ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA DOPRAVNÍ

Bc. Vojtěch Rulc

Výzkum řízení dynamiky autonomního vozidla

Diplomová práce

2015

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
d ě k a n
Konviktská 20, 110 00 Praha 1

K623 .....Ústav bezpečnostních technologií a inženýrství

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Bc. Vojtěch Rulc

Kód studijního programu a studijní obor studenta:

N 3710 – BD – Bezpečnost dopravních prostředků a cest

Název tématu (česky): **Výzkum řízení dynamiky autonomního vozidla**

Název tématu (anglicky): Research of Autonomous Vehicle Dynamic

Zásady pro vypracování

Při zpracování diplomové práce se řiďte osnovou uvedenou v následujících bodech:

- analýza stavu techniky v oblasti autonomních vozidel
- algoritmus registrace překážky a aktivní reakce vozidla
- algoritmus registrace dopravních informací
- aplikace algoritmů na zvolené technické zařízení
- testování zařízení a zobecnění poznatků

Rozsah grafických prací: dle pokynů vedoucího diplomové práce

Rozsah průvodní zprávy: minimálně 55 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury:
ACHTENOVÁ, Gabriela a KOREC, Ondřej. Autonomous driving algorithms in scaled environment. Journal of Middle European Construction and Design of Cars. 2011-01-1, vol. 9, issue 3, s. -. DOI: 10.2478/v10138-011-0016-y.
VLK, František. Automaticky a autonomně řízená silniční vozidla. Automa. Praha: FCC Public, 2013, roč. 19, č. 1, s. 12-15.

Vedoucí diplomové práce: **prof. Ing. Jan Kovanda, CSc.**
doc. Ing. Hedvika Kovandová, Ph.D.

Datum zadání diplomové práce: **30. června 2014**
(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání diplomové práce: **31. května 2015**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta dopravní
L. S.

doc. Ing. Václav Jirovský, CSc.
vedoucí
Ústavu bezpečnostních technologií a inženýrství

prof. Dr. Ing. Miroslav Svítek
děkan fakulty

Potvrzuji převzetí zadání diplomové práce.

Bc. Vojtěch Rulc
jméno a podpis studenta

V Praze dne..... 30. června 2014

Poděkování

Na tomto místě bych rád poděkoval všem, kteří mi poskytli podklady pro vypracování této práce. Zvláště pak děkuji prof. Ing. Janu Kovandovi, CSc. a doc. Ing. Hedvice Kovandové, Ph.D. nejenom za odborné vedení a konzultování mé práce, ale i za jejich veškerou pedagogickou činnost během mého studia. Dále bych chtěl poděkovat celému týmu podílejícímu se na nárazové zkoušce čelníku tramvaje za jejich účast v soutěži, ve které vyhráli Lego Mindstorms, které jsem mohl využít pro svou práci. Poděkovat musím i firmě KPM Consult za zprostředkování zapůjčení robota DaNI. Chtěl bych poděkovat i svým rodičům za veškerou podporu mého vzdělávání, celé široké rodině za jejich zájem o mé osvojování nových vědomostí, i mým spolužákům, díky nimž během studia vznikla přátelská atmosféra, která studium zpříjemňovala, a v neposlední řadě bych chtěl poděkovat všem mým přátelům za morální podporu, povzbuzování a za skvělé zážitky z období mého studia. Při děkování nesmím zapomenout ani na kolegu Tomáše Dlaska, se kterým jsem na části diplomové práce spolupracoval a který mi byl kdykoliv ochotně nápomocen.

Děkuji!

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci, zpracovanou na závěr studia na ČVUT v Praze, Fakultě dopravní.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 29. května 2015 ……

podpis

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní

Výzkum řízení dynamiky autonomního vozidla

diplomová práce
květen 2015
Bc. Vojtěch Rulc

ABSTRAKT

Předmětem této diplomové práce je výzkum řízení dynamiky autonomního vozidla. Nejprve analyzuje stav techniky autonomních automobilů a dále popisuje vytváření a aplikaci vybraných algoritmů na dva různé roboty, které slouží jako model vozidla. Nakonec z vývoje a testování algoritmů vyvozuje poznatky, které jsou uplatnitelné při další rozvíjející práci v této oblasti.

KLÍČOVÁ SLOVA

autonomní vozidlo, registrace překážky, registrace dopravních informací,
Lego Mindstorms, DaNI, LabVIEW

ABSTRACT

The diploma thesis focuses on research of the autonomous vehicle dynamics. Firstly, the level of technical equipment of contemporary autonomous vehicles is analysed. Farther on, a description of specific algorithms follows in the thesis. The control systems are intended to be used on two different robots, which simulate a vehicle model . Finally, further conclusions are drawn from the development and experimental tests of the algorithms. The derived knowledge are applicable for the future work and research in this area.

KEYWORDS

autonomous vehicle, obstacle recognition, traffic information recognition,
Lego Mindstorms, DaNI, LabVIEW

Obsah

Úvod

Intenzita automobilové dopravy stále stoupá a nezdá se, že by tomu mělo být jinak i v následujících letech. Zahušťující se silnice a neustálé zrychlování životního tempa, které vede i ke snaze urychlování dopravy, vytváří větší a větší nároky na schopnosti řidičů, kteří se v takovém prostředí pohybují. Od určitého bodu již není v lidských silách tento trend fyziologicky stíhat a v tu chvíli se dostávají ke slovu asistenční systémy ve vozidlech, které se snaží snižovat bezpečnostní rizika spojená s lidským faktorem a pomáhat člověku v situacích, na které sám nestačí. Trend vývoje asistenčních systémů naznačuje, že v nedaleké budoucnosti si s člověkem vymění role. Řídicím článkem v automobilu bude technika a asistenčním systémem se stane člověk. Od toho si pak slibujeme zvýšení komfortu cestování, zvýšení bezpečnosti na silnicích, snížení provozních nákladů z důvodu lepšího hospodaření s pohonnými energiemi, a mnoho dalších pozitiv.

Takový automobil, který sám sebe řídí a člověk vykonává pouze funkci supervizora, je označován jako autonomní automobil. A výzkumem řízení dynamiky autonomního vozidla se bude tato diplomová práce zabývat.

Cílem mé diplomové práce je zmapovat současný stav v oblasti autonomních vozidel a následně se pokusit o vytvoření algoritmů, které částečně takové vozidlo napodobí. Tyto algoritmy by pak měly být aplikovány na technické zařízení, na kterém budou otestovány. Z následného výzkumu a testování pak budou vyvozeny poznatky, které by měly sloužit k zefektivnění práce na nějakém podobném výzkumu nebo výzkumu navazujícím.

První kapitola popisuje historický vývoj autonomních automobilů, současný legislativní rámec této oblasti automobilismu a stručně vyjmenovává, jaké technologie jsou dnes využívány a jaký stupeň autonomie zhruba splňují. V neposlední řadě pak přináší několik úvah o problémových aspektech využívání umělé inteligence v automobilech.

Druhá kapitola zmiňuje existenci tzv. Tří zákonů robotiky a ustavuje je jako základní rámec pro všechny algoritmy, které budou v rámci této práce prezentovány.

Kapitola třetí už se zaobírá tvorbou prvního algoritmu a jeho aplikací na robota Lego Mindstorms NTX 2.0, který posloužil jako odrazový můstek pro další práci.

Ve čtvrté kapitole je popsán robot DaNI, na kterého jsou aplikovány další algoritmy, a který je pro tuto práci stěžejní. Je charakterizována nejenom jeho standardní výbava, ale i výbava, o kterou musel být pro potřeby výzkumu doplněn.

Pátá kapitola vysvětluje, základní principy algoritmů, které byly na robota aplikovány, a upozorňuje na problémy, které se vyskytly jak v průběhu programování, tak v průběhu testování.

Pro vypracování diplomové práce je snaha používat nejenom tištěnou literaturu, ale hlavně i internetové zdroje, které mají tu výhodu, že jsou aktuálnější, což je v tak rychle se rozvíjejícím oboru, jako jsou autonomní vozidla, velmi důležité.

1 Analýza stavu techniky v oblasti autonomních vozidel

V rámci teoretické části této práce jsem se pokusil zmapovat stav techniky v oblasti autonomních vozidel. Vzhledem k tomu, že se jedná o téma, které je pro výrobce automobilů z důvodu konkurenční výhody velmi citlivé, vycházel jsem pouze z informací veřejně známých a uveřejněných v médiích.

1.1 Historie

S nadsázkou se dá říct, že prvního autonomního řízení dopravního prostředku bylo dosaženo kolem roku 10 000 př. n. l., kdy se člověku poprvé podařilo absolvovat na hřbetu koně celou cestu až do cílové destinace bez dávání příkazů koni. [1]

Nicméně nejednalo se o automobil, a proto počátek autonomních automobilů musíme v historii hledat mnohem později. Různé více či méně úspěšné pokusy o představení poloautonomních automobilů začaly překvapivě už v 20. letech 20. století v automobilové velmoci – ve Spojených státech amerických. Ač tyto automobily jezdily bez posádky a ve své době byly jistě nepředstavitelným sci-fi, které se stalo skutečností, byly ve skutečnosti „pouze“ dálkově radiově řízeny. V dalším vývoji, především v 60. a 70. letech 20. st., byly představovány automobily, které byly již téměř autonomní, ale pro svůj provoz potřebovaly uzpůsobenou dopravní infrastrukturu. Například magnetické smyčky a kabely ve vozovce. [2]

Až roku 1987 se po několikaletém vývoji a testování podařilo Ernstu Dickmannsovi a jeho týmu z Bundeswehr University Munich vybavit dodávku Mercedes-Benz senzory a výpočetní technikou tak, že byla schopna samostatné jízdy v prázdných mnichovských ulicích rychlostí až 96 km/h. [2]

V témž roce v USA představila DARPA (Defense Advanced Research Projects Agency – Agentura pro výzkum pokročilých obranných projektů) svůj automobil, který v komplexním off-road terénu s prudkými svahy, průrvami, velkými kameny a vegetací urazil přes 600 m dlouhou trasu rychlostí 3,1 km/h. [2]

Obr. 1.1 – Autonomní Mercedes-Benz Ernsta Dickmannse

V 90. letech Ernst Dickmanns své autonomní automobily stále vylepšoval a absolvoval s nimi několik zajímavých předváděcích akcí. Příkladem budiž 1 000 km dlouhá jízda na tříproudé pařížské dálnici v roce 1994, kde identické semi-automatické automobily VaMP a Vita-2 předvedly své schopnosti v normálním provozu až do rychlosti 130 km/h. Tyto automobily byly ještě značně ovlivněny lidským zásahem do řízení, ale již o rok později, roku 1995, tomu bylo jinak. Dickmannsův Mercedes třídy S urazil 1 590 km dlouhou trasu z Mnichova do Kodaně a zpátky, na německých dálnicích místy až rychlostí 175 km/h, a to na 95 % trasy naprosto autonomně. Nejdelší úsek, který zvládl bez lidského zásahu, byl dlouhý 158 km. [2]

Ve stejné době se o podobný čin pokoušely i jiné výzkumné týmy, avšak nebyly již tak úspěšné. Carnegie Mellon University dosáhla na 5 000 km dlouhé trase 98,2% autonomního řízení, avšak jejich automobil autonomně pouze zatáčel, plyn a brzda byly ovládány člověkem. Dalším, kdo se o něco podobného pokusil, byl tým kolem Alberta Broggi z Parmské univerzity v Itálii. Jejich automobil zvládl průměrnou rychlostí 90 km/h ujet 1 900 km dlouhou trasu na severoitalských dálnicích s tím, že automobil se choval autonomně na 94 % cesty. Zajímavé na tomto projektu však bylo, že automobil byl, co se senzorů týče, vybaven pouze dvěma levnými černobílými kamerami, které na základě stereoskopického algoritmu vyhodnocovaly okolí automobilu. [2]

V roce 2004 vyhlásila vláda Spojených států amerických soutěž DARPA Grand Challenge. Americká armáda si od této soutěže slibovala urychlení vývoje autonomních automobilů tak, aby mohla v roce 2015 mít třetinu vozového parku již autonomní. Úkolem

zúčastněných týmů bylo zkonstruovat plně autonomní automobil, který zdolá trasu dlouhou 240 km vedoucí oblastí Mohavské pouště. Během prvního ročníku se žádnému ze soutěžních vozů do cíle dostat nepodařilo, a tak odměna dvou milionů dolarů pro vítěze nebyla vyplacena. Tato soutěž proběhla ještě o rok později, kdy se již trasu projet povedlo. V roce 2006 byla soutěž modifikována a soutěžící automobily musely být schopny se řídit pravidly silničního provozu. Soutěž probíhala na uzavřené americké vojenské základně George Air Force Base. Vítěznému automobilu se trasu dlouhou 96 km podařilo projet za méně než 6 hodin. [3]

V současné době na vývoji autonomního automobilu pracují všechny velké automobilky na světě. Jejich vyvinuté systémy jsou do praxe uváděny postupně ve formě semi-automatizace automobilů. Ať už se jedná o systém udržení jízdního pruhu, adaptivní tempomat nebo třeba autonomní parkování.

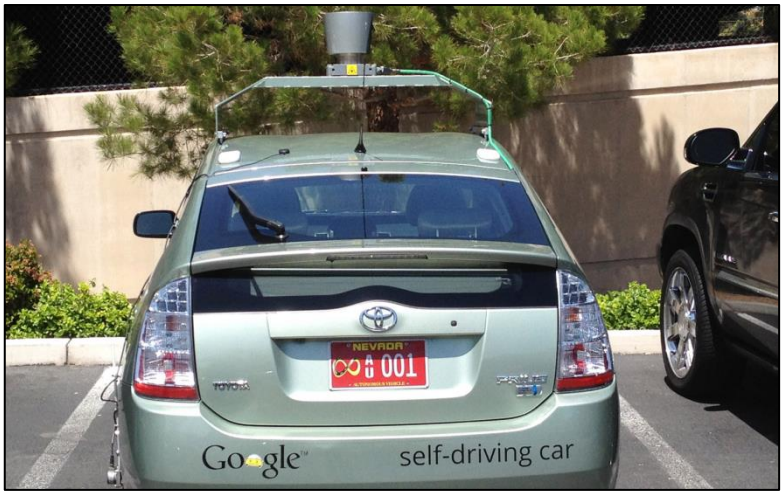
(Semi-)automatizace automobilů přináší i další pozitiva, na která se občas zapomíná. Díky tomu, že automobily jsou vybaveny stále větším množstvím senzorů a dokáží vyhodnocovat okolní prostředí a situace, ve kterých se pohybují, mohou díky tomu měnit i své jízdní vlastnosti (např. změnou nastavení odpružení) nebo zajišťovat korekční funkce v řízení neautonomního vozidla. [4]

1.2 Legislativa

Autonomní automobily jsou v současné době pro zákony naprosté většiny zemí úplně neznámým pojmem. Jejich existence není v zákonech nijak podchycena, natož aby byla nějak regulována. S tímto problémem se především po lobbistickém nátlaku Googlu, který sám vyvíjí autonomní automobily, začaly zabývat USA na úrovni jednotlivých států.

První, kdo přijal legislativu výslovně umožňující provoz autonomních automobilů, byl stát Nevada. Stalo se tak 16. června 2011. Od 1. března 2012 pak příslušný zákon vstoupil v platnost. Tímto zákonem byl jako orgán zodpovědný za bezpečnostní standardy určen Nevada Department of Motor Vehicles. Tato agentura zároveň vymezuje oblasti, ve kterých smí být automobily testovány. Příslušný zákon autonomní automobil definuje jako „motorové vozidlo, které používá umělou inteligenci, senzory a GPS k řízení sebe sama bez aktivního zásahu člověka“. Dále je v něm uvedeno, že „řidič“ nemusí udržovat pozornost ve chvíli, kde se automobil řídí sám. Google se snažil prosadit, aby se na „řidiče“ těchto automobilů nevztahoval zákaz používání mobilních

telefonů. Nicméně tento návrh prosazen nebyl. Kromě toho je ještě nařízeno, že při testech musí být v automobilu přítomna jak osoba na sedadle řidiče, tak i jeden cestující. [2]



Obr. 1.2 – První registrovaný autonomní automobil

V květnu 2012 byla v Nevadě udělena první licence pro autonomní automobil. Byla udělena upravené Toyotě Prius, která patří společnosti Google. Tento i další autonomní automobily jsou označeny červenou registrační značkou, na níž je znak pro nekonečno ∞, jež má symbolizovat auto budoucnosti, a poté vlastní číslo vozu. [2]

Dalším státem USA, který vytvořil zákon zabývající se touto problematikou byla Florida, která v červenci 2012 jen jednoduše deklarovala, že „stát Florida nezakazuje ani nereguluje testování nebo provoz autonomních automobilů na veřejných silnicích“. [2]

Dále následovaly státy California a Michigan. Ne všude ovšem byly podobné regulace přijaty. Zákon, který byl navržen v Coloradu, schvalovacím řízením neprošel. [2]

Průkopníkem v Evropě se stalo Spojené království Velké Británie a Severního Irska, které testování autonomních automobilů na veřejných silnicích povolilo v roce 2013. Do té doby směly být v Británii robotické automobily testovány výhradně na soukromých pozemcích. [2]

Žádné další státy zatím existenci autonomních automobilů do svých předpisů nepromítly, což může být problémem nejen v blízké budoucnosti, ale už i v současnosti. Už nyní totiž spousta automobilů nabízí systémy podpory řízení, ve kterých software de facto přebírá kontrolu nad vozem a řidič se tomu buď nemůže bránit nijak, nebo musí k převzetí kontroly učinit kroky, které ho při řešení krizové situace zdrží. Příkladem

nechť jsou systémy automatického intenzivního brzdění, které sice mohou odvrátit srážku hrozící přední části vozidla, ale prudkou decelerací mohou naopak zapříčinit řetězovou srážku automobilů. A člověk, na rozdíl od těchto systémů má schopnost méně či lépe zhodnotit, která z těchto srážek je v dané dopravní situaci méně vážná.

Ať se nám to líbí nebo ne, automobily stále více a více rozhodují za nás a stále více se řídí samy. Tento fakt, ale v legislativě není nijak podchycen, a proto i nadále zůstává z hlediska práva za chování automobilu zodpovědný řidič. A to i v případech, kdy je mu řízení automobilu odepřeno nějakým aktivním bezpečnostním systém.

1.3 V současnosti využívané technologie

Na základě průzkumu mezi 1500 respondenty trhu ve Spojených státech amerických, se odhaduje, že přibližně čtvrtina lidí je ochotná připlatit si za takřka plně autonomní vozidlo 4000 až 5000 $. Ovšem vzhledem k tomu, že nyní například jenom zařízení GPS, které dokáže zjišťovat polohu s přesností na centimetry, stojí zhruba 8000 $ a náklady na pořízení LIDARu se pohybují kolem 7000 $, je pochopitelné, že plně autonomní automobily by se dnes na trhu příliš neuplatnily. Proto se nejprve musí zlevnit technologie, které autonomní provoz automobilu umožňují. K tomu, aby se do komerčního provozu dostala plně autonomní vozidla, dojde podle některých odhadů kolem roku 2025. [5]

1.3.1 BMW

Nejpokročilejším semi-autonomním veřejnosti známým modelem automobilu značky BMW je upravené BMW řady 5. Tento automobil je od jiných neupravených modelů na první pohled k nerozeznání, ale je vybaven několika senzory, které automatické řízení zajišťují. Jedná se o dvě videokamery umístěné za čelním a zadním oknem, které zajišťují rozpoznávání vodorovného a svislého dopravního značení. Uvnitř předního a zadního nárazníku jsou umístěny dva laserové scannery a 3 radarové senzory, které kontrolují silnici před a za vozidlem. Vedle obou zpětných zrcátek jsou dva širokoúhlé laserové scannery, které monitorují levé a pravé okolí vozu. Čtyři ultrazvukové senzory umístěné nad koly zase kontrolují oblast blízko automobilu. V neposlední řadě je automobil vybaven ještě přijímačem diferenciální GPS, který automobilu sděluje jeho polohu s přesností na několik centimetrů. [6]

1.3.2 Mercedes-Benz

Model Mercedes-Benz S 500, který je již na trhu, umí kromě jiného třeba následovat automobil jedoucí před ním, nebo zastavit před překážkou, která mu stojí v cestě. Mercedes-Benz ovšem využívá ve svém automobilu trochu odlišnou techniku než BMW. I Mercedes je vybaven ultrazvukovými senzory, radarem a videokamerou, která rozeznává dopravní značení, ale krom tohoto má i stereo-videokameru, díky které vidí objekty před vozidlem prostorově. Zároveň je vybavena i infračervenou kamerou, která pravděpodobně pomáhá ostatním kamerám při snížené viditelnosti. [6]

1.3.3 Nissan

Modifikovaný Nissan Leaf EV, který se v listopadu 2013 začal testovat na japonských dálnicích, je schopen rychlostí 40 km/h najet na dálnici, pohybovat se po ní rychlostí 80 km/h, předjíždět pomalejší vozy a z dálnice zase sjet. [7] Za tímto účelem je vybaven předním a zadním radarem, videokamerou, předními, zadními a postranními laserovými scannery a čtyřmi širokoúhlými kamerami, které ukazují řidiči okolí vozu. [6]

1.3.4 Google

Společnost Google využívá při vývoji autonomních automobilů upravené modely Toyoty Prius a Lexusu. Ty mají na střeše upevněný LIDAR, který prostorově detekuje objekty v okolí vozu, a kamery, které této detekci pomáhají. Zároveň jsou tyto automobily vybaveny předním a zadním radarem, inerciální měřicí jednotkou určující pozici vozidla, speciálním otáčkoměrem kol a velmi přesnou digitální mapou. [6]

1.4 Problémový aspekt umělé inteligence

Ačkoliv jsou autonomní automobily zajisté součástí technologického vývoje a lze předpokládat, že jejich zavedení přinese mnoho pozitiv, které si každý dovede představit, je třeba upozornit i na problémy, se kterými se autonomní automobily budou, minimálně zpočátku, střetávat.

