty kombinace stavů, spouštěcí podmínky - detekce hran a následné přechody mezi stavy.



Obrázek 5.2: **Blokové schéma FSM**

5.2.2 Výpočet chyby

Určení pozice čáry probíhá stejným způsobem jako při úkolu sledování jedné čáry 5.1. Výpočet chyby v této úloze se liší tím, že pro každý ze stavů je definován jiný vzorec pro výpočet chyby řízení. V tabulce 5.3 jsou shrnuty všechny stavy.

Zavedení FSM vedlo k usnadnění práce při návrhu regulátoru. Protože výpočet chyby zachovává stejná znaménka pro dané směry zatáčení (kladná chyba znamená zatočení doprava a záporná naopak), bylo možné použít regulátor z předchozí úlohy. Z důvodu zvětšení rozsahu chyby však bylo vhodné upravit koeficienty zesílení složek regulátoru.

$$\texttt{CurrentLinePosError} \in \langle -128, 128 \rangle$$

Na obrázku 5.3 je zobrazen blokový diagram systému.

Problém sledování dráhy by bylo možné pojmout komplexněji. A sice s využitím informace, že požadovaná pozice na trati není vždy přesně uprostřed. Zatáčky lze rychleji projíždět po jiné trajektorii, než je střed dráhy. Toto by vedlo k nutnosti výpočtu optimální trajektorie a značně komplexnějšímu problému. Je to jedno z možných budoucích vylepšení algoritmu.

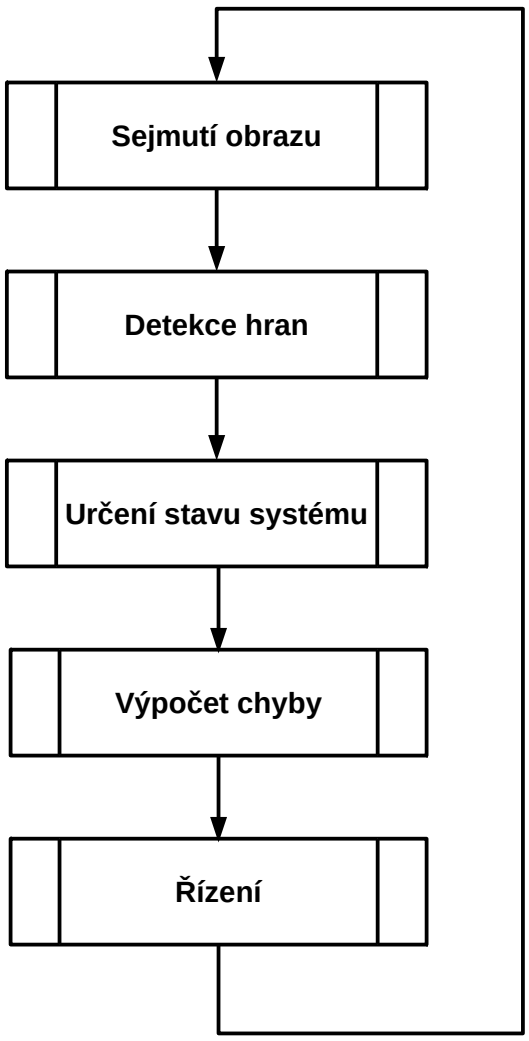
Tabulka 5.2: Přechodová tabulka FSM

Aktuální stav	Detekce	Příští stav	Akce
`Straight`	`NoEdge`	`Straight`	Jízda rovně
`Straight`	`PosEdge`	`Right`	Zatočení doprava
`Straight`	`NegEdge`	`Left`	Zatočení doleva
`Straight`	`LineFound < TargetPos`	`Right`	Zatočení doprava
`Straight`	`LineFound > TargetPos`	`Left`	Zatočení doleva
`Straight`	`multipleEdges`	`UknownTurn`	Poslední vykonávaná akce
`Right`	`NoEdge`	`Straight`	Jízda rovně
`Right`	`PosEdge`	`Right`	Zatočení doprava
`Right`	`LineFound`	`Right`	Zatočení doprava
`Right`	`NegEgde`	`Left`	Zatočení doleva
`Right`	`multipleEdges`	`UknownTurn`	Poslední vykonávaná akce
`Left`	`NoEdge`	`Straight`	Jízda rovně
`Left`	`NegEdge`	`Left`	Zatočení doleva
`Left`	`LineFound`	`Left`	Zatočení doleva
`Left`	`PosEdge`	`Right`	Zatočení doprava
`Left`	`multipleEdges`	`UknownTurn`	Poslední vykonávaná akce
`UnknownTurn`	`multipleEdges`	`UknownTurn`	Poslední vykonávaná akce
`UnknownTurn`	`NoEdge`	`Straight`	Jízda rovně
`UnknownTurn`	`PosEdge`	`Right`	Zatočení doprava
`UnknownTurn`	`NegEdge`	`Left`	Zatočení doleva
`UnknownTurn`	`LineFound < TargetPos`	`Right`	Zatočení doprava
`UnknownTurn`	`LineFound > TargetPos`	`Left`	Zatočení doleva

V následující kapitole bude popsáno nastavování a testování vytvořeného systému.