Především to bude psychický blok cestujících, kteří budou mít problém přijmout fakt, že jejich zdraví a život leží plně v rukou algoritmu řízení, kterému budou muset slepě důvěřovat. Kvůli tomuto, myslím si, nebude mít penetrace autonomních automobilů v celém vozovém parku takový rapidní růst, jaký předpovídají experti z oblasti automobilového průmyslu.

Další problém vyvstane ve chvíli, kdy bude muset řidič řídit automobil manuálně. Takový řidič, který bude jezdit naprostou většinu času „na automat“ nebude mít tuto činnost dostatečně procvičenou tak, aby byl schopen vozidlo ovládat bezpečně. S podobným problémem se potýkají i piloti dopravních letadel, kteří po téměř celou dobu letu využívají autopilot. U nich je tento problém eliminován pravidelnými lety na simulátorech, kde musí uměle vyvolané krizové situace řešit. Neumím si však podobnou praxi představit u řízení automobilů širokou veřejností.

Dle mého názoru se největším úskalím užívání autonomních automobilů stane kyberbezpečnost. Z automobilu se stane, vlastně je z něj již nyní, počítač, který je virtuálně napadnutelný. Riziko napadení budeme moci sice snižovat, ale nikdy jej nebudeme moci eliminovat úplně. A autonomní automobil, nad kterým získá kontrolu vnější útočník, nebo počítačový vir, představuje nebezpečí, které si dokážeme do největších důsledků jen stěží představit.

2 Tři zákony robotiky

Tři zákony robotiky jsou zákony, které ve své sci-fi povídce Hra na honěnou (angl. Runaround), odehrávající se mimochodem v roce 2015, definoval v roce 1942 spisovatel Isaac Asimov [8]. Ač se může zdát, že úryvky sci-fi literatury do diplomové práce technické vysoké školy nepatří, nelze tyto zákony právě u tohoto tématu nezmínit. Postupem času se totiž staly základními dogmatickými pravidly, které jsou na roboty kladeny, a mají svůj význam především z pohledu bezpečnostního inženýrství. Jejich znění je následující:

> *1) Robot nesmí ublížit člověku nebo svou nečinností dopustit, aby bylo člověku ublíženo.*
>
> *2) Robot musí uposlechnout příkazů člověka, kromě případů, kdy jsou tyto příkazy v rozporu s prvním zákonem.*
>
> *3) Robot musí chránit sám sebe před poškozením, kromě případů, kdy je tato ochrana v rozporu s prvním nebo druhým zákonem.* [9]

V roce 1952 Asimov tato tři pravidla doplnil ještě o nultý zákon. Byl publikován v povídce Nadace a říše (Foundation and Empire) a zní takto:

> *0) Robot nesmí ublížit lidstvu nebo svou nečinností dopustit, aby lidstvu bylo ublíženo.* [8]

Položíme-li si otázku, kteří roboti by měli tyto zákony dodržovat, odpovíme si přirozeně, že všichni. Na druhou stranu ale musíme dodat, že v dnešní době, a netroufám si odhadovat, zda to v budoucnu bude jinak, jsou roboti jen strojem plnícím příkazy vloženého algoritmu bez nějaké skutečné umělé nebo přirozené inteligence, takže dodržování těchto zákonů závisí na lidském faktoru – především na programátorovi. Takže ve výsledku tyto zákony definovaly základní rámec, uvnitř kterého se musím v rámci celé práce pohybovat. Lze namítnout, že tento rámec je natolik přirozený, že ho netřeba zmiňovat, ale tomu bych oponoval tím, že je dobré ho mít jasně uspořádaný a definovaný.

3 Lego Mindstorms NTX 2.0

Jedná se o stavebnici lega, která je doplněna o aktivní prvky. Stavebnice kromě tradičních lego dílků obsahuje „NTX kostku“, což je ve skutečnosti malý počítač s jednoduchým displejem. Do této výpočtové jednotky lze pomocí konektorových kabelů RJ12 zapojit 3 servomotory a 4 senzory. NTX kostka dokáže i komunikovat pomocí bezdrátového standardu Bluetooth, které slouží hlavně pro propojení s počítačem a pro dálkové ovládaní. Přehledná a detailnější specifikace NTX kostky je uvedena v tabulce 3.1. V balení Lego Mindstorms jsou 2 dotykové senzory, jeden ultrazvukový, a jeden senzor rozpoznávající barvy. Dále lze dokoupit senzory vyráběné třetími stranami. Na trhu jsou k dostání měřiče teploty, kompas, gyroskop, infračervený přijímač, RFID čtečka, akcelerometr, měřič tlaku, voltmetr, dotykový panel, infračervený měřič kratších vzdáleností a videokamera. Jejich napojení na jednu NTX kostku je však omezeno výše zmíněným počtem 4 senzorových slotů.

Obr. 3.1 – Lego Mindstorms v použitém sestavení

Tuto stavebnici lze v základní sestavě v současné době pořídit za cenu okolo 600 $. Cena jednotlivých senzorů se zpravidla pohybuje mezi 20 a 50 $. Kromě kamery, jejíž cena se pohybuje kolem 150 $.

Nicméně dnes už lze sehnat i novější evoluci této stavebnice. Nejnovější je Lego Mindstorms EV3, které je zpětně kompatibilní a lze ho koupit paradoxně za nižší cenu – okolo 400 $.

Pro mé potřeby jsem za spolupráce s kolegou Dlaskem sestavil pásové vozidlo se dvěma aktivními servomotory a ultrazvukovým senzorem. Servomotory jsou opatřeny optickým snímačem otáček, který dle oficiální specifikace pracuje s přesností 1° [10]. Ultrazvukový senzor měří vzdálenost překážky před ním až do 233 cm, a to s přesností na 3 cm [10].

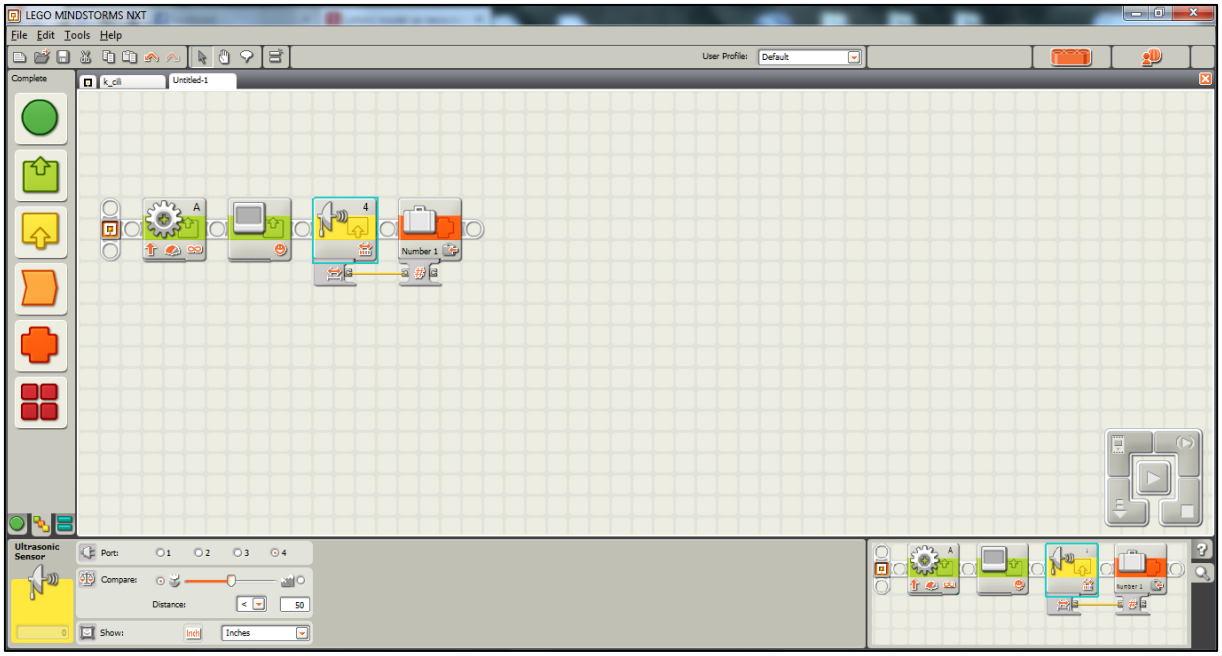
Tabulka 3.1 – Technická specifikace NTX kostky [11]

Procesor	Atmel 32-Bit ARM AT91SAM7S256 48 MHz 256 KB FLASH-RAM 64 KB RAM
Co-Procesor	Atmel 8-Bit AVR, ATmega48 8 MHz 4 KB FLASH-RAM 512 Byte RAM
Operační systém	Proprietární
Senzorové porty	4, analogový digitalní: 9600 bit/s (IIC)
Motorové porty	3, digitální
Komunikace	Bluetooth USB 2.0
USB komunikace	Plná rychlost (12 Mbit/s)
Uživatelské rozhranní	4 tlačítka
Obrazovka	LCD, monochromatická, 100 x 64 px

3.1 Vývojové prostředí

Lego Mindstorms se dá programovat v mnoha jazycích a prostředích. Namátkou zmíním Actor-lab, Enchanting, LabVIEW, Robolab, Robomind, RoboRealm, ROS a Lego

Mindstorms NTX Support from Simulink [10]. Já jsem si vybral prostředí, které Lego dodává v základní sestavě stavebnice, a to Lego MINDSTORMS NXT 2.0 Software. Předpokládal jsem, že toto vývojové prostředí, tím že je oficiální a dodává se se stavebnicí na CD, bude odladěné, a tedy nevzniknou nějaké softwarové chyby, které bych jako začátečník těžko odlišovali od programovacích chyb, a zároveň by mohlo být uzpůsobené i lidem, kteří s podobným programováním nemají žádné zkušenosti.



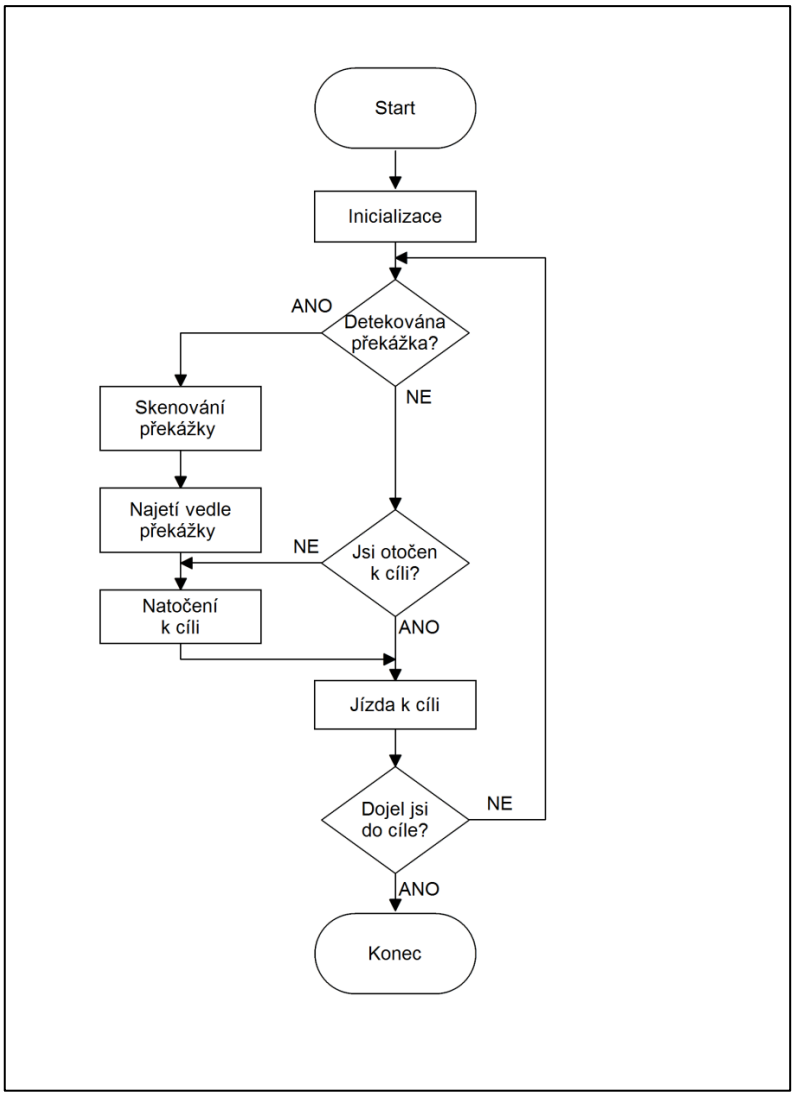
Obr. 3.1 – Ukázka Lego MINDSTORMS NXT 2.0 Software

V tom se mé předpoklady potvrdily. Toto vývojové prostředí je jednoduché, tvorba v něm intuitivní a po krátké chvíli pochopí jeho filosofii každý. Jedná se o vizuální programování, které funguje na principu přemisťování a propojování základních stavebních bloků, jež se de facto rovnají do blokových schémat. Takovýto způsob programování je uživatelsky přívětivý i k dětem, o což patrně společnosti Lego šlo. Zaměření na přívětivost k dětem je znatelná i z vizuálního zpracování programovacího prostředí, které oplývá mnoha barvami a tvary, jež děti zaujmou. Ovšem vizuální rozmanitost s sebou bohužel přináší i problém snížené přehlednosti. Po tom, co je do plochy vyskládáno větší množství stavebních bloků, začíná mít uživatel problém se v celém algoritmu orientovat. K usnadnění orientace mu sice slouží volitelné okno v pravé dolní části obrazovky, jež zobrazuje přehled celého kódu a červeným obdélníkem naznačuje výřez, který právě zobrazuje hlavní okno, ale toto nemusí být úplně dostatečné. Poměrně užitečnou funkcí, která by práci zpřehlednila, by byla funkce

„Lupa“ („Zoom“), ovšem takovouto funkci program postrádá. Zároveň jako negativní stránku tohoto vývojového prostředí vidím jeho hardwarovou náročnost. Výrobce sice uvádí, že systémové požadavky pro tento software jsou: Dual core processor 2.0 GHz nebo vyšší a paměť RAM 2GB nebo vyšší [12], ale vývoj algoritmu pro Lego Mindstorms NTX jsem prováděl na notebooku Asus K50AB s těmito základními hardwarovými parametry: AMD Athlon X2 Dual-Core QL-65 2,10 GHz, 4 GB RAM, ATI Radeon HD 4570 (512 MB); a při dokončování programování už se práce stávala kvůli dlouhým prodlevám a zamrzání velmi neproduktivní, až frustrující. Vzhledem k tomu, že se v základu jedná o hračku pro děti, přijde mi náročnost programu neúměrná a věřím, že při větší optimalizaci by bylo možné ji snížit.

3.2 Popis algoritmu použitého na Lego Mindstorms

Mnou navržený a posléze naprogramovaný algoritmus dokáže dovést robota z výchozí pozice na zadané cílové souřadnice a en-route se vyhnout překážkám. Celý algoritmus lze rozdělit do jednotlivých fází, které v této kapitole popíšu. Pro správné zvládnutí pohybu se robot musí správně orientovat v prostoru, resp. na ploše. A k tomu je potřeba několik goniometrických úvah a výpočtů, které budou uvedeny taktéž.



Obr. 3.2 – Zjednodušené schéma použitého algoritmu

3.2.1 Iniciace

Při spuštění programu dojde nejprve k iniciaci souřadného systému. Souřadný systém je zadán takto: kartézský souřadný systém počátkem souřadného systému ležícím ve středu robota, který stojí v místě startu, x-ová osa je rovnoběžná s příčnou osou robota (kladné hodnoty v pravém směru) a y-ová osa je rovnoběžná s podélnou osou robota (kladné hodnoty v dopředném směru). Lze tedy říci, že souřadnice cíle nejsou vztaženy k místnosti, nebo Zemi, ale jsou vztaženy k robotovi. To proto, že robot sám o sobě nemá žádný senzor, z něhož by při iniciaci získal informace o své poloze vůči okolnímu prostředí. Je tedy jednodušší vytvořit souřadný systém kolem něho a jeho pohyb prostorem vztahovat k výchozímu bodu.

Na závěr iniciační fáze je nastavena ještě dvouvteřinová prodleva, aby nedošlo k nějakým nepodchyceným přechodovým jevům, které by mohly ovlivnit přesnost senzorů a motorů.

3.2.2 Detekce překážky

Tato fáze se cyklicky opakuje a zjišťuje se v ní, zda ultrazvukový senzor při jízdě vpřed zaznamenal překážku, která je blíže než 15 cm. Zde se algoritmus větví. Pokud překážka detekována není, provede se fáze *3.2.3 Natočení k cíli* a následně *3.2.4 Jízda k cíli*. Pokud však překážka zaznamenána je, proběhne *3.2.5 Skenování překážky* a *3.2.6 Najetí vedle překážky*.

3.2.3 Natočení k cíli

Dříve, než robot vyjede (při prvním zjišťování přítomnosti překážky robot ještě nevyjel), musí se otočit ke svému cíli. Jak již bylo uvedeno v kapitole *3.2.1*, je cíl zadán pomocí souřadnic. Díky tomuto zadávání cíle v souřadnicovém systému, můžeme k výpočtu natočení k cíli využít goniometrických funkcí a následující rovnice:

$$\alpha = \tan^{-1} \frac{x_{cíle} - x}{y_{cíle} - y} \ , \tag{3.1}$$

kde α je úhel, o který se musí robot pootočit, aby mířil k cíli (doprava kladná hodnota, doleva záporná); x a y jsou souřadnice místa, kde se robot aktuálně nachází. Veškeré výpočty vzdáleností uvnitř tohoto algoritmu jsou prováděny v jednotkách centimetrů. To jednak kvůli tomu, že vývojové prostředí nabízí výstup z ultrazvukového senzoru pouze v centimetrech, nebo palcích, ale jednak i z toho důvodu, že vzhledem k velikosti robota a rozloze prostředí, ve kterém se pohybuje, považuji použití centimetrů praktičtější, než užívání základní jednotky SI – metr.

3.2.4 Jízda k cíli

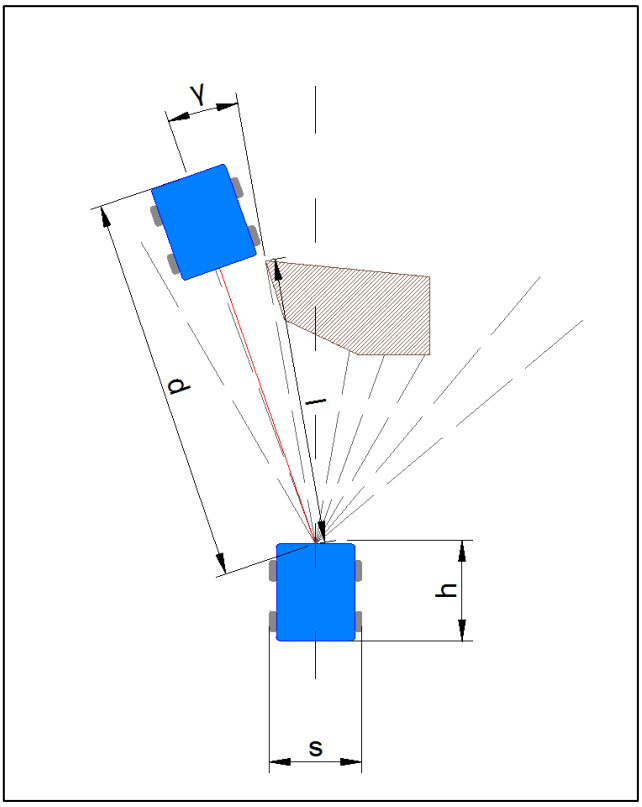
Po otočení robota k cíli je uveden do pohybu směrem přímo vpřed, který probíhá buď doby, než ultrazvukový senzor zaznamená překážku, nebo do chvíle, kdy dosáhne cíle. To je zjišťováno neustálým přepočítáváním aktuální polohy určené na základě předchozí polohy a detekce počtu otáček motorů. Vzhledem k tomu, že je potřeba počítat s nepřesnostmi senzorů, je možné výpočet zjednodušit. Po natočení se k cíli, je vždy vypočtena hodnota *l*, která značí přímou vzdálenost mezi aktuální polohou a cílem. Při jízdě k cíli (ve smyslu fáze algoritmu) je zjišťována ujetá vzdálenost od posledního

otočení se na cíl, která je pak odečtena od vzdálenosti *l*. Je samozřejmé, že není možné tento výpočet provádět ve skutečně kontinuálním čase. Vždy je prováděn v diskrétních časových okamžicích, a tedy není možné, aby rozdíl těchto dvou hodnot byl někdy roven čisté nule. Proto je pro stav, který prohlásíme za dosažení cíle, dána podmínka, že absolutní hodnota rozdílu těchto dvou vzdáleností je menší než 7 cm.

Ve chvíli, kdy je cíle dosaženo, robot zastaví, přehraje oslavnou fanfáru a spuštěný program ukončí.

3.2.5 Skenování překážky

Nalezne-li robot překážku v jeho aktuální dráze, tj. indikuje-li jeho přední ultrazvukový senzor nějaký objekt ve vzdálenosti nižší než 15 cm, zastaví, couvne zpátky, aby nebyl překážce příliš blízko, a začne ji skenovat, jak je na obr. 3.3 naznačeno šedými přerušovanými čarami. Vzhledem k tomu, že ultrazvukový senzor je na robotovi umístěn napevno, musí se robot při skenování překážky otáčet celý. Skenování provádí otáčením se doprava postupně o určitý konstantní úhel, kdy při každém pootočení změří vzdálenost k překážce. V okamžiku, kdy senzor již žádný objekt nezaznamená, nebo jej zaznamená až v dostatečně velké vzdálenosti, pootočí se ještě o poslední krok, aby se ujistil, že se nejedná pouze o úzkou mezeru, nebo chybu měření a následně se otočí zpátky do výchozí polohy, ve které k překážce přijel. Obdobné skenování provede i nalevo a na základě porovnání počtu pootočení vpravo a vlevo vyhodnotí, kterým směrem bude výhodnější překážku objet – která cesta bude kratší.



Obr. 3.3 – Skenování překážky a najetí vedle ní

3.2.6 Najetí vedle překážky

Po oskenování překážky se robot natočí tak, aby mohl najet vedle překážky a přitom s ní nekolidoval. Toho je docíleno pomocí následujícího výpočtu:

$$\gamma = \sin^{-1}\frac{s/2}{l} \tag{3.2}$$

Tento úhel γ se přičte k úhlu pootočení k rohu překážky β na výhodnější straně a robot se může opět vydat na cestu.

Ovšem abychom věděli, jak daleko musí tímto směrem popojet, aby se dostal vedle překážky, kde se zase může natočit zpět na svůj původní cíl, musíme provést ještě jeden výpočet.

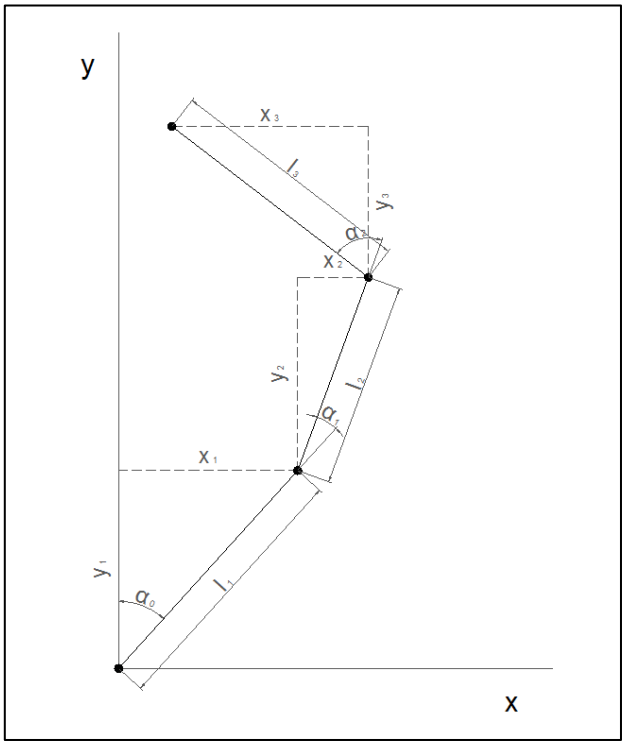
$$d = \sqrt{l^2 - (s/2)^2} + h \tag{3.3}$$

3.2.7 Výpočet aktuální pozice v prostoru

Po najetí vedle překážky je nutné, aby se robot otočil opět směrem ke svému původnímu cíli. K tomu je zapotřebí znát souřadnice místa, ve kterém se robot zrovna nachází. Vzhledem k tomu, že pro tento účel není robot vybaven žádným pozičním senzorem, je

nutné aktuální souřadnice dopočítávat podle směru a délky pohybu, odvozené ze senzorů, které měří otáčky motorů.

Jedná se o tzv. dead reckoning, neboli navigaci výpočtem. Tato obecná metoda je založená na odhadu současné polohy na základě předchozí přesně určené polohy. [13] Vzhledem k tomu, že v našem případě je touto jedinou jistou polohou počáteční souřadnice [0,0], na které se robot nachází při svém startu, je po několika změnách směru jízdy odhad polohy už poměrně nepřesný. O tom se podrobněji zmíním ještě později.



Obr. 3.4 – Výpočet aktuálních souřadnic x, y

K výpočtu se používá následujících rovnic:

$$x_n = x_{n-1} + l \cdot \sin(\alpha + \alpha_{n-1}) \tag{3.4}$$

$$y_n = y_{n-1} + l \cdot \sin(\alpha + \alpha_{n-1}) \tag{3.5}$$

Kde x_n a y_n jsou souřadnice polohy robota po n jeho potočeních a popojetích, l je délka vzdálenosti, kterou robot ujel mezi dvěma pootočeními a úhel α je úhlem, o který byl robot natočen vůči souřadnému systému před tím, než se pootočil a úhel α_{n-1} je úhel, o který se robot naposledy potočil. Přitom se držíme úmluvy, že natočení robota doprava, tj. podle směru hodinových ručiček, chápeme jako pootočení o kladnou hodnotu úhlu α a otočení proti směru hodinových ručiček jako pootočení o zápornou hodnotu úhlu α.

3.3 Poznatky z testování

Algoritmus jsem samozřejmě netestoval až po úplném naprogramování, ale již v průběhu. Vždy po dokončení nějaké dílčí části. To bylo důležité proto, aby se snáze hledaly chyby, které vyvstaly buď při programování, nebo už při úvaze nad algoritmem samotným. V jednoduchých dílčích částech se chyby hledají snáze, než v komplexním celku. A to obzvláště ve vývojovém prostředí, jakým je Lego MINDSTORMS NXT 2.0 Software. Narážím v tuto chvíli na absenci tzv. breakpoints (do češtiny přeložitelné jako „zarážky“), kterými lze spuštěný program postupně zastavovat a provádět inspekci jednotlivých proměnných, čímž lze zjišťovat, zda chyba vznikla již před, nebo až za breakpointem.

Aby bylo možné počítat v metrických jednotkách, bylo nejprve nutné zjistit poměr mezi pootočením motorů a ujetou vzdáleností. Vzhledem k tomu, že teoretický výpočet založený na obvodu kola by nemusel, vzhledem k tomu, že by zanedbával některé skutečnosti jako prokluz a pružnost pásu, odpovídat přesně skutečnosti, rozhodl jsem se zjistit tento poměr („převod“) empiricky. Po provedení empirického měření a průměrování jsem došel k tomu, že pro popojetí o 100 cm je potřeba otočení kol o 3600° a pro otočení robota o 180° je potřeba, aby se motory otočily o 185° (každý jiným směrem).

Při testování první verze programu zajišťujícího *3.2.5 Skenování překážky* jsem objevil skutečnost, kterou jsem při úvodním návrhu nijak neuvažoval. Jde o to, že když se robotovi dá příkaz k otočení se o určitý úhel jedním směrem a poté o stejný úhel opačným směrem – tedy zpátky, nevrátí se robot do původní polohy. Nejmarkantnější je to při okamžité změně směru bez ustalovací pauzy. Lze usuzovat, že problém je spojený se setrvačností robota, setrvačností vnitřních mechanismů motorů a mírně odlišných gumových pásů na levé a pravé straně. Pro řešení tohoto problému se nabízely dvě možnosti:

1) Přestavět robota tak, aby byl ultrazvukový senzor umístěn na otočné hlavici a mohl se otáčet sám bez otáčení celého robota. K tomu by byl použit dosud nevyužitý třetí motor.
2) Vzít v úvahu, že se problém projevoval především u pootočení o velký úhel, a namísto původního záměru provádět skenování překážky vzorkováním

ultrazvukovým senzorem naměřené vzdálenosti během rovnoměrného pohybu, a provádět tuto činnost jednotlivými pootočeními robota o vzorkovací úhel.

Ačkoliv by realizace prvního řešení byla elegantnější a pro pohyb robota i přesnější, rozhodl jsem se z důvodu, že jsem věděl, že po dokončení práce na Lego Mindstorms budeme mít k dispozici robota, který již otočným ultrazvukovým senzorem disponuje, pro účely srovnání obou přístupů, robota nepřestavovat a nechat senzor ve fixní poloze. Tím, že jsem využil druhou alternativu řešení, tedy to, že se robot vždy otočí o určitý malý úhel, zastaví, změří vzdálenost k překážce a teprve poté se znovu otočí o malý úhel, se výše popsaný problém podařilo z velké části eliminovat.

Bohužel se eliminace nezdařila úplně. Problém dále setrvával v tom, že pootočení motorů o řekněme 360° není stejné jako o dvakrát 180°. Ujetá vzdálenost se v obou případech liší. Pro téměř úplné odstranění tohoto problému s ovládáním motorů by bylo potřeba rozkouskovat nejenom otáčení při skenování, ale všechny pohyby, což ovšem samozřejmě není možné. Pohyb robota by byl pomalý a neelegantní. Také by více zatěžoval baterie. Vzhledem k tomu, že práce na *tomto* robotovi má sloužit pouze k základnímu ověření navrženého algoritmu, a ne k nějaké praktické aplikaci, rozhodl jsem se tento problém dále neřešit a pouze počítat s tím, že pohyb robota a dead reckoning bude nepřesný.

Další problematickou záležitostí se ukázal vyzařovací kužel ultrazvukového senzoru. Stalo se tomu ale, musím přiznat, jen kvůli mé zkreslené představě o fungování tohoto senzoru. Představoval jsem si, že ultrazvukový senzor měří vzdálenost obdobně jako laser pomocí jakéhosi paprsku, ovšem v tomto případě zvukového. Což je samozřejmě nesmysl. Můj omyl mi byl jasný hned, jakmile senzor začal hlásit i překážky, které neležely přímo v jeho ose. Došlo mi, že tento senzor, vzhledem k principu šíření zvuku, detekuje objekty ve vyzařovacím kuželu a pro potřeby geometrických výpočtů bude potřeba zjistit, jak je velký. Opět jsem tento parametr zjišťoval empiricky a došel jsem k tomu, že ultrazvukový senzor vyzařuje kužel s vrcholovým úhlem o velikosti 5°. Tato vlastnost ultrazvukového senzoru mi nakonec, ač se to původně zdálo jako komplikace, ve výsledku ušetřila práci a část kódu, čímž celkový kód zjednodušila, a vlastně i optimalizovala. Jde o to, že při paprskovém měření, by se při skenování překážky musel robot po zaznamenání rohu překážky ještě několikrát pootočit, aby ověřil, zda je vedle překážky dostatečná mezera, aby jí mohl projet, zatímco u kuželového měření v určité

vzdálenosti už senzor proměřuje tak širokou oblast, že již na jedno měření lze prohlásit, zda zjištěnou mezerou robot projede.

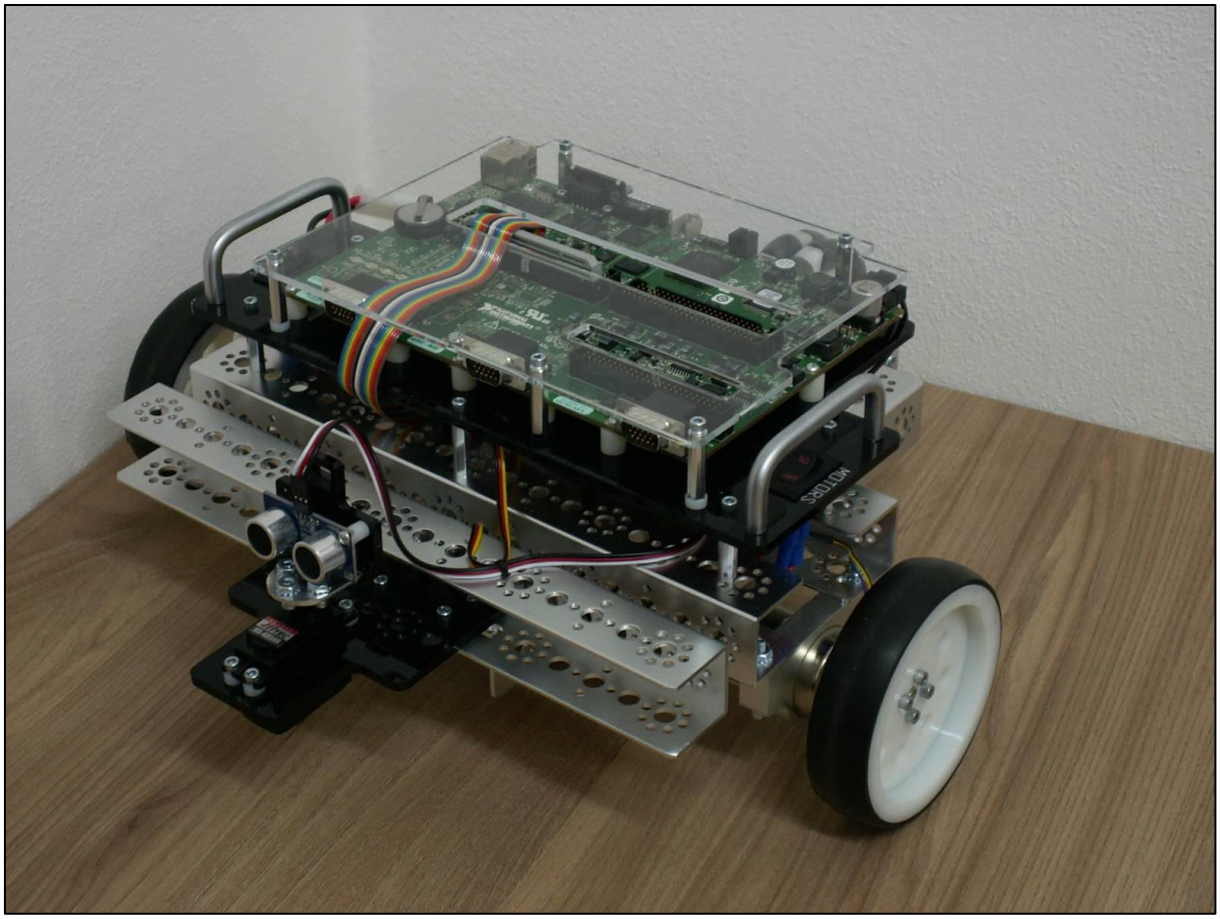
Dalším nepříjemným zjištěním, které jsem ovšem předpokládal od začátku a které se později pouze potvrdilo, bylo chování robota na rozdílných površích. Protože pohyb pomocí pásového mechanismu využívá při zatáčení smýkání pásů, ukázalo se, že poměr mezi úhlem otočení kola a zatočením robota je na každém povrchu jiné. Tím pádem je nemožné nechat robota jezdit na rozdílných površích a očekávat od něj pohyb se stejnou přesností.

Z těchto zjištění vyplývající doporučení pro kohokoliv, kdo by na práci na Lego Mindstorms navazoval, si dovolím přehledně uvést v následujících bodech:

- Jednotlivé části programu testovat zvlášť.
- Využít třetí motor pro otáčení ultrazvukového senzoru.
- Pokusit se zajistit nějakým způsobem homogenizaci a identičnost obou gumových pásů.
- Nechápat zkoumaný prostor před ultrazvukovým senzorem jako jednodimenzionální paprsek, ale jako kužel.
- Vytvořit podpůrný program sloužící pro zjišťování poměru mezi otočením kola a pohybem robota, který následně umožní snadnou kalibraci pokaždé, kdy bude robot předváděn na jiném povrchu.

4 NI LabVIEW Robotics Starter Kit 2.0 – DaNI

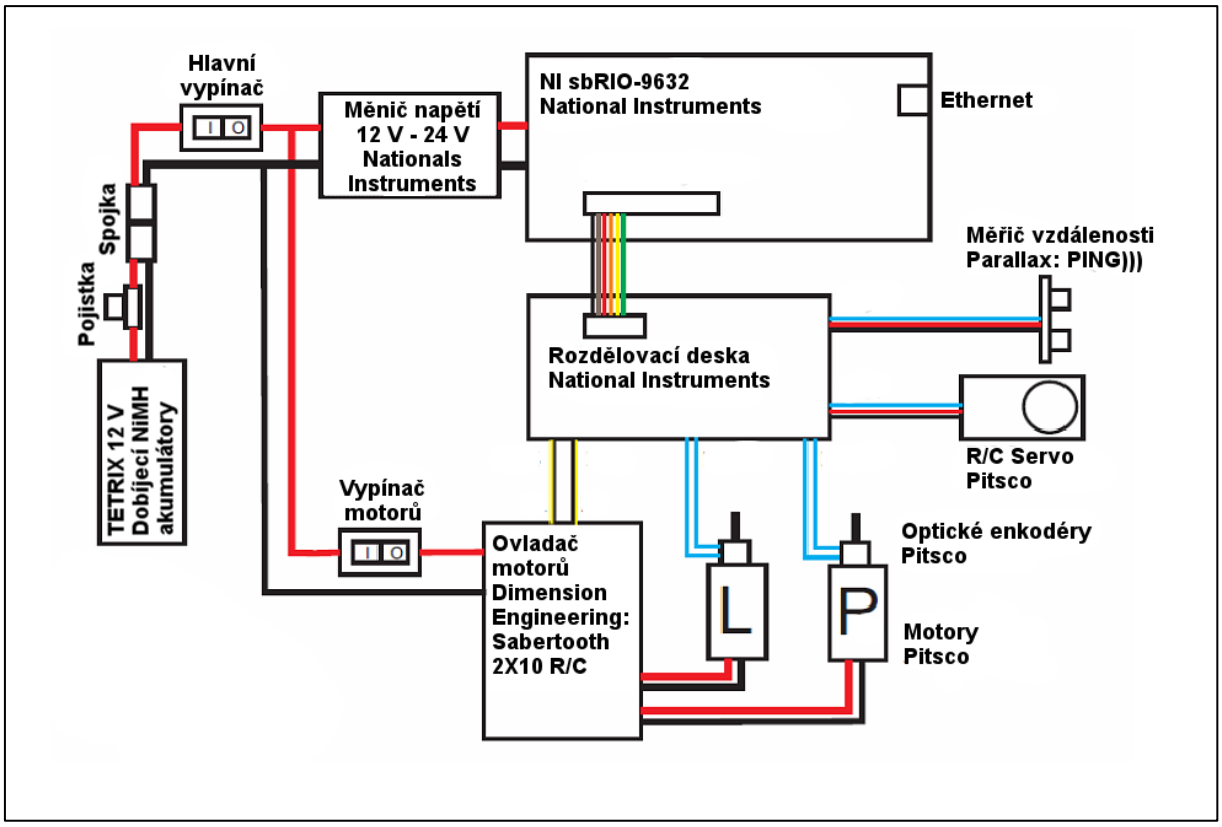
Další má aplikace algoritmů do robota využívala poněkud sofistikovanější řešení, než je Lego Mindstorms, a to NI Robotics Starter Kit 2.0 přezdívaný DaNI. Jedná se o sadu hardwaru a podpůrného softwaru, který má usnadňovat vytváření prototypů v oblasti pojízdných robotů.



Obr. 4.1 – NI LabVIEW Robotics Starter Kit 2.0 – DaNI

4.1 Standardní vybavení robota DaNI

Tato sada obsahuje NI sbRIO-9632, ultrazvukový měřič vzdálenosti Parallax PING))) a baterie, to vše umístěné na pojízdné platformě opatřené dvěma elektromotory poháněnými koly a třetím volným všesměrovým kolem (tzv. omni-wheel).



Obr. 4.2 – Schéma zapojení součástí NI LabVIEW Robotics Starter Kit 2.0 [14]

4.1.1 NI sbRIO-9632

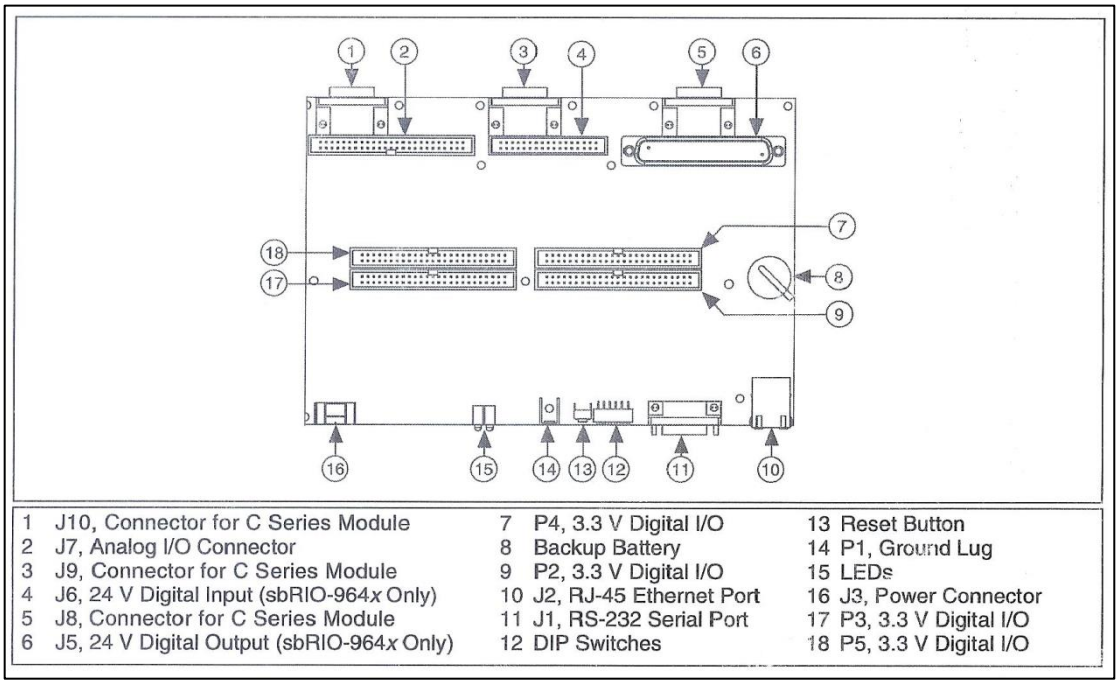
NI Single board reconfigurable I/O (NI sbRIO), tedy jednodeskový nastavitelný systém se vstupy a výstupy je programovatelný vestavěný systém s real-time procesorem, programovatelným rozhraním FPGA (programovatelné hradlové pole) a analogovými a digitálními vstupy i výstupy [15].