Tabulka 5.3: FSM - výpočet chyby

Stav	Vzorec pro výpočet chyby
`Straight`	`Error = 0`
`Right`	`Error = CurrentLinePosition`
`Left`	`Error = CurrentLinePosition - NUM_LINE_SCAN`



Obrázek 5.3: **Blokový diagram**

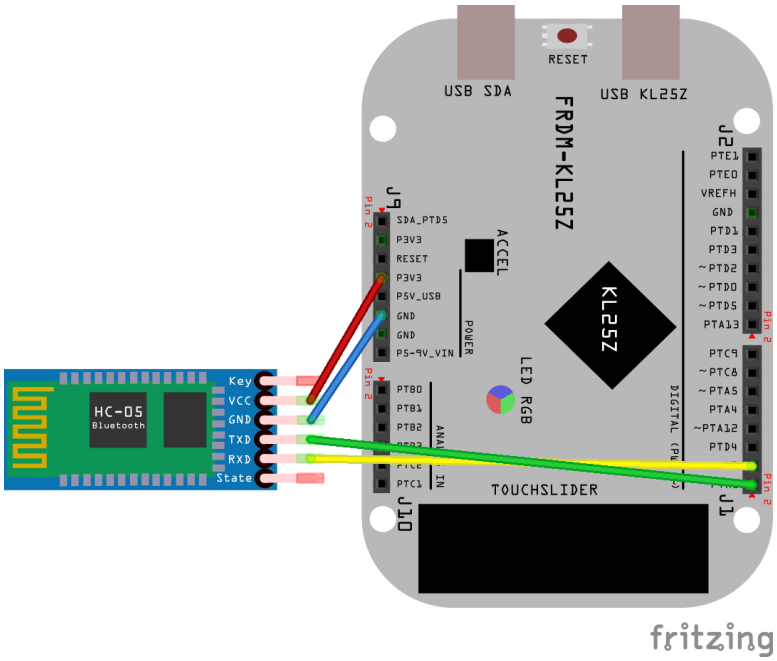
Kapitola 6

Nastavení a testování systému

Po naprogramování potřebných systémů bylo nezbytné nastavit parametry regulátoru. Protože nebyl k dispozici přesný matematický model, bylo nutné postupně vyzkoušet různé hodnoty parametrů a iterativně se přiblížit k vhodným hodnotám.

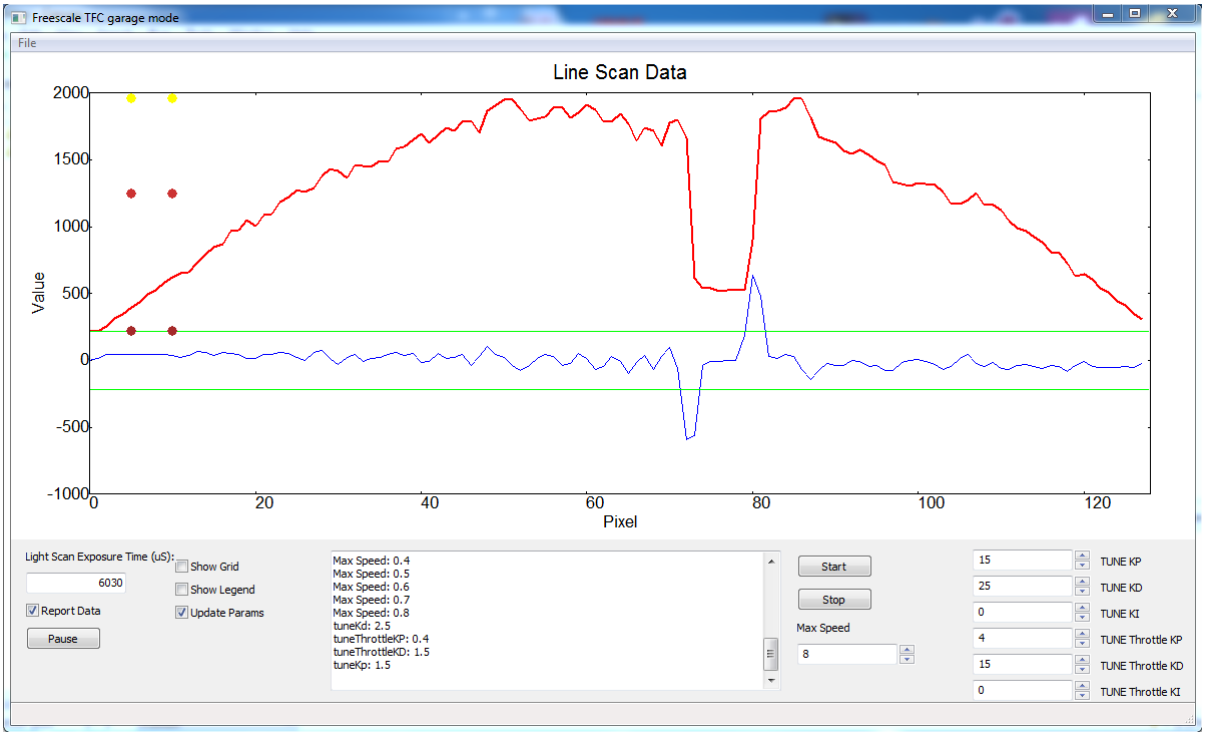
6.1 Bezdrátová komunikace a program pro nastavování parametrů

Aby bylo možné parametry ladit plynule za jízdy, byla vytvořena bezdrátová komunikace mezi robotem a počítačem. Na řídící jednotku byl připojen bluetooth modul, pomocí kterého byl vytvořen bezdrátový sériový port.



Obrázek 6.1: **Zapojení bluetooth**

Pomocí sériového portu je možné bezdrátově posílat příkazy pro změny parametrů přímo do mikroprocesoru robota. K zadávání příkazů lze využít například příkazový řádek emulátoru PuTTY [13]. Pro větší komfort nastavování byl rozšířen vizualizační program. Na obrázku 6.2 je snímek z nastavování parametrů regulátoru.



Obrázek 6.2: **Program pro nastavování parametrů**

Při jízdě je také možné sledovat data naměřená kamerou. Posílání dat však zatěžuje procesor, a proto byl tento mód používán pouze pro účely hrubého ladění. Vizualizaci je možné vypnout a program používat jen pro změnu parametrů regulátoru.

V pravém dolním rohu obrazovky jsou číselníky pro nastavování hodnot a uprostřed ovládacího panelu je terminál zobrazující informace o aktuálním nastavení.

6.2 Nastavování parametrů

Program pro nastavování parametrů regulátoru značně usnadnil celý proces ladění.

Nejdříve byly nastaveny proporcionální složky regulátorů na hodnotu 1. To znamená, že při největší chybě byla přední kola vychýlena v plném rejdu a pohon vozidla byl ponechán na volnoběh. Následovalo zkoušení různých hodnot pro proporcionální složku regulátorů. Ukázalo se, že je žádoucí zvětšit P složku regulátoru zatáčení a snížit P složku pro brzdění robota.

Jízdní vlastnosti vozidla byly vylepšeny přidáním derivačních složek do obou regulátorů. Tyto složky se uplatňují především při prudkém nájezdu do zatáček. Derivační složka zde způsobuje větší přibrzdění hnacích kol a ostřejší zatáčení vozidla.

Pokus o přidání integrační složky vedl k rozkmitání a zhoršení řízení, proto nakonec není uplatněna.

V následující tabulce 6.1 jsou uvedeny hodnoty parametrů, které byly nastaveny při posledním testování.

Tabulka 6.1: Parametry regulátorů

Parametr	Hodnota
`Kp`	1,5
`Ki`	0
`Kd`	3
`ThKp`	0,2
`ThKi`	0
`ThKd`	4,5

Testování prokázalo funkčnost robota a upozornilo na jeho nedostatky. Tato zjištění jsou cenným podkladem pro budoucí vylepšování, která budou nastíněna v závěrečné kapitole.

Kapitola 7

Závěr

Záměrem práce bylo vytvoření řídícího systému pro autonomního robota. Tento cíl byl úspěšně splněn, vozidlo je schopné autonomního řízení. Tento projekt byl možností proniknout do nízkoúrovňového programování embedded systémů, které je značně odlišné od abstraktních jazyků z vyšší úrovně.