Konkrétně NI sBRIO-9632 obsahuje:

- 2M hradel Xilinx Spartan FPGA
- 400 MHz procesor
- 128 MB DRAM a 256 MB permanentní paměti
- Sériový port RS232 pro periferní zařízení
- 110 3.3 V (5V/TTL kompatibilní) digitálních I/O
- 2 se společnou zemí a 16 diferenciálních 16 bitových analogových vstupů se vzorkováním 250 kS/s

- 4 16 bitové analogové výstupy se vzorkováním 100 kS/s
- 10/100 BASE-T Ethernet port
- Napájecí vstup na 19 až 30 V stejnosměrného proudu

[14]



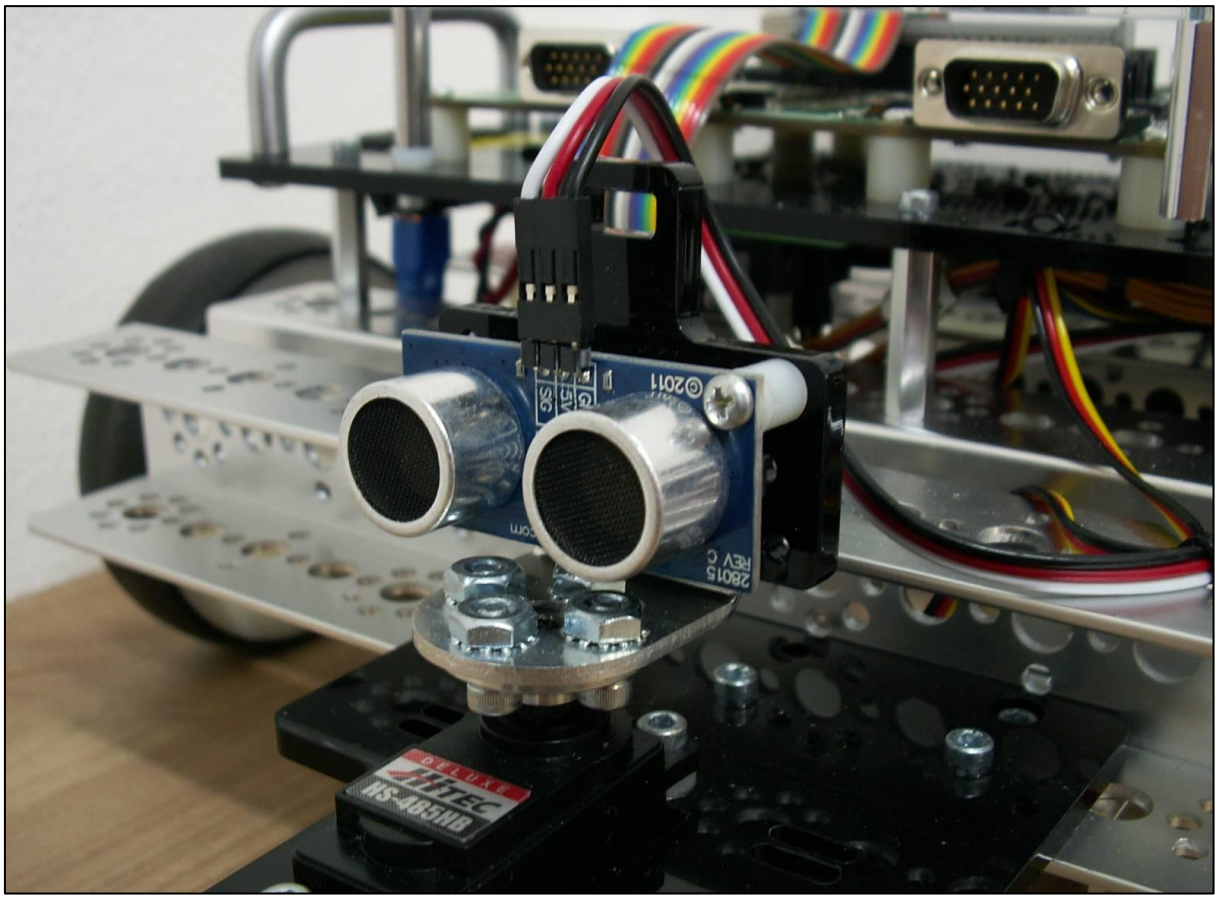
Obr. 4.3 – Základní schéma NI SbRIO-9632 [16]

4.1.2 Parallax PING)))

Parallax PING))) ultrazvukový senzor vydáváním krátkých ultrazvukových dávek a posloucháním ozvěny detekuje objekty před ním. Tyto krátké 40kHz dávky se za standardních laboratorních podmínek pohybují vzduchem přibližně rychlostí 344 m/s (rychlost závisí především na teplotě vzduchu a lze ji spočítat pomocí vzorce 4.1), zasahují objekt a odráží se od něj zpět k senzoru. Na základě prodlevy mezi vydáním zvuku a jeho ozvěnou je pak zjištěna vzdálenost překážky od senzoru. Rozsah tohoto senzoru uvádí výrobce 0,02–3 m [14].

$$c_{vzduch} = 331{,}5 + (0{,}6 \cdot T_{vzduch}), \qquad (4.1)$$

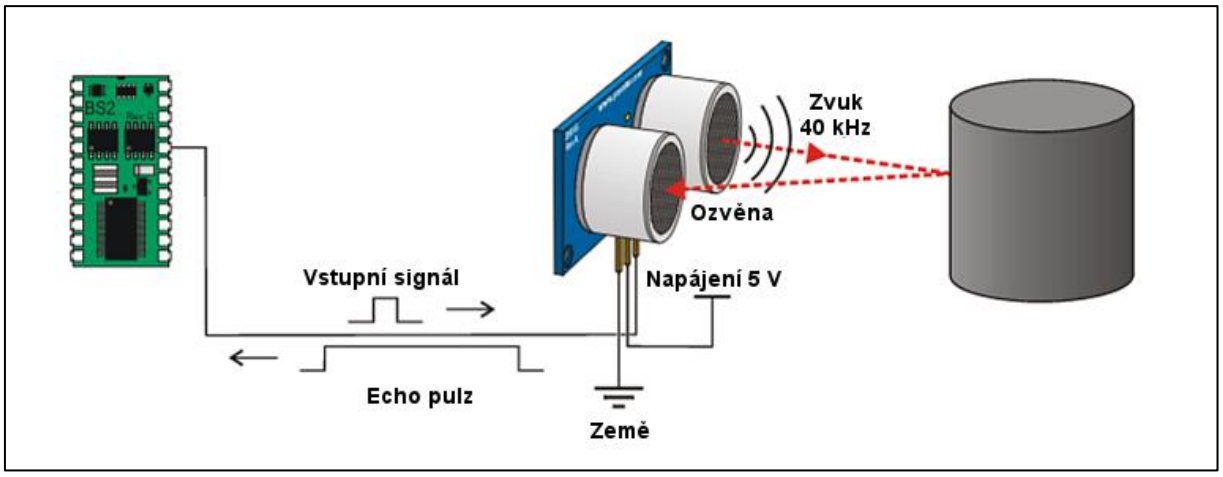
kde c_{vzduch} je rychlost šíření zvuku ve vzduchu v $m \cdot s^{-1}$ a T_{vzduch} je teplota vzduchu v °C.



Obr. 4.4 – Ultrazvukový senzor Parallax PING)))

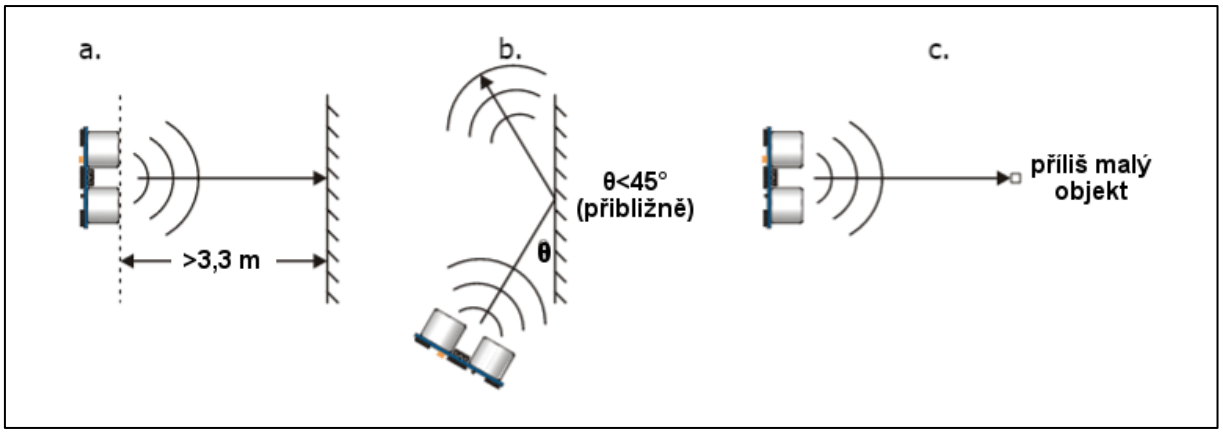
Parallax PING))) se svým okolím, konkrétně s mikrokontrolerem, komunikuje pomocí jediného I/O pinu. Jako vstupní trigger slouží kladný TTL (Transistor-Transistor-Logic) impulz o délce 2 až 5 μs a echo pulzem je kladný TTL impuls o délce 115 μs až 18,5 ms [14].

Poté, co senzor přijme skrze I/O vstupní trigger, vyšle ultrazvukový signál a zároveň začne na I/O vracet echo pulz do chvíle, než zachytí ozvěnu vyslaného ultrazvukového signálu. Na základě délky echo pulzu (a tedy i doby, než se vyslaný zvuk vrátil k senzoru) lze při znalosti nebo předpokladu rychlosti šíření zvuku ve vzduchu dopočítat vzdálenost k překážce, od níž se zvuk odrazil. Schéma tohoto procesu včetně schématu zapojení senzoru je znázorněno na obr. 4.5.



Obr. 4.5 – Schéma fungování a zapojení senzoru Parallax PING))) [17]

Ultrazvukový senzor vzhledem k principu svého fungování nemůže správně měřit vzdálenost ke všem objektům. Aby bylo měření relevantní, je potřeba, aby měřené objekty odrážely ultrazvukový signál zpátky k senzoru. Vzdálenost k některým objektům proto může senzor vyhodnotit nesprávně, nebo ji nezměřit vůbec. Tato skutečnost se v průběhu aplikace mých algoritmů na robota DaNI ukázala jako zásadní. Hlavní tři příklady takovýchto problematických objektů jsou uvedeny na obr. 4.6.



Obr. 4.6 – Příklad tří problematicky detekovatelných objektů

4.1.3 Motory a optické enkodéry Pitsco

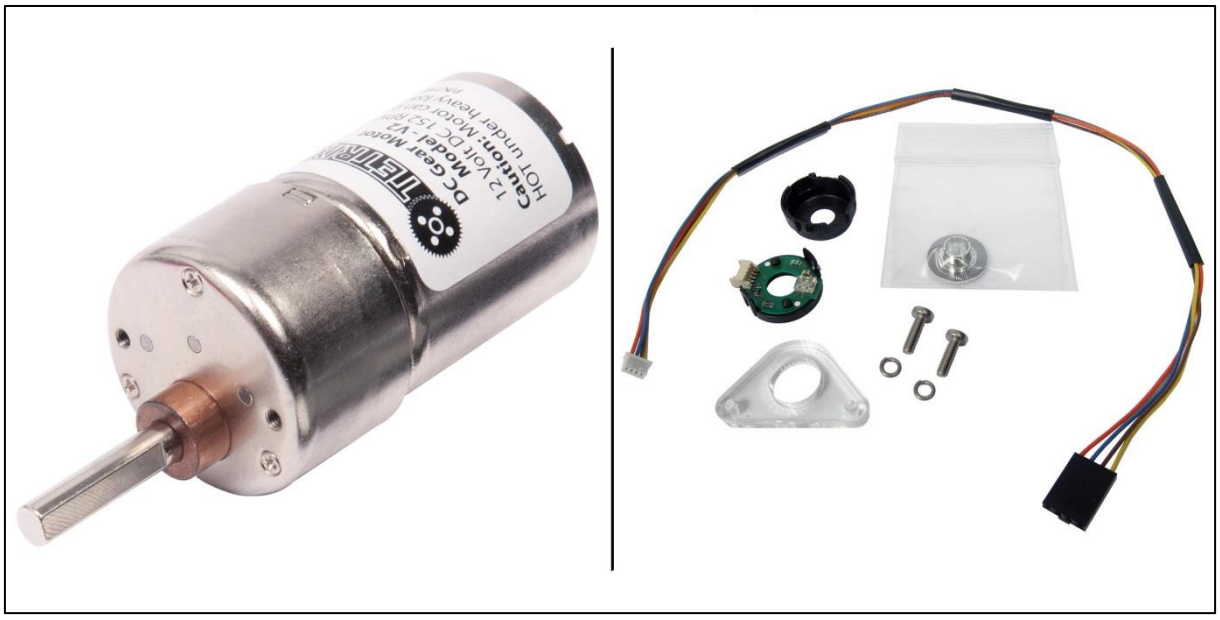
Motory i enkodéry měřící rychlost otáček motoru jsou produkty firmy Pitsco Education. Na robotovi DaNI se jedná konkrétně o TETRIX MAX DC Gear Motor a Pitsco NI Encoder Kit. Parametry těchto součástí jsou následující:

Tabulka 4.1 – Parametry TETRIX MAX DC Gear Motor [14], [18]

Napájecí napětí	6 až 13,8 V (typicky 12 V)
Točivý moment	2 Nm
Maximum otáček za minutu	152 min^{-1}

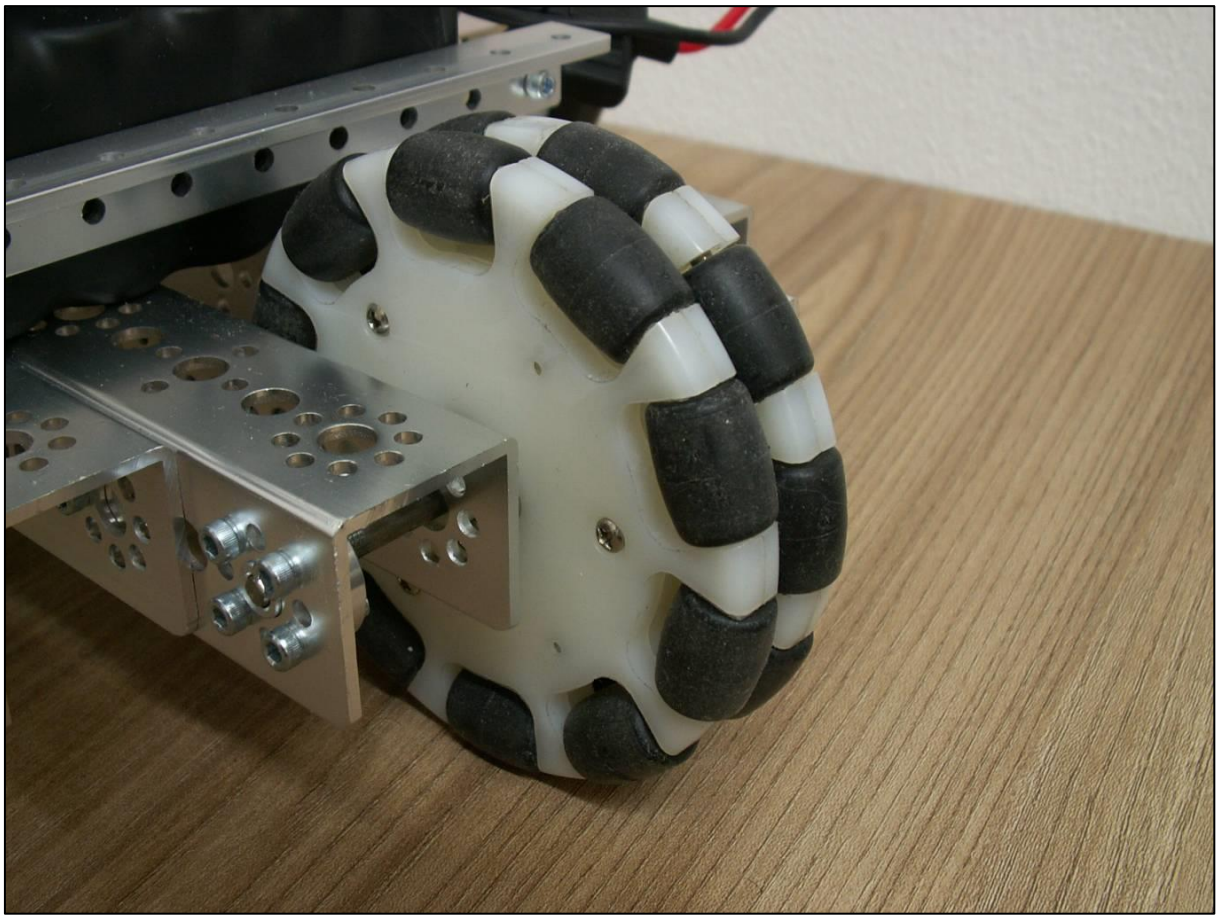
Tabulka 4.2 – Parametry Pitsco NI Encoder Kit [14]

Napájecí napětí	5 V
Rozlišení cyklů na otáčku (CPR)	100
Rozlišení pulsů na otáčku (PPR)	400



Obr. 4.7 – TETRIX MAX DC Gear Motor a Pitsco NI Encoder Kit [19], [20]

Za zmínku stojí i nestandardní elegantní řešení uspořádání podvozku robota. Aby bylo možné robota řídit jednoduše jen pomocí nastavování různých rychlostí otáček levého a pravého motoru, a nebylo nutné přizpůsobovat tomuto řízení ještě natáčení nebo diferencované řízení zadní nápravy, je robot konstruován jako tříkolka, jejíž třetí kolo, umístěné vzadu, je volné všesměrové kolo. To zajišťuje zadní částí robota zachovat v rovině (tj. dimenzi, ve které se robot prakticky pohybuje) všechny tři stupně volnosti, jen s minimálním třením.



Obr. 4.8 – Volné všesměrové kolo

4.2 Doplnění výbavy robota DaNI

Aby bylo možné prezentovat na robotovi všechny aplikace algoritmů, které jsem plánoval já nebo můj kolega Tomáš Dlask, museli jsme standardní výbavu doplnit o další periferní zařízení.

Jak již bylo zmíněno v kapitole 4.1.1, je řídicí deska vybavena ethernetovým kabelovým rozhraním. Právě skutečnost, že je rozhraní jen kabelové, způsobovalo při začátcích naší práce s robotem problémy. Kvůli omezené délce síťového kabelu se mohl robot pohybovat jen ve velmi malém prostoru a zároveň bylo nutné ho neustále hlídat, aby v případě chybně naprogramovaného algoritmu tento prostor neopustil a vytržením kabelu nezpůsobil škody na sobě samém, nebo na vybavení fakultní laboratoře. Právě pro toto nepraktické omezení jsme se rozhodli vybavit robota wifi routerem.

Vzhledem k tomu, že jsme během našich experimentů chtěli vyzkoušet i videodetekci, bylo potřeba vybavit robota dále i zařízením pro zpracování obrazu. Jako nejvhodnější

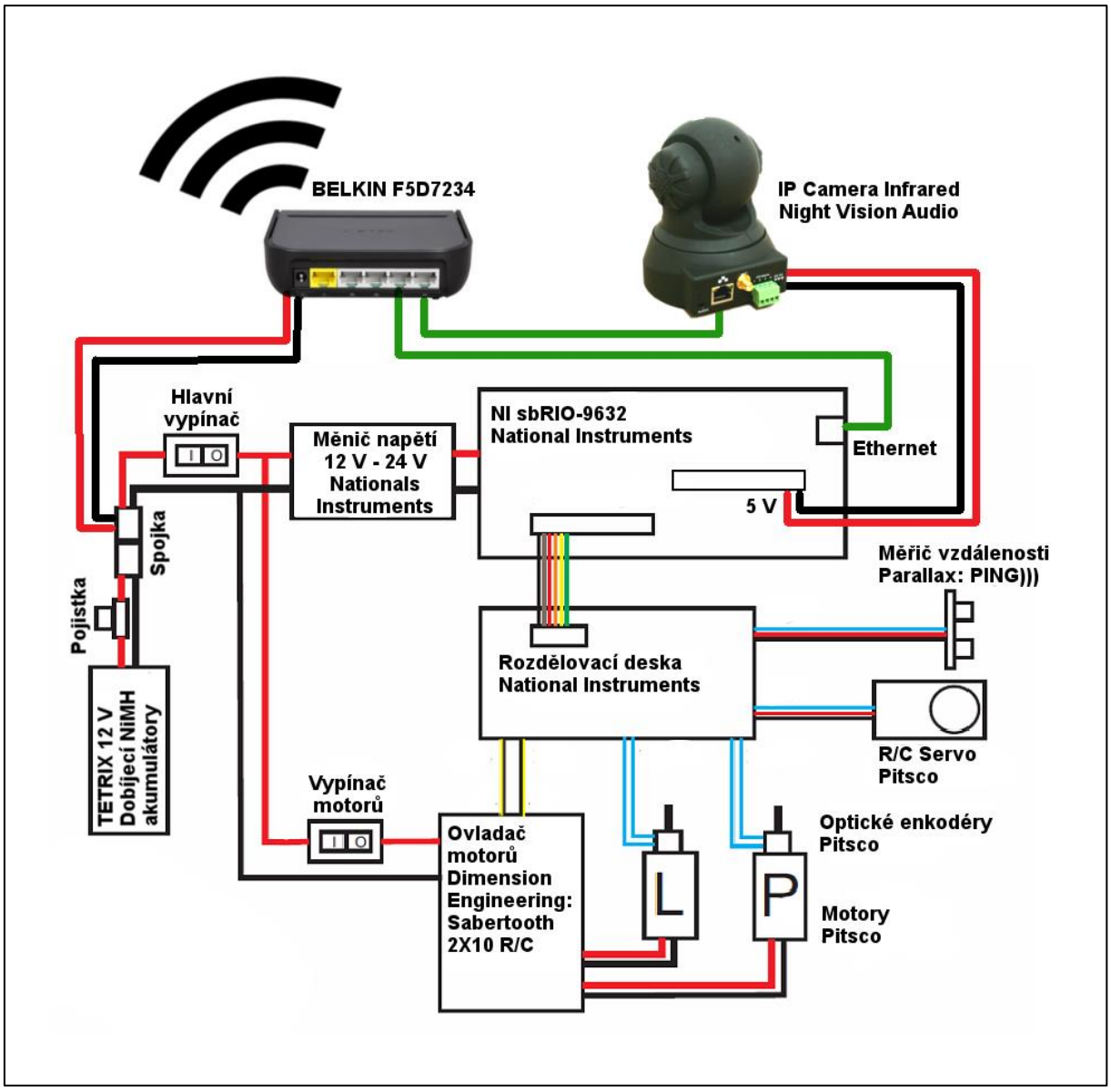
se nám jevilo využití IP kamery nebo webkamery, ale blíže se o procesu rozhodování nad touto věcí zmíním později v příslušné kapitole.