Programování v low-level jazycích dává vývojáři možnost čistě vidět, jak vyvíjený systém funguje uvnitř. Při porozumění základním principům je možné zefektivnit i složité algoritmy tak, aby je bylo možné spustit na zařízeních disponujících menším výkonem.

Kromě samotného regulátoru pro řízení bylo nutné navrhnout mnoho dílčích algoritmů. Jedním z nich byla základní metoda pro detekci hran v obrazu, která ukázala, že při zpracování dat je někdy lepší použít jinou perspektivu pohledu. Pro úlohu sledování dráhy byl navržen a naprogramován stavový automat.

Vytvořen byl též vizualizační program, který byl následně rozšířen pro potřebu nastavování parametrů regulátoru. Tato část byla vytvořena v programovacím jazyce Python, který narozdíl od jazyka C patří mezi programovací jazyky vyšší úrovně. Pro vytvoření grafického uživatelského prostředí byl využit modul wxPython, což je multiplatformní obal ke knihovně wxWidgets implementované v jazyce C++.

Pro řízení robota byly naimplementovány dva regulátory. První z nich ovládá natočení předních kol a druhý řídí rychlost hnacích motorů.

V průběhu realizace projektu a při závěrečném testování bylo objeveno několik oblastí, které by mohly vést ke zlepšení řízení vozidla.

Jednou z možností pro vylepšení funkčnosti řízení je zdokonalení systému pro automatické nastavení integračního času kamery. Tento nápad byl v posledních dnech práce částečně rozveden a byla provedena základní realizace systému. Ačkoliv algoritmus nastavuje integrační čas správně při pozvolném přechodu osvětlení, skokové změny představují složitější problém.

Dalším vylepšením by mohlo být přidání senzorů. Tím by bylo možné do výpočtů zavést jiné veličiny a zkvalitnit tak řízení.

Funkčnost systému pro sledování čáry byla vyzkoušena na několika tratích. Regulace při těchto testech obstála.

Sledování dráhy bohužel nebylo možné řádně vyzkoušet a zhodnotit. Test proběhl pouze jednou s příznivými výsledky, které bude v budoucnu nezbytné ještě potvrdit.

Reference

[1] ams AG. TSL1401CL 128x1 linear sensor array. 2011.

[2] ARM. Mbed manual. [online, cit. 10. 3. 2015] `https://developer.mbed.org/handbook/Homepage`.

[3] Corrado Guarino Lo Bianco, Aurelio Piazzi, and Massimo Romano. Smooth control of a wheeled omnidirectional robot. 2014. [online, cit. 10. 4. 2015] `http://www.researchgate.net`.

[4] Jason Brownlee. Finite state machines (FSM) Finite state machines as a control technique in Artificial Intelligence (AI). 2009. [online, cit. 20. 4. 2015] `http://www.scribd.com/doc/23724065/Finite-State-Machines-FSM`.

[5] Darren Burnhill. Ackerman steering principle. [online, cit. 10. 4. 2015] `http://www.rctek.com/technical/handling/ackerman_steering_principle.html`.

[6] Doc. Ing. Jiří Melichar, CSc. *Lineární systémy (Učební text).* ZČU Plzeň, katedra kybernetiky, 2001.

[7] Python Software Foundation. Webové stránky Python, [online, cit. 10. 2. 2015] `https://www.python.org/`.

[8] Inc. Freescale. Codewarriror resources. [online, cit. 6. 5. 2015] `https://www.freescale.com/webapp/sps/site/homepage.jsp?code=CW_HOME`.

[9] Desmond King-Hele. *Erasmus Darwin's Improved Design for Steering Carriages.* The Royal Society of London, 2002.

[10] Tzuliang Loh. Interactive control systems tutorial. [online, cit. 16. 4. 2015] `https://www.mathworks.com/academia/student_center/tutorials/`.

[11] G. W. Lucas. A tutorial and elementary trajectory model for the differential steering system of robot wheel actuators. [online, cit. 10. 4. 2015] `http://rossum.sourceforge.net`.

[12] William F. Milliken and Douglas L. Milliken. *Race Car Vehicle Dynamics.* SAE International, 1994.

[13] PuTTY. A Free Telnet/SSH Client [online, cit. 24. 4. 2015] `http://www.chiark.greenend.org.uk/~sgtatham/putty`.

[14] Wikipedia. H bridge. [online, cit. 20. 4. 2015]
`https://en.wikipedia.org/wiki/H_bridge`.

[15] Wikipedia. Pid regulátor. [online, cit. 22. 4. 2015]
`https://cs.wikipedia.org/wiki/PID_regul%C3%A1tor`.