4.2.1 BELKIN F5D7234

S přihlédnutím k tomu, že pro naše účely je dostačující obyčejný router s podporou wifi a alespoň dvěma sloty RJ-45, které jsou v dnešní době nejběžnějšími sloty pro síťovou komunikaci, rozhodli jsme se, abychom neplýtvali penězi, zakoupit nějaký router „z druhé ruky“, který musel splňovat pouze jedno nepříliš standardní kritérium – totiž aby bylo možné ho napájet napětím, které lze na robotovi najít. Přicházelo tak v úvahu napětí 5 V, které lze získat z desky NI sbRIO-9632, nebo napětí 12 V, kterými disponují baterie. Po základním průzkumu trhu jsem zjistil, že běžných routerů, které mají napájecí napětí 5 V, příliš není a 12V routery se většinou pohybují ve vyšší cenové relaci. Ovšem i tak se nám podařilo za 200 Kč získat BELKIN F5D7234 napájený 12 volty.

Obr. 4.9 – BELKIN F5D7234

Protože router je dodávaný s adaptérem, který převádí 230 V AC na 12V DC, bylo potřeba dvoužilový kabel od tohoto adaptéru oddělit a využít z něj pouze konektor, který do routeru pasuje. Abychom příliš (a hlavně trvale) nezasahovali do hardwaru zapůjčeného robota, zapojili jsme kabel do spojky mezi bateriemi a hlavním vypínačem.



Obr. 4.10 – Schéma zapojení periferií na robota DaNI

Jak je znázorněno na obr. 4.10, je do routeru kabelově připojen jak robot DaNI, tak IP kamera, kterou popíši v následující kapitole. Dále router vysílá wifi signál, pomocí něhož je možné připojit počítač, který pak má přístup k obrazovému výstupu z kamery a pomocí LabVIEW robota ovládá.

4.2.2 IP Camera Infrared Night Vision Audio

Je třeba začít hned tím, že jsem si vědom, že specifikace kamery, která je uvedena v nadpisu této kapitoly, je příliš obecná. Je tomu tak ovšem proto, že se jedná o neznačkový výrobek patrně čínské provenience, a přesnějším typovým označením tuto IP kameru označit nejde.

Obr. 4.11 – IP Camera Infrared Night Vision Audio

Kameru jsme na rozdíl od routeru nemuseli nikde shánět a půjčovali jsme si ji pro naši práci ze školní laboratoře. Kamera je napájena 5 V, které lze získat z I/O patice na NI sbRIO-9632. Ovšem i v tomto případě jsme byli vystaveni obdobnému problému jako v případě routeru – jak levně získat správný konektor. Dodávaný adaptér opět na vstupu pracoval s 230 V, a v tomto případě, protože se jednalo o majetek fakulty, nebylo možné kabel přeříznout a napojit přímo na řídicí desku. Nezbylo nám tedy nic jiného, než improvizovat. Při detailním zkoumání napájecího konektoru jsme si všimli, že je podobný konektoru, kterým byly před patnácti lety vybaveny nabíjecí adaptéry na mobilní telefony značky Nokia. Po ověření této hypotézy pomocí změření rozměrů konektoru, především vnějšího průměru 3,5 mm, už nebyl žádný problém nějakou takovouto vyřazenou nabíječku sehnat a kabel přeříznout.

Napájení tedy bylo vyřešeno, ale vyvstal problém s kompatibilitou IP kamery a LabVIEW (tj. programovacím prostředí, k jehož popisu dojde v příslušné kapitole). Aby bylo možné využít v LabVIEW obrazový vstup kamery, je potřeba mít buď kameru, která je od výrobce tohoto programu (National Instruments) přímo podporovaná, nebo webkameru, která se k počítači připojuje pomocí USB. Je jasné, že kamery s přímou podporou od NI se pohybují ve vyšší cenové hladině, kterou jsme si nemohli dovolit, a webkameru s USB nebylo možné, kvůli naší snaze o bezdrátovost, použít. Po několika marných snahách, se mi tento problém podařilo vyřešit pomocí softwaru, který dokáže

obraz z IP kamery vložit do emulované webkamery (přesněji řečeno dokáže využít DirectShow API). Program se příznačně jmenuje IP Camera Adapter 2.0 a jako freeware jsme ho získali z [21].

4.3 LabVIEW

Vývojové prostředí LabVIEW (Laboratory Virtual Instruments Engeneering Workbench – laboratorní pracoviště virtuálních přístrojů) je produktem společnosti National Instruments.

„Prostředí LabVIEW, někdy nazývané též jako G-jazyk (tedy „grafický“ jazyk), je vhodné nejen k programování systémů pro měření a analýzu signálů, řízení a vizualizaci technologických procesů různé složitosti, ale také k programování složitých systémů, jako je třeba robot. S určitou nadsázkou lze říci, že prostředí LabVIEW nemá omezení své použitelnosti.

Hlavním cílem virtuální instrumentace je nahradit dočasně nebo i trvale prostorově, finančně a mnohdy i časově náročné využití technických prostředků (hardware) řešením virtuálním (zdánlivým) za přispění programových prostředků (software) a zejména pak grafickými a vizuálními prostředky a zprostředkovat tak uživateli maximální názornost. Toto řešení umožňuje rychlé navrhování nových aplikací i provádění změn v konfiguraci, což je u realizace skutečnými nástroji za pomoci reálných součástek často velice nákladné nebo přímo nemožné.“ [22]

Virtuální instrumentace se odráží i v označení souborů, se kterými se pracuje. Nazývají se virtuální instrumenty, běžně se v LabVIEW používá zkratka VI, a zároveň takový soubor používá příponu souboru *„.vi“.*

LabVIEW disponuje mnoha zásuvnými moduly, které jsou uzpůsobeny různému využití. Pro mou práci je stěžejní modul LabVIEW Robotics, který již v základu obsahuje některé VI určené pro práci se standardními roboty, mezi které patří i DaNI.

4.3.1 Základní rozdíly oproti Lego Mindstorms

Při základním seznamování se s LabVIEW Robotics jsem měl obtíže s pochopením rozdílů mezi filosofickými přístupy použitými právě v tomto vývojovém prostředí a v prostředí Lego Mindstorms, které jsem používal při práci na předchozím robotovi.

Původně jsem se domníval, že vzhledem k tomu, že pod vývojovým prostředím Lego Mindstorms je podepsána společnost National Instruments, bude LabVIEW Robotics jen sofistikovanější a rozsáhlejší prostředí zproštěné od dětské vizualizace. Což nebyla špatná domněnka, ale zároveň nebyla úplná.

Rozdílem, který byl na první pohled patrný, bylo, že v LabVIEW je možno vytvářet velké množství paralelních procesů, což Mindstorms úplně snadno nelze. V Mindstorms se ze startu každého programu táhnou tři základní procesní vlákna, která sice mohou být dále větvena, ale už to, že se člověk v paralelizaci může pohybovat jen po těchto vláknech, je podstatným rozdílem oproti LabVIEW, kde je paralelním procesem automaticky každý proces, který není závislý na procesu předešlém.

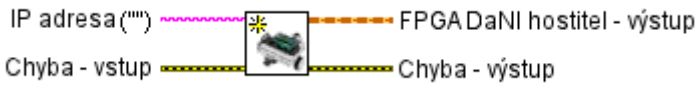
Velmi užitečným rozdílem je u LabVIEW existence čelního panelu (front panel). Jedná se o jakýsi řídicí pult každého programu. Mohou na něm být umístěny kontrolky, indikátory, grafy, tlačítka a jiné řídicí prvky. Díky němu je možné sledovat a měnit vybrané parametry systému za běhu programu. Pro mé následovníky v práci na robotovi je vhodné na tomto místě uvést důležitou, a v LabVIEW vůbec nejpoužívanější, klávesovou zkratku CTRL+E, která přepíná v rámci jednoho programu mezi jeho blokovým schéma a čelním panelem.

Další odlišností, kterou jsem si v začátcích stále nechtěl připustit, je, že LabVIEW neobsahuje žádné základní jednoduché bloky (VI), které by robotovi přikazovaly jízdu vpřed, vzad, otočení vpravo a otočení vlevo. LabVIEW je totiž vybaveno mnohem sofistikovanějšími bloky, které určují rychlost stáčení robota, a jeho rychlost podélnou a příčnou (konstrukční uspořádání robota DaNI samozřejmě příčnou rychlost neumožňuje), o kterých bude řeč v následující kapitole.

V neposlední řadě je rozdíl v tom, jak už jsem naznačoval v kapitole 3.3, že LabVIEW na rozdíl od Mindstorms disponuje breakpointy, které velmi usnadňují práci s testováním programu. Mají jen jednu nevýhodu. Pokud se program zastaví na breakpointu ve chvíli, kdy má robot své elektromotory v pohybu, přestane program pochopitelně svou činnost vykonávat, ale motory setrvávají v předem určeném pohybu (například pohánějí robota stále vpřed). Na tuto skutečnost je třeba stále myslet a v inkriminovaný okamžik, pokud nechceme, aby se robot i nadále pohyboval, než zanalyzujeme situaci po nastalém breakpointu, samostatným vypínačem vypnout motory.

4.3.2 Nejdůležitější bloky LabVIEW Robotics

Initialize Starter Kit 2.0

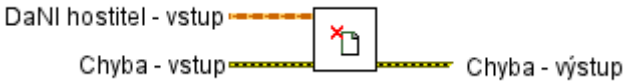


Zahajuje komunikaci s FPGA na robotovi, kterého má ovládat, a vrací referenci k použití čtení a zapisování na FPGA [23]. Tento blok musí být použit na začátku algoritmu a veškeré bloky, které využívají robota, musí být sériově napojeny na jeho výstup.

Initialize Simulated Starter Kit 2.0

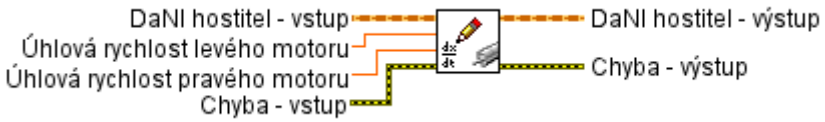
Obdobné jako předchozí blok, ale s tím rozdílem, že se používá pro program spouštěný v simulačním prostředí.

Close Starter Kit



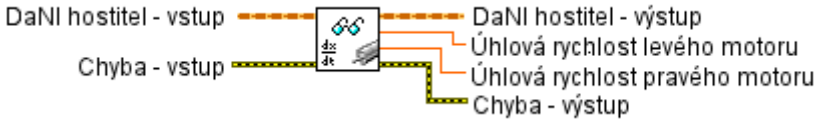
Uzavírá komunikaci s FPGA na reálném nebo simulovaném robotovi. Když komunikace skončí, motory, ultrazvukový senzor a otáčení ultrazvukovým senzorem přestávají být aktivní. [23]

Write DC Motor Velocity Setpoints



Zadává hodnoty rychlosti řídicím motorům robota DaNI. Levý a pravý motor jsou definovány podle své polohy na robotovi při pohledu zezadu. Maximální rychlost motorů je 15,7 rad/s. [23]

Read DC Motor Velocities

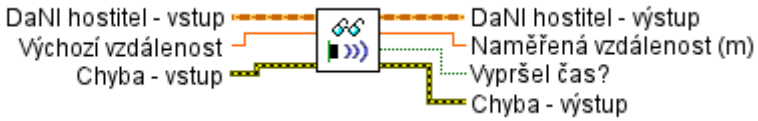


Vrací hodnoty rychlosti levého a pravého motoru na robotovi DaNI podle směru hodinových ručiček. Levý a pravý motor jsou definovány stejně jako u bloku výše. [23]

Write Sensor Servo Angle

Zadává úhel otočení servo motorku na kterém je umístěn ultrazvukový senzor. Pakliže ultrazvukový senzor není zarovnán se servo motorkem, musí se dát pozor na to, že se tento úhel může od úhlu natočení senzoru lišit. [23]

Read PING))) Sensor Distance



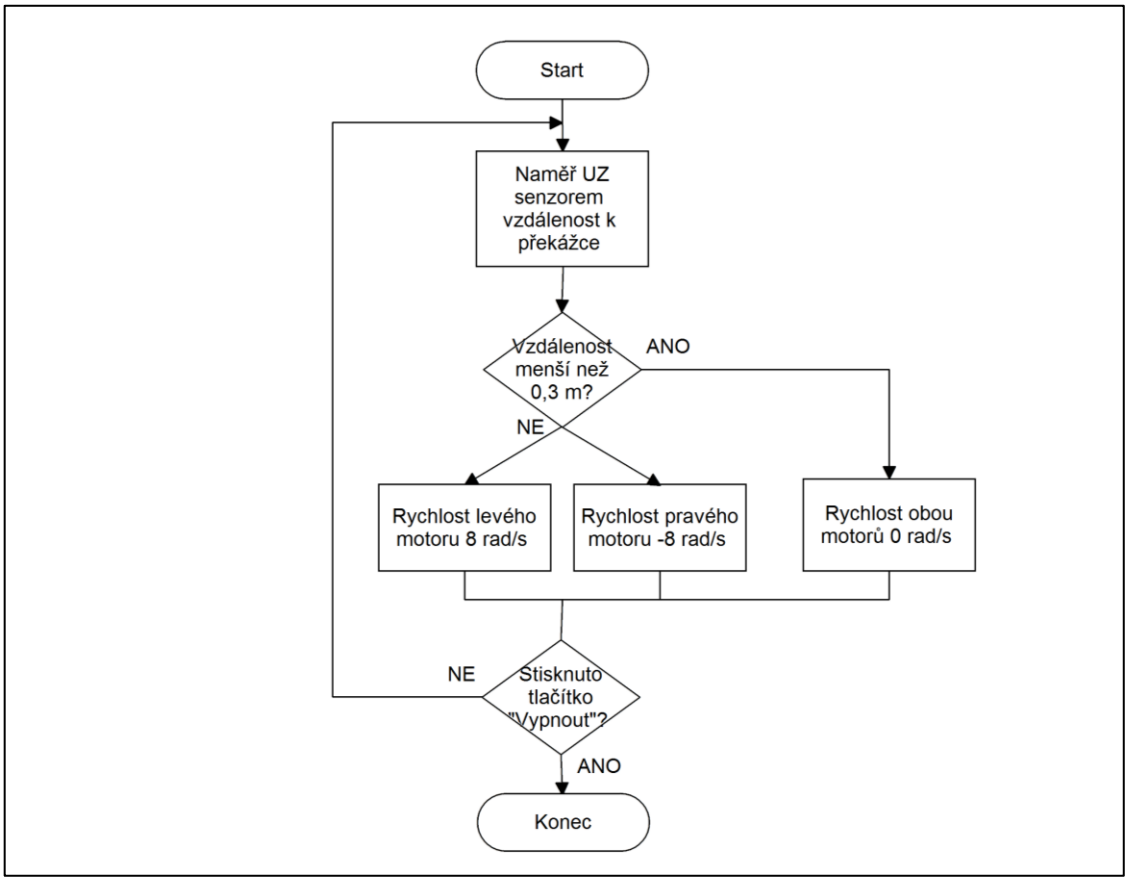
Čte a vrací hodnotu mezi ultrazvukovým senzorem a nejbližší překážkou, kterou senzor detekuje. Maximální vzdálenost je 3 m. Pokud senzor nedetekuje žádnou překážku, má boolean „Vypršel čas?“ hodnotu TRUE. [23]

5 Aplikace algoritmů na robota DaNI

Oproti programování robota Lego Mindstorms jsem se rozhodl pro robota DaNI připravit hned několik různých programů. Toto rozhodnutí jsem udělal z toho důvodu, že programování v LabVIEW je složitější, a proto není vůbec snadné začít pracovat hned na složitém algoritmu. Výhodnější je pochopit nejprve jednotlivé podčásti komplexního problému. Zvlášť se naučit ovládat motory, využívat senzory, zpracovávat obraz, apod. Navíc robot DaNI (po dovybavení kamerou a wifi) v kombinaci s možnostmi LabVIEW umožňuje více možností různých využití než Lego Mindstorms.

5.1 Zastavení před překážkou

Jako nejjednodušší možný algoritmus, na kterém bych vyzkoušel základní ovládání motorů a získávání údajů ze senzoru, jsem si vybral takový algoritmus, který zajistí zastavení jedoucího robota ve chvíli, kdy jeho ultrazvukový senzor zaznamená předmět v určené vzdálenosti. V mém případě jsem zvolil vzdálenost ≤ 0,3 m.



Obr. 5.1 – Zjednodušené schéma algoritmu zastavení před překážkou

Algoritmus je ve své podstatě cyklus, ve kterém se nejprve zjistí vzdálenost, kterou aktuálně naměřil ultrazvukový senzor, a poté se na základě vyhodnocení vzdálenosti

robot buď pohybuje konstantní rychlostí vpřed, nebo zastaví. Následně se ověří, zda uživatel na čelním panelu stiskl tlačítko pro ukončení programu, nebo se má cyklus znovu opakovat. Prodlevu mezi opakováním jsem zvolil 20 ms, což je doba, která je dostatečná k tomu, aby robot reagoval včas (z pohledu člověka téměř okamžitě) a zároveň získal bohatou časovou rezervu pro ultrazvukové měření.

Po prohlédnutí obr. 5.1 se nezasvěcený čtenář zajisté podiví, proč se pro jízdu vpřed nastavuje pravému motoru záporná rychlost. I já jsem se při realizaci tohoto programu zprvu divil, protože při nastavení obou rychlostí kladných, se robot otáčel na místě, ale vysvětlení je jednoduché. Tím, že jsou motory pro levé a pravé kolo na robotovi uloženy opačně – zrcadlově, je pro jízdu jedním směrem nutné, aby se pravý motor otáčel na opačnou stranu než levý.

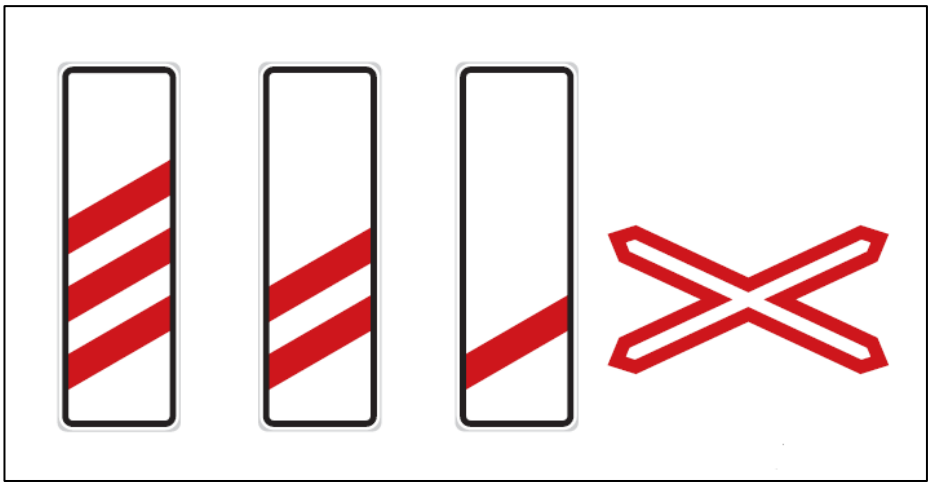
Vzhledem k jednoduchosti tohoto programu se při jeho testování nevyskytly žádné podstatné problémy a nic mi nebránilo v tom, zaměřit své úsilí na další algoritmus.

5.2 Vizuální rozpoznání dopravních informací

Protože by zpracování algoritmu, který by upozorňoval na veškeré dopravní značení, bylo příliš náročné (hlavně databázově a výpočetně), rozhodl jsem se pro rozpoznávání značek, spojených s železničním přejezdem. K tomuto rozhodnutí jsem dospěl po konzultaci s kolegou Tomášem Dlaskem, který právě na problematice železničních přejezdů staví svou diplomovou práci a vytvoření takového programu by uvítal.

Po prodiskutování této problematiky jsme se rozhodli, že pro upozornění na železniční přejezd by měl být algoritmus schopen rozpoznat a upozornit konkrétně na následující značení:

- "Návěstní deska (240 m)" (A 31a)
- "Návěstní deska (160 m)" (A 31b)
- "Návěstní deska (80 m)" (A 31c)
- „Výstražný kříž pro železniční přejezd jednokolejný“ (A 32a)

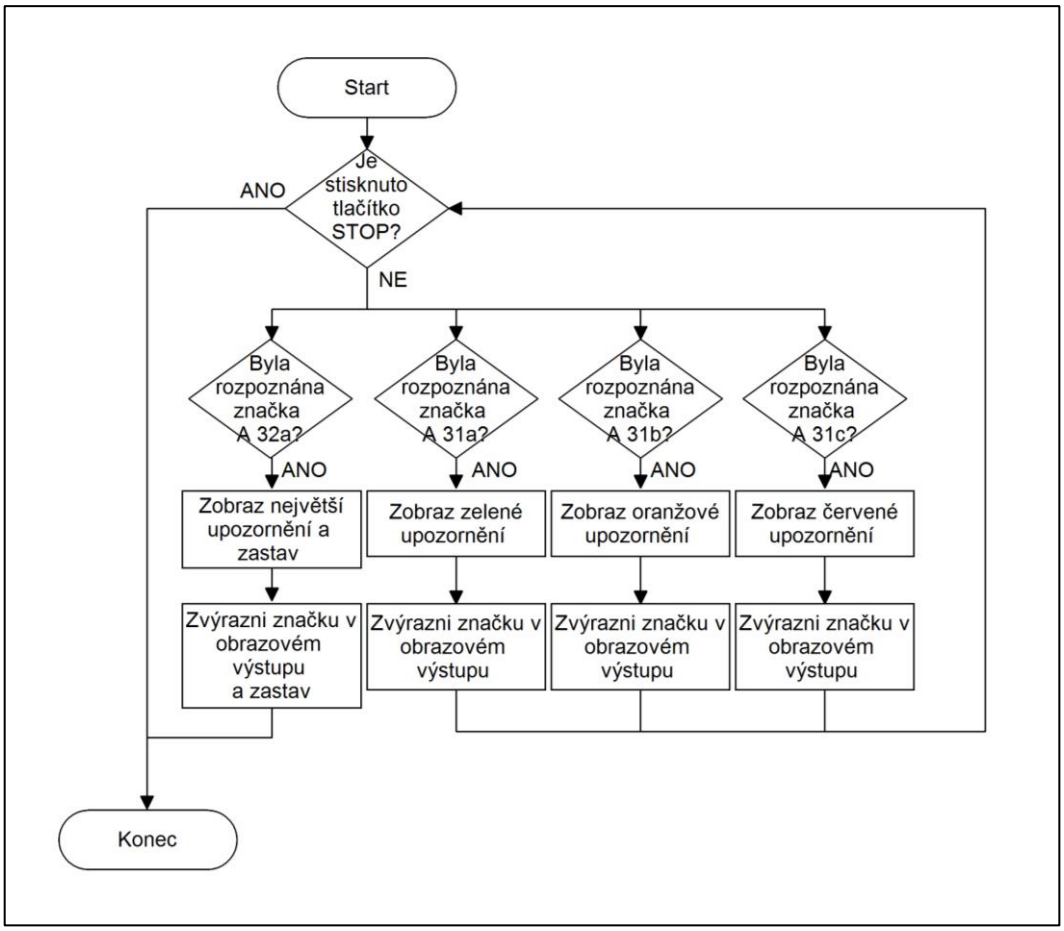


Obr. 5.2 – Dopravní značení A 31a, A 31b, A 31c, A 32a [24]

Upozornění na značení A 31 by pak mělo řidiče pouze informovat o blížícím se přejezdu a upozornění na A 32, neboli ondřejský kříž, by mělo být maximálně varující a, ačkoliv je to v praxi z důvodu bezpečnosti a práva nemožné, v našem modelovém případě by měl DaNI před tímto značením i zastavit.

Pro návrh algoritmu se nabízely dvě cesty. Buď proces rozpoznávání pojmout sériově, nebo paralelně. Sériovým procesem rozumím takový algoritmus, který by během jízdy robota vyhledával značku A 31a, a až teprve ve chvíli, kdy by bylo toto značení rozpoznáno, začala by se vyhledávat značka A 31b, pak A 31c a A32a. Tento přístup by oproti paralelnímu vyhledávání, kdy se všechny značky vyhledávají naráz, velmi výrazně snížil výpočtovou náročnost. Program by nemusel během každého projetí cyklu zpracovávat obraz čtyřikrát, ale jen jednou. Ovšem vzhledem k tomu, že by v případě nerozpoznání jedné značky z celé série byl proces dále nepoužitelný, jsme od této varianty ustoupili. Značka by mohla být nerozpoznána nejenom tím, že by byl špatně vyhodnocen obraz z kamery, nebo tím, že by obraz z kamery byl krátkodobě znehodnocen například oslněním, ale značka by mohla u silnice z mnoha důvodů i reálně chybět.

Zvolili jsme tedy proces paralelní, jehož nevýhodou je sice náročnost, ale ta je převážena větší odolností proti chybám.



Obr. 5.3 – Zjednodušené schéma algoritmu rozpoznávání dopravních značek

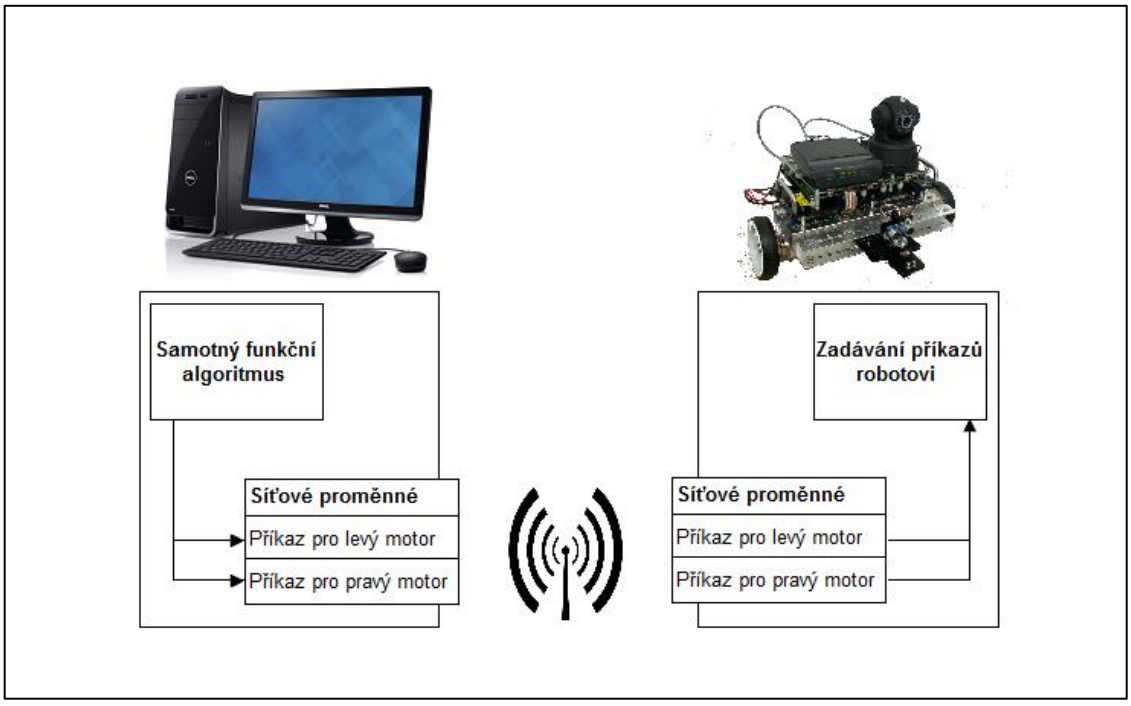
Pro napodobení reálného využití takovéto aplikace jsme se rozhodli vytvořit na čelním panelu jakousi „přístrojovou desku“ s kontrolkami a prvky pro řízení robota – vertikální posuvník pro řízení podélné rychlosti a horizontální posuvník pro řízení rychlosti stáčení. Zároveň jsme čelní panel nechali zobrazovat výstup z kamery umístěné na robotovi, aby člověk, který bude robota řídit, měl obdobný obrazový vjem, jako kdyby v robotovi seděl.

Pro zpracování obrazového výstupu z IP kamery, potažmo emulátoru TWAIN webkamery, je použit blok Vision Acquisition, který zdrojové video převede na formát použitelný pro další zpracování. K němu je pak využit blok Vision Assistant. Ten dokáže vytvářet, editovat a spouštět vnitřní aplikace modulu Vision.

Předchozí algoritmus (5.1 Zastavení před překážkou) byl před svým spuštěním, kompilován, celý přenesen na výpočtovou jednotku robota a tam probíhaly veškeré potřebné výpočty, přičemž na počítači byl zobrazen pouze čelní panel. Protože ale v algoritmu pro rozpoznávání dopravního značení potřebujeme zpracovávat obrazový

vstup pomocí modulu Vision, který nejde spustit na výpočetní jednotce robota, museli jsme přistoupit k rozdělení této aplikace na dvě části. Jednu, která bude spuštěna v robotovi, a druhou, která bude spuštěna v počítači, a vzájemně mezi sebou budou komunikovat síťovými proměnnými.

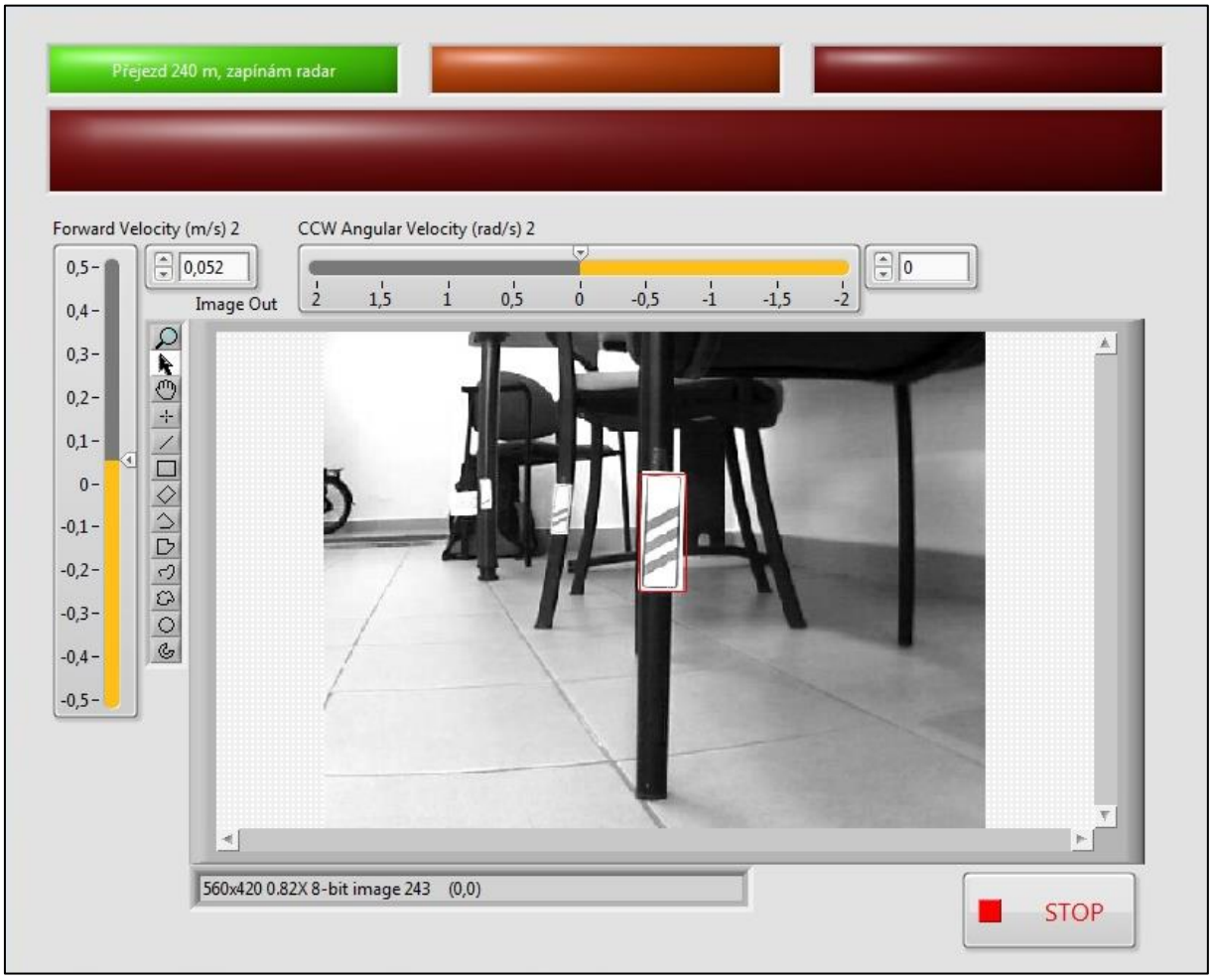
Po několika pokusech o různá rozdělení algoritmu jsme se nakonec rozhodli, že nejlepší bude celý funkční algoritmus spustit v počítači, a v robotovi necháme spuštěný pouze program pro základní kontrolu motorů. Tento program v robotovi bude číst síťové proměnné, které budou vyjadřovat požadovanou rychlost otáčení motorů, a toto číslo bude motorům rovnou zadávat.



Obr. 5.4 – Schéma rozdělení programu

Pro rozpoznávání dopravního značení uvnitř bloku Vision Assistant obraz nejprve převedeme do odstínů šedi. K tomu využíváme převod přes kanál zelené barvy, neboť tím jsme na černo-bílo-červených dopravních značkách získali nejlepší kontrast. Ve chvíli, kdy máme obraz již černobílý a dostatečně kontrastní, přichází ke slovu funkce nazvaná Pattern Matching, jež v obrazu vyhledává předem nadefinovaný vzor. Jako vzor jsme nepoužili čistý, dalo by se říci laboratorní, vzor dopravní značky jako na obr. 5.2, ale námi vytvořené papírové modely těchto značek nasnímané přes IP kameru. Chtěli jsme totiž, aby hledaný vzor vypadal stejně, jako bude značka vypadat při nasnímání stejnou kamerou. Tyto vzory jsme definovali pro všechny výše zmíněné značky.

Jakmile je obraz takto zpracovaný, blok Vision Assistant nám vrátí následující data: počet nalezených obrazových vzorů jednotlivých druhů, souřadnice nalezených vzorů a zpracovaný obraz. Pokud je v obraze nalezena jedna nebo více značek jednoho druhu, vyhodnotí systém situaci jako pozitivní rozpoznání dopravního značení a příslušně zareaguje (podle schématu na obr. 5.3).



Obr. 5.5 – Čelní panel programu pro rozpoznávání dopravních informací

Díky skutečnosti, že funkce Pattern Matching nerozpoznává v obrazu značky v případě, že se jejich velikost výrazněji liší od velikosti vzoru, nestane se, aby program upozornil naráz na všechny značky. Ani v případě, že jsou v obraze všechny viditelné (jako na obr. 5.5). Upozorňuje vždy jen na značení, které je v určité „správné“ vzdálenosti před kamerou, a tím má v obraze „správnou“ velikost.

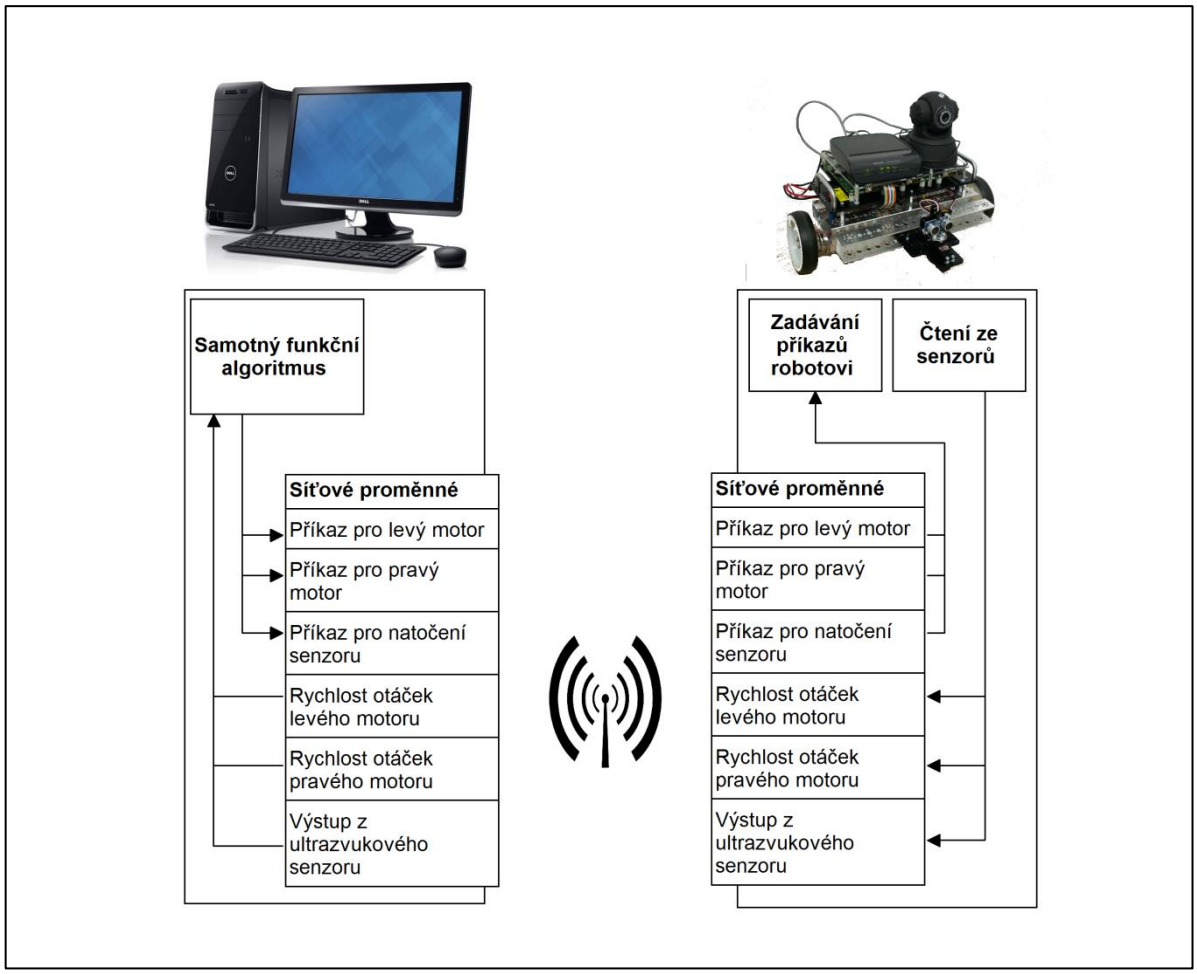
Při testování vyšlo najevo, že program má občas problémy s rozpoznáním značek A 31a, A 31b a A 31c, které mezi sebou zaměňuje. Je to způsobeno tím, že tyto značky se od sebe z celoplošného hlediska liší jen minimálně, a při nevhodném nastavení prahu

tolerance je pak rozdíl v podobě jednoho červeného pruhu tolerován. Na druhou stranu ale nemůžeme kvůli námi používanému předzpracování obrazu nastavit práh tolerance příliš přísně. Tím, že používáme IP kameru, z níž dostáváme obraz již komprimovaný do formátu MJPEG, je výstup již částečně znehodnocen. Pro případné zlepšení výsledků bych navrhoval využití kvalitnější kamery, která obraz nijak nekomprimuje. Její pořízení ovšem vyžaduje nemalé finanční náklady.

5.3 Adaptivní tempomat

Adaptivní tempomat (ACC – Adaptive Cruise Control) je označení pro elektronický asistenční systém, který *„udržuje nastavenou rychlost v závislosti na odstupu vpředu jedoucího vozu. V případě potřeby automobil dokáže samočinně přibrzdit nebo naopak zrychlit podle aktuální dopravní situace.“* [25]

Na základě předchozího algoritmu a jeho rozdělení na dva spolu komunikující programy jsem vytvořil další algoritmus, který právě adaptivní tempomat napodobuje. Tu část programu, která je zpracovávána v robotovi jsem musel pro tyto účely upravit tak, že jsem k ovládání motorů přidal i ovládání ultrazvukového senzoru a čtení hodnot, které vrací. Přestože by to pro fungování právě tohoto algoritmu stačilo, rozhodl jsem se tuto část programu rozšířit ještě dále, a vytvořit z ní jakousi obecnou kdykoliv využitelnou platformu, která je prakticky rozhraním mezi síťovými proměnnými a interaktivním ovládáním robota.



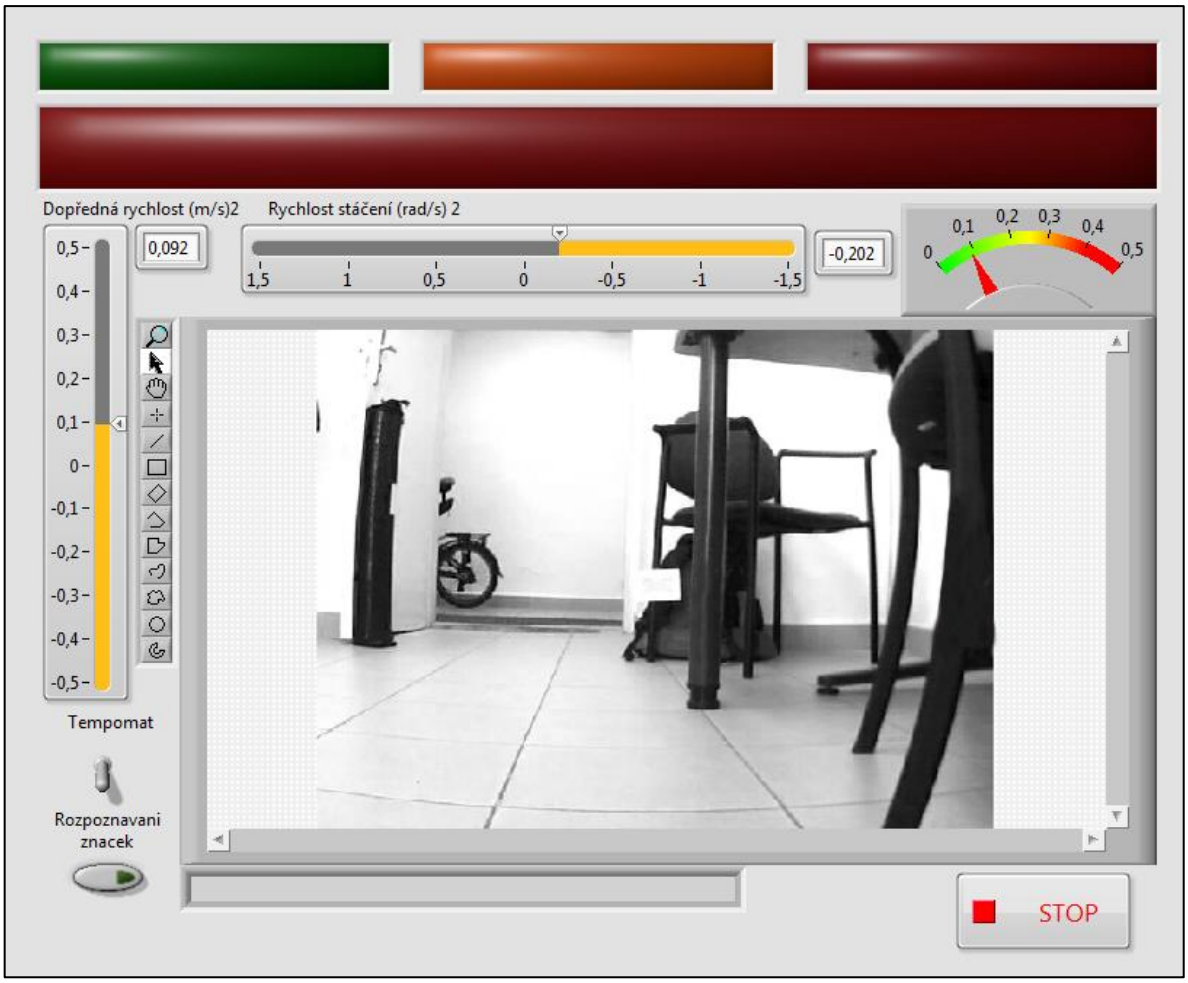
Obr. 5.6 – Schéma úplného rozdělení programu

Vytvořil jsem tedy program, který se na robotovi automaticky spustí při každém zapnutí robota. Aby bylo jasné, že již neprobíhá bootování, ale systém už je plně spuštěn a očekává další instrukce, provede robot hned na začátku spuštění programu pohyb ultrazvukovým senzorem doleva a doprava (jako kdyby se rozhlédl) a dále už jen přijímá instrukce pro otáčení motory a ultrazvukovým senzorem a vysílá do sítě údaje získané z ultrazvukového senzoru a enkodérů otáček na motorech.

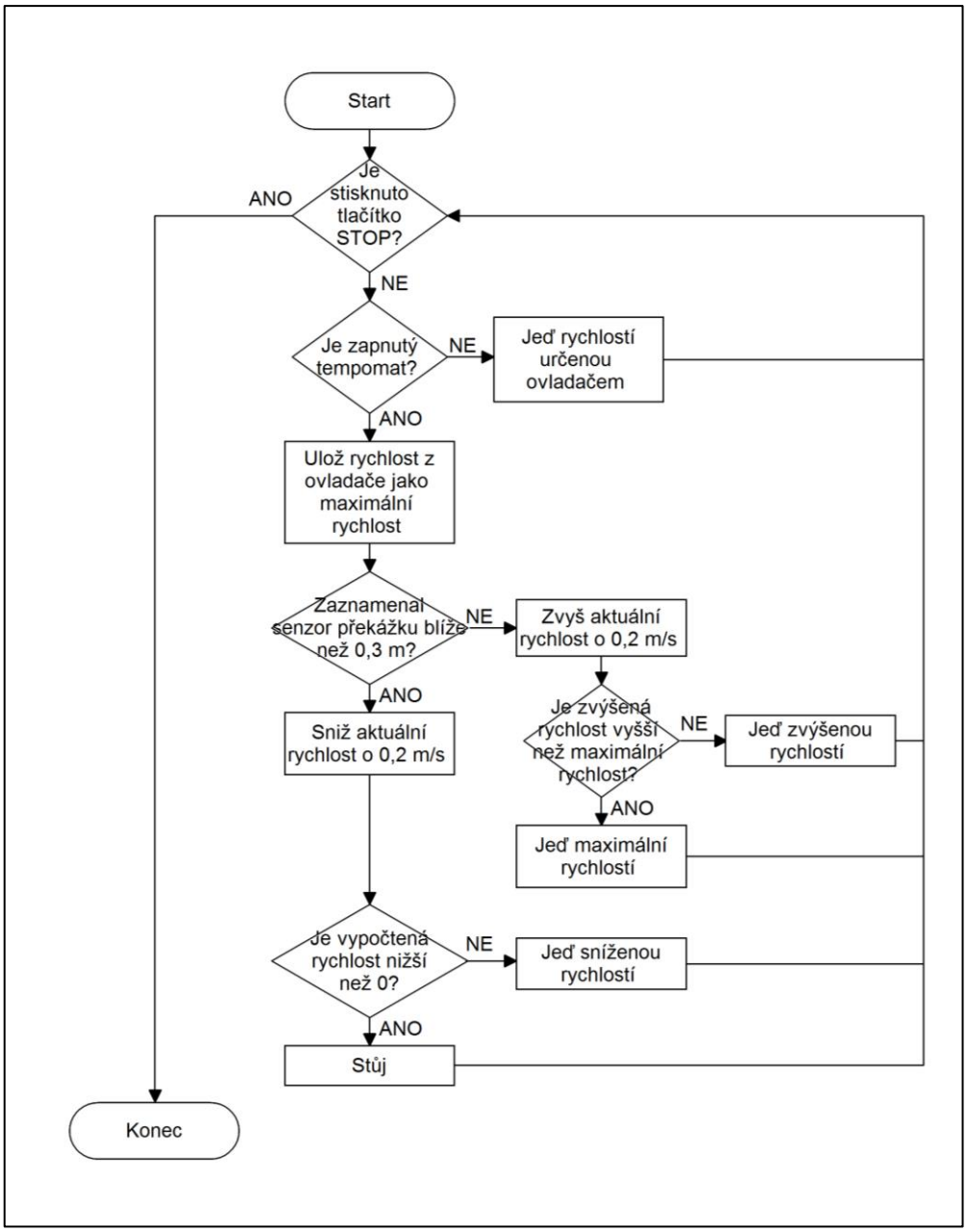
Vzhledem k tomu, že při mé aplikaci programu na robota DaNI lze při jeho rychlosti a hmotnosti zanedbat délku brzdné dráhy, a tedy ji můžeme považovat za nulovou, mohu si dovolit můj algoritmus vytvořit tak, aby na rozdíl od nejběžnějších ACC nejenom samočinně přibrždˊoval, ale v případě nutnosti i úplně zastavil.

Pro řízení jsem zvolil následující strategii: Uživatel skrze čelní panel ovládá rychlost robota a směr, kterým má jet, podobně jako když v automobilu ovládá plynový pedál a otáčí volantem. Přitom má možnost kdykoliv stisknout tlačítko, jímž aktivuje tempomat, který poté udržuje poslední zadanou rychlost až do chvíle, kdy uživatel tempomat vypne

– tj. znovu stiskne tlačítko (protože se při manuálním řízení rychlost ovládá posuvníkem, nesnažím se napodobit vypnutí tempomatu tak, jak je použito v praxi – sešlápnutím pedálu brzdy nebo plynu), nebo do chvíle, kdy se ve vzdálenosti ≤ 30 cm před robotem objeví překážka. V tu chvíli robot přestane udržovat předem zadanou rychlost a určuje rychlost takovou, aby se pokud možno držel od objektu před ním dále, než je vzdálenost 30 cm. V praxi to pak znamená, že pokud se objekt před ním pohybuje v podélném směru pomaleji, než je na tempomatu nastavená rychlost, robot kopíruje jeho rychlost. A to v sestupném směru až do rychlosti 0 m/s. Couvání v mém algoritmu neuvažuji, protože by v reálné praxi bylo jeho použití z bezpečnostního hlediska značně komplikované.



Obr. 5.7 – Čelní panel programu pro adaptivní tempomat



Obr. 5.8 – Zjednodušené schéma algoritmu adaptivního tempomatu

Pro usnadnění řízení, a hlavně možnost řízení robota bez přímého vizuálního kontaktu, jsem v čelním panelu ponechal obrazový výstup z IP kamery. Tento obrazový výstup nemá na fungování algoritmu žádný vliv a slouží skutečně pouze ke zvýšení komfortu ovládání.

5.4 Aktivní vyhnutí se překážce při jízdě k cíli

Nejsložitějším úkolem, pro který jsem robota naprogramoval, je autonomní jízda na předem zadané souřadnice a aktivní vyhýbaní se překážkám, které mu stojí v cestě. Pro tuto funkci jsem programoval už robota Lego Mindstorms a algoritmus popisuji

v kapitole 3.2. Na robota DaNI jsem se pokusil aplikovat algoritmus, který by co možná nejvíce odpovídal algoritmu již dříve použitému. To aby bylo možné porovnat chování obou robotů. I přes to, že bylo mým zájmem mít algoritmy stejné, nemohl jsem se v průběhu programování vyhnout určitým změnám – např. otáčení ultrazvukovým senzorem, couvání v případě potřeby, apod.

Stejně jako u Lego Mindstorms jsem měl v plánu využít bloky pro základní pohyby: blok pro jízdu vpřed, blok pro otočení vpravo, pro otočení vlevo, pro couvání a pro zastavení. Ovšem veškeré bloky implicitně obsažené v modulu LabVIEW Robotics umožňují robotovi zadávat pouze rychlost jeho pohybu, nikoliv rozsah. Není tedy možné zadat robotovi, aby se otočil například o 90° doleva. První úkol, před kterým jsem tedy stál, bylo vytvoření si vlastních bloků (subVI), které budou při základních pohybech zohledňovat nejenom rychlost, ale i rozsah.

Proto bylo potřeba experimentálně zjistit vztah mezi dobou pohybu robota danou rychlostí a změnou jeho polohy během této doby. Určil jsem si, že robot bude při všech pohybech otáčet motory konstantní rychlostí ± 8 rad/s. A pohyb robota jsem se rozhodl složit z pohybů, které se budou cyklicky opakovat po 20 ms. Bude-li tedy potřeba, aby se robot pohyboval 1 sekundu, musí se cyklus s daným pohybem provést 50×.

Aby bylo ale možné jako vstupní hodnotu do bloku používat přímo požadovanou vzdálenost, o kterou má robot popojet, nebo úhel, o který se má otočit, bylo potřeba vytvořit ještě přepočtový koeficient.

$$k = \frac{t}{\alpha \cdot p} \tag{5.1}$$

kde: t..... potřebný čas pro uskutečňování pohybu
α..... požadovaný úhel otočení robota (případně délka popojetí)
k..... přepočtový koeficient
p..... perioda pro opakování cyklu pohybu (v mém případě 0,02 s)

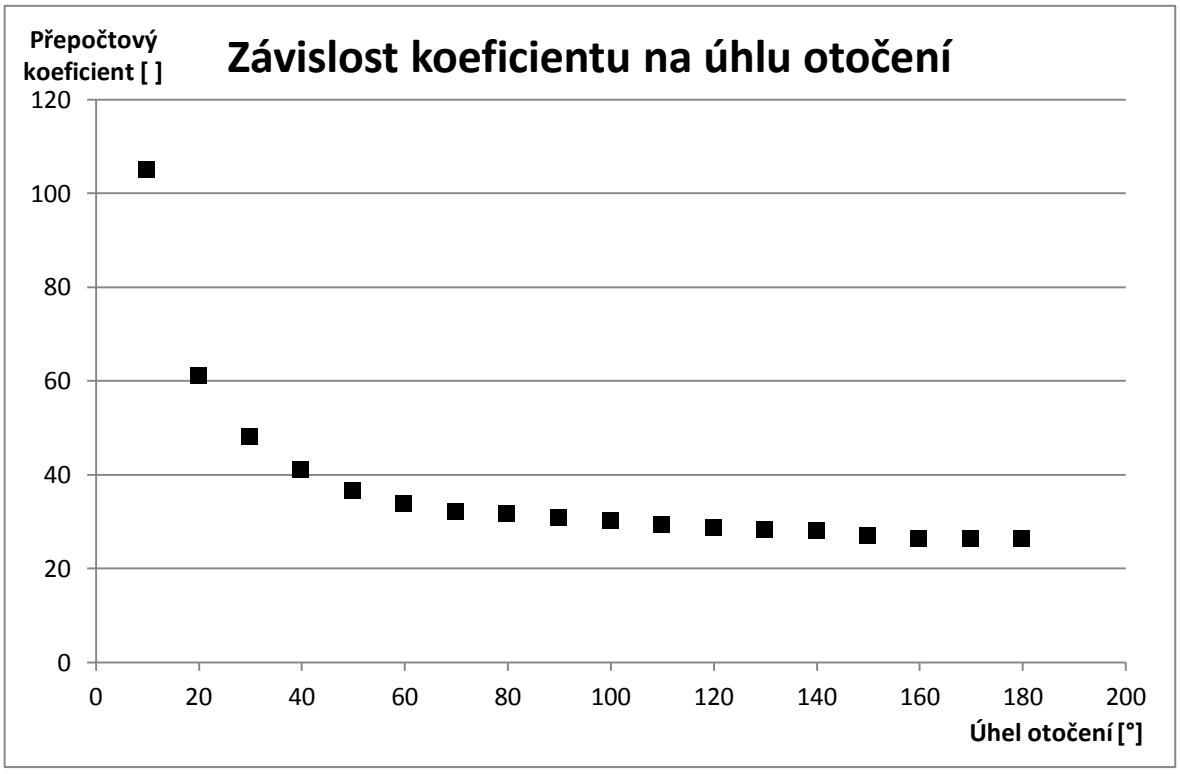
Pak už nezbývalo nic jiného, než při zadané rychlosti 8 rad/s a zadaném požadovaném rozsahu pohybu zkoušet, jaký přepočtový koeficient zajistí, aby hodnota potřebného času pro uskutečnění pohybu odpovídala správné hodnotě.

Během experimentálního měření se mi potvrdil předpoklad, že přepočtový koeficient nebude při různých rozsazích pohybu konstantní, ani lineární (viz tabulka 5.1).

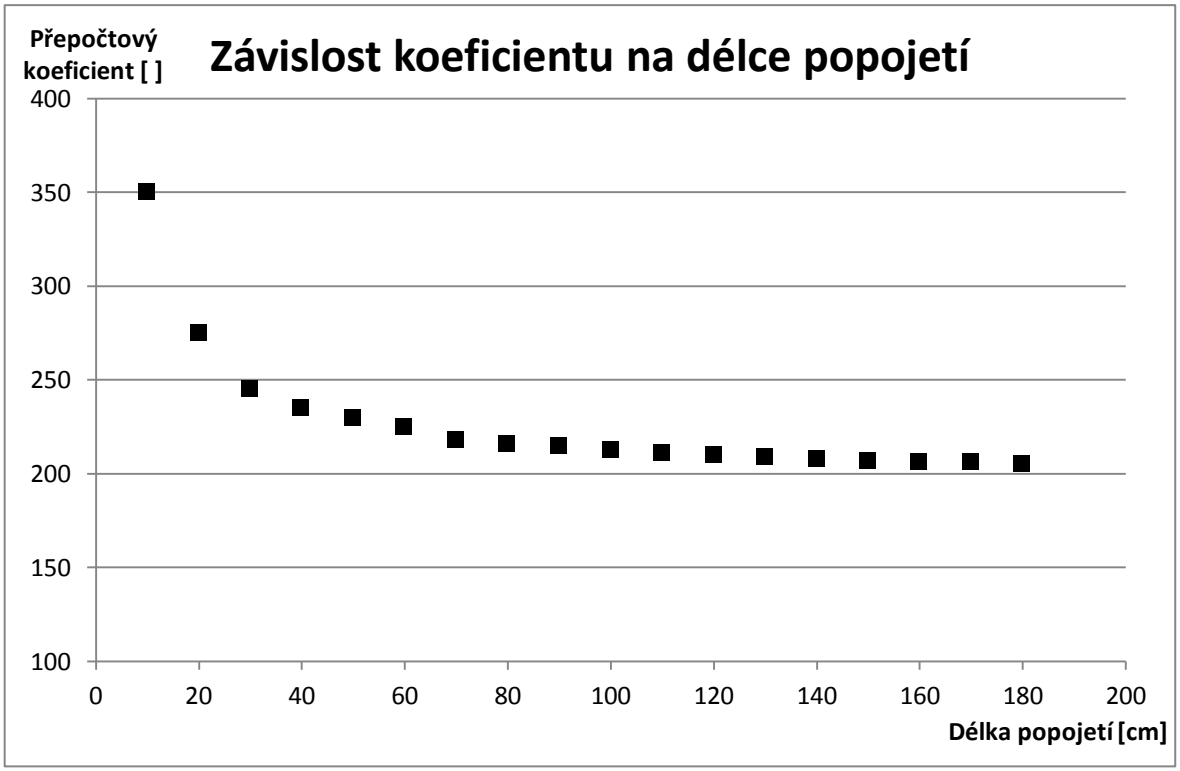
Tabulka 5.1 – Experimentálně zjištěné přepočtové koeficienty

Požadovaný úhel otočení	Přepočtový koeficient
10	105,0
20	61,0
30	48,0
40	41,0
50	36,5
60	33,8
70	32,0
80	31,5
90	30,8
100	30,0
110	29,3
120	28,5
130	28,3
140	28,0
150	27,0
160	26,3
170	26,3
180	26,3
⋮	
360	24,5

Požadovaná délka popojetí	Přepočtový koeficient
10	350
20	275
30	245
40	235
50	230
60	225
70	218
80	216
90	215
100	213
110	211
120	210
130	209
140	208
150	207
160	206
170	206
180	205
⋮	
360	202



Obr. 5.9 – Závislost přepočtového koeficientu na úhlu otočení



Obr. 5.10 – Závislost přepočtového koeficientu na délce popojetí

Získané hodnoty jsem pro ilustraci znázornil v grafech na obr. 5.9 a obr. 5.10. Z nich je pak patrné, že pokud chceme tyto body proložit křivkou a provést regresi, což jsem prováděl, zjistíme, že křivka se poměrně nepravidelně vlní. Pokud chci mít výsledný předpis funkce dostatečně přesný, musel bych získané body prokládat křivkou s velmi složitým předpisem, a proto jsem se rozhodl, že efektivnější bude, jednotlivé intervaly mezi naměřenými hodnotami proložit přímkou, a získat tak pro každý interval hodnot vlastní lineární předpis ve tvaru $k = a \cdot \alpha + b$.

Tabulka 5.2 – Parametry přímek pro jednotlivé intervaly

Otáčení		
Interval [°]	**a**	**b**
<10,20)	-4,400	149,00
<20,30)	-1,300	87,00
<30,40)	-0,700	69,00
<40,50)	-0,450	59,00
<50,60)	-0,275	50,25
<60,70)	-0,175	44,25
<70,80)	-0,050	35,50
<80,90)	-0,075	37,50
<90,100)	-0,075	37,50
<100,110)	-0,075	37,50
<110,120)	-0,075	37,50
<120,130)	-0,025	31,50
<130,140)	-0,025	31,50
<140,150)	-0,100	42,00
<150,160)	-0,075	38,25
<160,170)	0,000	26,25
<170,180)	0,000	26,25
<180,360)	-0,010	28,00

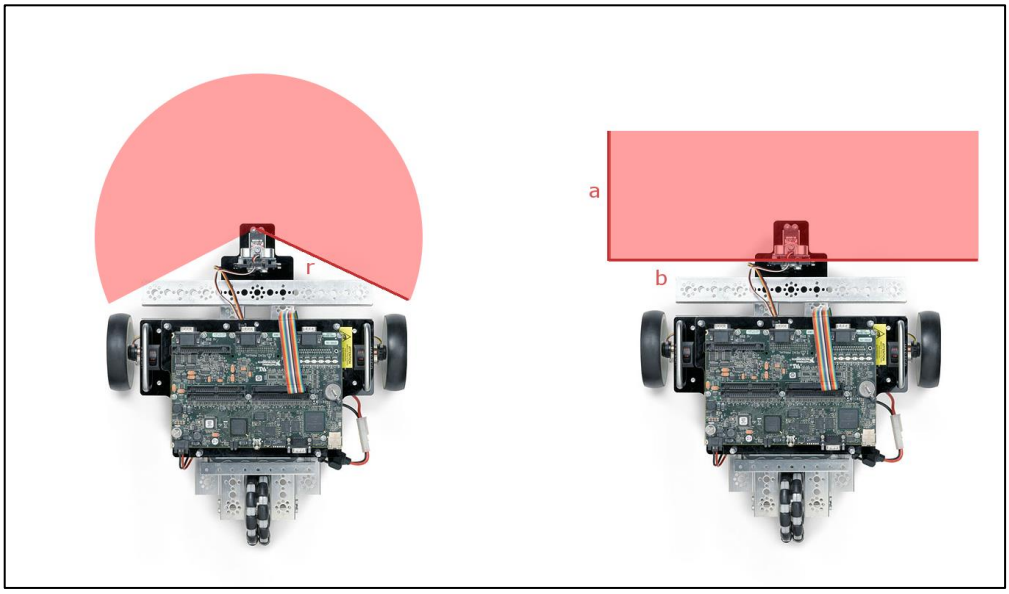
Popojíždění		
Interval [m]	**a**	**b**
<10,20)	-7,50	425
<20,30)	-3,00	335
<30,40)	-1,00	275
<40,50)	-0,50	255
<50,60)	-0,50	255
<60,70)	-0,70	267
<70,80)	-0,20	232
<80,90)	-0,10	224
<90,100)	-0,20	233
<100,110)	-0,20	233
<110,120)	-0,10	222
<120,130)	-0,10	222
<130,140)	-0,10	222
<140,150)	-0,10	222
<150,160)	-0,10	222
<160,170)	0,00	206
<170,180)	-0,10	223
<180, ∞)	-0,02	208

Jak znázorňuje tabulka 5.2, nejsou vypočtenými hodnotami ošetřeny všechny intervaly. Je vynechán interval 0 – 10 a v případě otáčení robota i všechny hodnoty nad 360°. To, že je vynechán první zmíněný interval, je způsobeno tím, že v tomto intervalu se při přibližování k nule koeficient limitně blíží nekonečnu. Proto, když je vypočteno, že se má se robot pootočil o úhel menší než 10° nebo má popojet o méně než 10 cm, automaticky se hodnota zaokrouhlí na hodnotu 10 a použije se příslušný koeficient. A otočení o úhel větší než 360° nejsou brány v potaz z toho důvodu, že takováto otočení nejsou pro náš algoritmus potřeba. Pokud by se měl robot otáčet o více jak 360° jedním směrem, je výhodnější, aby se otočil rovnou směrem opačným.

Vypočtené lineární závislosti pro jednotlivé intervaly jsem poté zanesl do základních pohybových bloků a na základě vstupu do těchto bloků (žádaný rozsah pohybu) je pak vypočten a použit příslušný koeficient.

Další záležitost, kterou jsem řešil před samotným programováním celého algoritmu, byla „nárazníková zóna“, tedy prostor, ve kterém robot bude reagovat na okolní objekty. Při

programování Lego Mindstorms jsem toto řešit nemusel, protože robot ultrazvukovým senzorem neotáčel, a tedy veškeré objekty, které nalezl, byly jen uvnitř výseče pokrytí senzoru. Ale protože DaNI svým senzorem otáčí, dokáže jím prozkoumat mnohem větší oblast.



Obr. 5.11 – Uvažované tvary nárazníkové zóny

Když jsem přemýšlel, jak nárazníkovou zónu pojmout, vykrystalizovaly mi v úvahách nakonec dva tvary: kruhový a obdélníkový. Zjišťování, zda je nějaká překážka uvnitř kruhu, v jehož středu leží ultrazvukový senzor je triviální záležitost – vzdálenost naměřená senzorem se jen porovnává s definovaným poloměrem. Navíc má tento tvar tu nevýhodu, že je-li překážka přímo v ose robota, ten na ní reaguje mnohem dříve, než když je překážka před pravým nebo levým kolem.

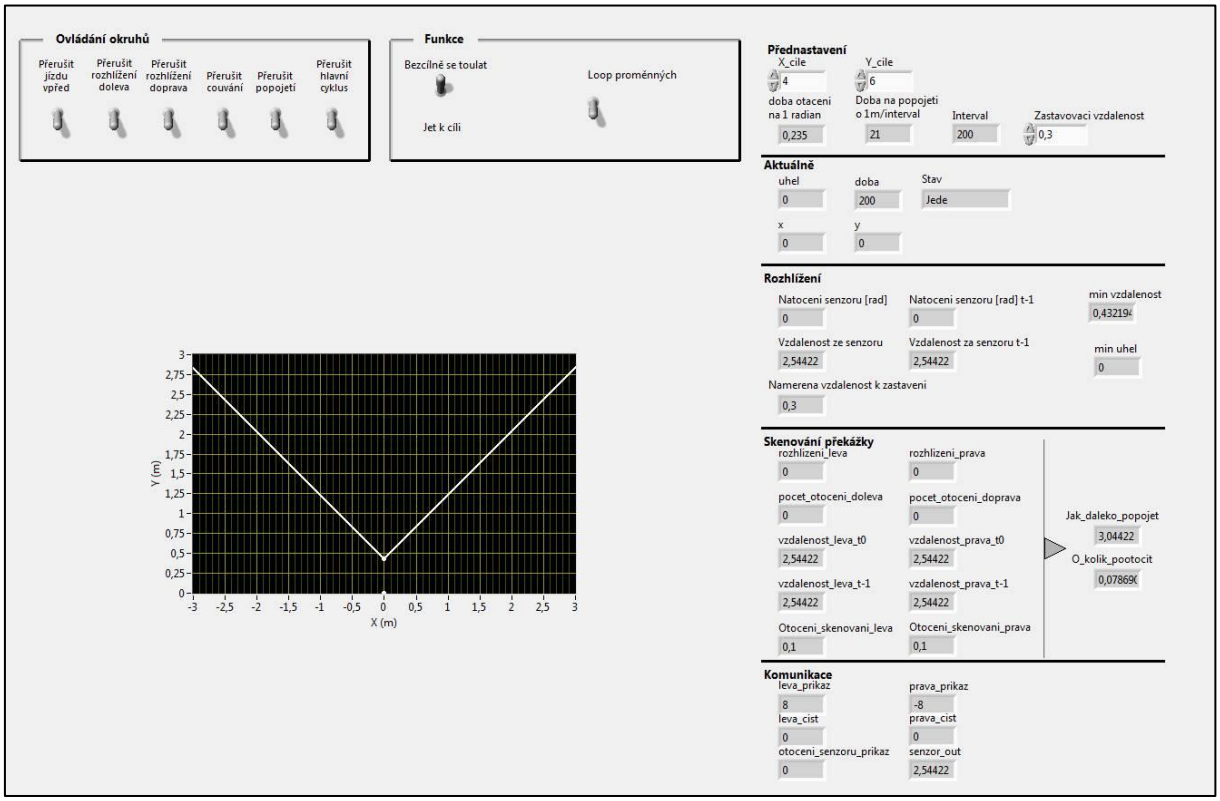
Proto jsem se rozhodl, že vyzkouším nárazníkovou zónu ve tvaru obdélníku. Pro výpočet, zda se předmět nachází uvnitř nárazníkové zóny, se využívá následujících vztahů:

$$v \leq \frac{b}{2 \cdot \cos(90 - \varepsilon)} \wedge v \leq \frac{a}{\sin(90 - \varepsilon)} \qquad (5.2)$$

kde: v ..... senzorem naměřená vzdálenost k objektu
ε..... úhel natočení senzoru od podélné osy robota
a ..... délka nárazníkové zóny
b ..... šířka nárazníkové zóny

Samotné rozhlížení se pak vykonává postupným otáčením ultrazvukového senzoru do pěti poloh. Tyto polohy jsou definovány úhly od osy robota. Konkrétně jde o 0 rad; 0,5 rad; 1 rad; -1 rad; -0,5 rad. Jednotlivá pootočení jsou vykonávána v intervalu, který je nastavitelný na čelním panelu. Vhodné jsou hodnoty pod 200 ms.

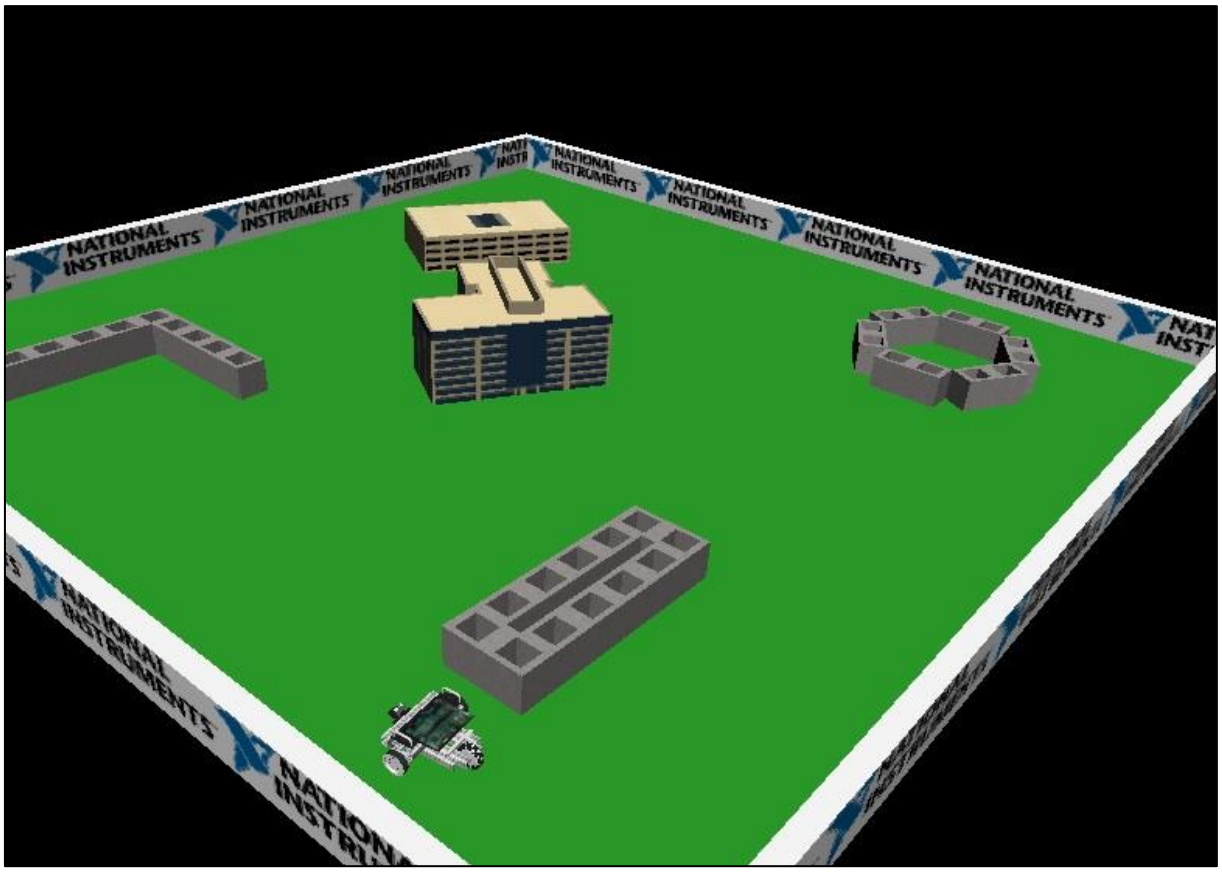
Pokud se však i přesto z jakéhokoliv důvodu dostane robot do takové blízkosti překážky, že jejich vzájemná vzdálenost je menší, než polovina šířky robota, a vzniká tím riziko, že by při manévrování (především otáčení se) mohl do překážky narazit, od překážky odcouvá a teprve poté se jí začne vyhýbat.



Obr. 5.12 – Čelní panel programu pro aktivní vyhnutí se překážce při jízdě k cíli

5.4.1 Poznatky ze simulace

Abych se vyhnul tomu, že by DaNI kvůli chybě v algoritmu špatně reagoval a například se sám poškodil, nejprve jsem vše programoval pro odsimulování v simulačním prostředí, které LabVIEW Robotics nabízí. Robota jsem umístil do prostoru ohraničeného mantinely, uvnitř kterého se nachází několik objektů různého tvaru.



Obr. 5.13 – Ukázka ze simulačního prostředí LabVIEW Robotics

Během simulací bylo samozřejmě zapotřebí odstranit některé chyby vzniklé programováním a vyladit některé parametry (například rozměry nárazníkové zóny). Bohužel se ukázalo, že přepočtové koeficienty v blocích pro základní pohyby v simulaci neodpovídají reálným hodnotám. Proto bylo potřeba koeficienty změnit. Zjistil jsem, že v simulaci jsou koeficienty pro všechny rozsahy pohybu konstantní. Pro otáčení (tentokrát v jednotkách radiánů) je přepočtový koeficient k = 235 a pro popojíždění s otáčkami motorů 5 rad/s je koeficient k = 625. Tyto koeficienty jsem zadal do nových bloků základních pohybů, které jsem vytvořil přímo pro simulaci.

Nové základní pohybové bloky bylo nutné vytvořit i z jiného důvodu, než jen změnění přepočtových koeficientů. Pokud je totiž vytvořeno uživatelem subVI, které na vstupu využívá výstup z bloku Initialize Starter Kit 2.0 (viz kapitola 4.3.2), nelze již do stejného vstupu zapojit výstup z bloku Initialize Simulated Starter Kit 2.0. A tak ačkoliv se jedná o algoritmus se stejnou funkcí, je pro potřeby simulace nutné vytvořit kopie subVI a změnit u nich adekvátně očekávaný vstup a výstup.

Po nastavení správných parametrů algoritmus úspěšně fungoval, robot se autonomně pohyboval prostředím a vyhýbal se překážkám.

Problematické se ukázalo pouze dopočítávání polohy robota pomocí dead-reckoningu. Toto dopočítávání bylo poměrně nepřesné a stávalo se, že robot nebyl schopen zastavit přímo na zadaných souřadnicích cíle. Tato skutečnost ohledně nepřesného dead-reckoningu je ovšem znázorněna i v příkladovém programu, který je součástí LabVIEW, takže lze předpokládat, že se nejedná o chybu mého algoritmu, ale o nepřesnost robotových senzorů.

5.4.2 Poznatky z reálného testování

Když jsem algoritmus začal testovat v reálném prostředí, ukázalo se, že i přes fungující simulaci robot nesprávně reaguje na přítomnost překážek a naráží do nich. Pokusil jsem se toto napravit změnou rozměrů nárazníkové zóny, což sice přineslo zlepšení, ale pro bezkolizní fungování by bylo potřeba nastavit rozměry nárazníkové zóny tak velké, že by robot prakticky nebyl schopen nějakého smysluplného pohybu uvnitř místnosti.

Abych analyzoval, kde přesně se nachází problém a jak ho odstranit, rozhodl jsem se nejprve vykreslovat do grafu vzdálenosti, které robot naměří při svém pětifázovém rozhlížení, později, abych byl získal přesnější povědomí o tom, jak robot svým senzorem vnímá okolí, jsem vykresloval graf z rozhlížení, jehož krokem byl úhel 4°. Díky tomuto postupu jsem zjistil, že problém s odrazivostí ultrazvuku popsaný na obr. 4.6, je markantnější, než jsem zprvu předpokládal. Největší potíže má ultrazvuk s odhalením malých objektů a objektů, jejichž odrazová plocha je vůči senzoru pod malým úhlem.

Ukázalo se tedy, že ultrazvukový senzor není pro přesné rozpoznávání náhodných překážek dostatečně přesný, a bylo by vhodné využít jiný typ senzoru. Nabízel by se radar, nebo LIDAR, které se v praxi v automobilech již poměrně běžně využívají.

V reálném prostředí je pak nepřesnost dopočítávání polohy robota v prostoru metodou dead reckoningu ještě nepřesnější, než v simulaci. A to i přes to, že jsem si od robota DaNI sliboval mnohem lepší přesnost tohoto odhadu, než u Lego Mindstorms. Ale i přes lepší řešení podvozku robota a jeho kontaktu s podložkou se tento předpoklad vůbec nenaplnil. Nejpravděpodobněji je to způsobeno nedostatečně přesnými enkodéry otáček motorů, které jsou popsány v kapitole 4.1.3.

Závěr

V diplomové práci byl vytvořen přehled současného stavu techniky v oblasti autonomních vozidel. Byl zmíněn jejich historický vývoj, současný legislativní rámec a problémy, se kterými je třeba se vypořádat dříve, než se autonomní automobily dostanou do běžného komerčního provozu.

Dále bylo sestaveno několik algoritmů (Zastavení před překážkou; Vizuální rozpoznání dopravních informací; Adaptivní tempomat; Aktivní vyhnutí se překážce při jízdě k cíli), které byly po naprogramování aplikovány na modelová zařízení. Jako tato zařízení posloužil robot Lego Mindstorms NTX 2.0 a robot NI LabVIEW Robotics Starter Kit 2.0, jinak nazývaný DaNI.

Ze zkušeností získaných při programování a testování robota Lego Mindstorms lze říci, že tato robotická stavebnice se skvěle hodí k úplným začátkům práce s roboty. Práce ve vývojovém prostředí je intuitivní, robot snadno přestavitelný, a v případě zájmu je možnost rozšířit jednoduchou senzorovou výbavu. Využití tohoto robota si umím nejlépe představit při výuce v roboticky zaměřeném předmětu na střední nebo vysoké škole. Pro univerzitní výzkum už vážnější využití nemá, ale nepředpokládám, že tyto ambice jeho výrobce měl.

Naproti tomu robot DaNI je velmi sofistikované zařízení, jehož pochopení a práce s ním zpočátku jednoduchá není. Začít pracovat s tímto robotem pouze na základě manuálů a příruček je velmi obtížné. Pro pracovní komfort je potřeba prozkoumat příkladové programy, které nabízí software LabVIEW, a především získat zkušenosti a know-how, které je skrze příručky nepřenositelné. Když se člověk ale dostane přes tyto počáteční překážky, pochopí, jak všestranný aparát má k dispozici. K tomuto robotovi lze připojit téměř jakékoliv periferní zařízení a rozšířit tak v případě potřeby jeho schopnosti, kterýmkoliv směrem.

Bohužel se při testování algoritmů ukázalo, že ultrazvukový senzor, kterým je robot vybaven, není pro naše potřeby dostatečně přesný. Proto bych navrhoval opatřit ho v budoucnu senzorem přesnějším. Nejlépe použít LIDAR nebo radar. Ovšem ani námi nainstalovaná IP kamera nebyla úplně ideální. Pro další práci s obrazovým vstupem bych doporučil investovat peníze do pořízení kamery, která obraz nekomprimuje. Zpracování detailnějšího obrazu by pak bylo mnohem přesnější a spolehlivější. Dále by bylo vhodné robota dovybavit zařízením pro zjišťování polohy.

Pevně věřím, že tato diplomová práce je přínosem nejenom pro mě, protože díky ní jsem se krom spousty jiného naučil pohybovat v LabVIEW, ale mohla by být zároveň přínosem pro akademickou obec. Především pak pro studenty, kteří by mnou získané poznatky mohli využít při práci na stejném robotovi, jehož šíře využití je skutečně obrovská. Znovu musím zdůraznit, že robot DaNI je natolik všestranným nástrojem, že na práci s ním by mohlo postavit své závěrečné práce mnoho studentů nižších ročníků. Já sám bych chtěl na tuto práci navázat v doktorském studiu a využít robota jako platformu pro práci se senzorem, který se používá v automobilech.

Citovaná literatura

[1] Autonomous Cars. *Autonomous Cars.* [Online] [Citace: 23. dubna 2014.] http://www.autonomouscars.com/.

[2] Autonomous car. *Wikipedia, The Free Encyclopedia.* [Online] [Citace: 15. dubna 2014.] http://en.wikipedia.org/wiki/Autonomous_car#cite_note-148.

[3] DARPA Grand Challenge. *Wikipedia, The Free Encyclopedia.* [Online] [Citace: 20. dubna 2014.] http://en.wikipedia.org/wiki/DARPA_Grand_Challenge.

[4] **Vlk, František.** Automaticky a autonomně řízená silniční vozidla. *Automa.* 2013, roč. 19, č. 1, stránky 12-15.

[5] **Vidrman, Tomáš.** Budou na silnicích jezdit auta bez řidičů? *Trendy.* Český rozhlas, 29. dubna 2015.

[6] **Knight, Will.** Driverless Cars Are Further Away Than You Think. *MIT Technology Review.* [Online] [Citace: 20. dubna 2014.] http://www.technologyreview.com/featuredstory/520431/driverless-cars-are-further-away-than-you-think/.

[7] **Santos, Alexis.** Nissan Leaf prototype becomes first autonomous car to hit Japanese highways. *Engaget.* [Online] [Citace: 4. května 2014.] http://www.engadget.com/2013/11/26/nissan-leaf-is-first-autonomous-car-on-japanese-public-roads/.

[8] **Maléřová, Michaela.** *Tři zákony robotiky ve vědecko-fantastické literatuře.* Olomouc : Filozofická fakulta Univerzity Palackého, 2010.

[9] **Asimov, Isaac.** Hra na honěnou. *Robohistorie I.* Praha : TRITON, 2004.

[10] Lego Mindstorms NXT. *Wikipedia, The Free Encyclopedia.* [Online] 13. února 2015. [Citace: 2. března 2015.] http://en.wikipedia.org/w/index.php?title=Lego_Mindstorms_NXT&oldid=646906741.

[11] **Roberta.** Unterschiede EV3/NXT. [Online] [Citace: 2. března 2015.] http://roberta-home.de/sites/default/files/Tutorial_Unterschiede_EV3_NXT_07-01-2013.pdf.

[12] **Lego.** Zákaznická podpora. *Lego MINDSTORMS.* [Online] [Citace: 2. března 2015.] http://www.lego.com/cs-cz/mindstorms/support.

[13] **Kamal, Ibrahim.** WFR, a Dead Reckoning Robot. *IKALOGIC.* [Online] 15. dubna 2008. [Citace: 26. března 2015.] http://www.ikalogic.com/wfr-a-dead-reckoning-robot/.

[14] **National Instruments.** NI LabVIEW Robotics Starter Kit Datasheet. [Online] [Citace: 13. dubna 2015] http://www.ni.com/datasheet/pdf/en/ds-217.

[15] **National Instruments.** What Is NI Single-Board RIO? *Single-Board RIO.* [Online] [Citace: 13. dubna 2015] http://www.ni.com/singleboard/whatis/.

[16] **National Instruments.** NI Single-board RIO Quick Reference - sbRIO-961x/963x/964x. 2008. 325062A-01.

[17] **Parallax.** PING))) Ultrasonic Distance Sensor . *Parallax.* [Online] 4. února 2013. [Citace: 14. dubna 2015.] https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf.

[18] **Pitsco Education.** Technical Specs - TETRIX MAX DC Gear Motor. [Online] 26. září 2013. [Citace: 16. dubna 2015.] https://c10645061.ssl.cf2.rackcdn.com/resources/tetrix/152motor739530.pdf.

[19] **Pitsco Education.** TETRIX® MAX DC Gear Motor. [Online] [Citace: 16. dubna 2015.] http://www.pitsco.com/TETRIX_DC_Gear_Motor.

[20] **Pitsco Education.** NI Encoder Kit. [Online] [Citace: 16. dubna 2015.] http://www.pitsco.com/NI_Encoder_Kit.

[21] **Khlebovich, Pavel.** *IP Camera Adapter 2.0.* [Online] [Citace: 20. ledna 2015.] http://ip-webcam.appspot.com.

[22] **Vlach, Jaroslav, Havlíček, Josef a Vlach, Martin.** *Začínáme s LabVIEW.* Praha : BEN - technická literatura, 2008. ISBN 987-80-7300-245-9.

[23] **National Instruments.** *Nápověda programu Labview 2014.*

[24] *Vyhláška MDaS 30/2001 Sb. ze dne 10. ledna 2001, kterou se provádějí pravidla provozu na pozemních komunikacích a úprava a řízení provozu na pozemních komunikacích (v původním znění).* **Česká republika.** 2001. Sbírka zákonů České republiky.

[25] **Audi.** Radost z větší volnosti. *Audi.* [Online] [Citace: 21. dubna 2015.] http://app.audi.cz/technika/tempomat/.

Seznam obrázků

Seznam tabulek

Seznam příloh na CD

Příloha č. 1: Zdrojový soubor programu pro Lego Mindstorms
Příloha č. 2: Zdrojový soubor projektu s programy pro robota DaNI