VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY
FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

NÁVRH A REALIZACE VESTAVĚNÉHO SYSTÉMU ŘÍZENÍ MOBILNÍHO ROBOTU BENDER II

DESIGN AND REALIZATION OF EMBEDDED CONTROL SYSTEM FOR MOBILE ROBOT BENDER II

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
JAN MEINDL

VEDOUCÍ PRÁCE
SUPERVISOR
doc. Ing. STANISLAV VĚCHET, Ph.D.

BRNO 2015

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2014/2015

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Jan Meindl

který/která studuje v **bakalářském studijním programu**

obor: **Aplikovaná informatika a řízení (3902R001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Návrh a realizace vestavného systému řízení mobilního robotu Bender II

v anglickém jazyce:

Design and realization of emebedded control system for mobile robot Bender II

Stručná charakteristika problematiky úkolu:

Cílem práce je návrh a realizace vestavného systému určeného pro autonomní mobilní robot Bender II. Navržený systém musí umožnovat řízení směru a rychlosti pohyby robotu. Dále musí umožnovat snímání dat z ultrazvukových snímačů vzdálenosti a dalších telemetrických snímačů. Součástí práce je návrh API pro komunikaci s nadřazeným počítačem.

Cíle bakalářské práce:

1) Seznamte se s aktuálními možnostmi návrhu vestavěných systémů.

2) Navrhněte požadovaný systém vhodný pro robota Bender II.

3) Navržený systém realizujte.

4) Proveďte ověřovací experimenty.

Seznam odborné literatury:

[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series). Intelligent robotics and autonomous agents. The MIT Press, August 2005.

[2] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents). The MIT Press, June 2005.

Vedoucí bakalářské práce: doc. Ing. Stanislav Věchet, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2014/2015.

V Brně, dne 24.11.2014

L.S.

Ing. Jan Roupec, Ph.D. Ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. Děkan fakulty

Abstrakt
Tato bakalářská práce se zabývá návrhem vestavěného řídicího systému pro čtyřkolový mobilní robot Bender II. V první části se zabývá rozborem použitelných platforem pro vestavěný řídicí systém a kinematikou čtyřkolového podvozku. V druhé části se věnuje popisu mikrokontroléru Mbed, praktické konstrukci napájecího zdroje pro řídicí systémy a dále je popsán rámec vytvořený pro komunikaci s nadřazeným počítačem. V poslední kapitole se věnuji konkrétnímu řešení měření rychlosti a transformace těchto údajů do souřadnic vyžadovaných nadřazeným systémem, kterým je robotický framework ROS.

Summary
This thesis describes design of the embedded control system for a four-wheel mobile robot Bender II. The first part deals with analysis of applicable platforms for embedded control system and four-wheel chassis kinematics. The second part focuses on describing the microcontroller Mbed, practical construction of the power supply for control systems and describes the framework created for communication with superior computer. The last chapter deals with one particular solution of speed measurement and transformation of this data into coordinates requested by supervisory system, which is a robotic framework ROS.

Klíčová slova
Řídicí vestavěný systém, Ackermanovo řízení, Bender, Mbed, ROS.

Keywords
Control embedded system, Ackerman steering, Bender, Mbed, ROS.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně dle pokynů vedoucího a s použitím uvedené odborné literatury.

Jan Meindl

MEINDL, J.*Návrh a realizace vestavěného systému řízení mobilního robotu Bender II.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 47 s. Vedoucí diplomové práce doc. Ing. Stanislav Věchet, Ph.D..

Na tomto místě chci poděkovat doc. Ing. Stanislavu Věchetovi, Ph.D. za vedení, cenné rady a přátelský přístup během zpracování této bakalářské práce.

Velké poděkování patří mé rodině za podporu a přátelům za trpělivost během tvorby.

A protože celá bakalářská práce byla vytvořena jen s použitím svobodného softwaru, děkuji všem tvůrcům a přispěvatelům open-source projektů, bez kterých by tato práce nemohla vzniknout v této podobě.

Jan Meindl

Obsah

1. Úvod

Značný nárůst dostupného výpočetního výkonu v posledních letech se promítá do mnoha oblastí vývoje a přináší možnosti, které by ještě před deseti lety byly nemyslitelné. Náročné matematické výpočty už nepřísluší jenom velkým výpočetním centrům, ale díky miniaturizaci a energetické úspornosti se přesouvají i na mobilní zářízení. Tento trend může být dobře demonstrován novým procesorem od Nvidie s označením Tegra X1, který má při spotřebě 10 watt větší výkon než superpočítač ASCI Red, který byl v roce 2000 nejvýkonnějším superpočítačem světa se spotřebou 500 000 watt[2]. Za 15 let se tedy výpočetní výkon vztažený na jednotku wattu padesáttisíckrát zvětšil.

Jednou z oblastí, která se díky tomuto nárůstu výpočetního výkonu mohla začít rychle rozvíjet, je mobilní robotika a speciálně autonomní mobilní robotika. Vize, ve kterých budou automobily jezdit autonomně a ve kterých se vyskytují „chytří“ mobilní roboti je velmi stará[3]. Ovšem o tom, že je technologicky možné vyrobit bezpečné autonomní vozidlo se začalo mluvit v roce 2005, kdy DARPA zorganizovala velký závod autonomních vozidel nazvaný DARPA Grand Challenge. V něm měla vozidla za úkol projet bez lidského zásahu trasu dlouhou 132 mil[1][4] v blízkosti města Los Angeles[5]. Na obrázku 1.1 je zachycen robot Case na pozadí právě tohoto města. Trasa byla naplánovaná tak, aby vedla přes několik úzkých horských průsmyků, tunely a po několika mostech. Mapa s trasou byla týmům předána dvě hodiny před startem. Soutěž přes všechny náročné podmínky dokončilo 5 týmů, první s časem pod 7 hodin.

O dva roky později, tedy v roce 2007 se konala soutěž DARPA Urban Challenge, ve které měla vozidla za úkol za dobu kratší než 6 hodin projet trasu dlouhou 60 mil v městském prostředí. Tedy s respektováním všech pravidel provozu, jiných vozidel, přechodů, chodců, semaforů a značek[6].

V roce 2012 DARPA podepsala kontrakt s Boston Dynamics na vývoj humanoidního robota[7]. Ten samý rok začalo dvouleté terénní testování čtyřnohých polo-autonomních robotů schopných nést přes 400 Kg nákladu[2][8]. Účelem je podpora vojáků v terénu, kdy tento robot dokáže sám následovat jednotku. Na začátek června je naplánované finále velké soutěže autonomních robotů s názvem DARPA Robotics Challenge, ve které musí robot bez pomoci člověka zvládat úkoly jako řídit vozidlo, vystoupat po schodech či manipulovat s ručním nářadím[1].

Společnost Google již od roku 2009 velmi intenzivně vyvíjí svůj systém pro autonomní automobily. V současné době vlastní licenci na testování autonomních vozidel ve 4 amerických státech a v autonomním módu absolvovali přes 1 000 000 km, ve velké míře po městech, dosud bez nehody[9]. V tomto roce začne testování v provozu nových, pro tento účel speciálně vyrobených vozidel, které již postrádají volant a pedály[10].

I ostatní výrobci automobilů vyvíjejí systémy pokročilého řízení, počínaje systémy pro automatické parkování a konče aktivní ochranou proti srážce[11], nebo dokonce aktivním autopilotem[12].

[1] Soutěž se jela poprvé již v roce 2004. Tehdy ovšem skončila poměrně neúspěšně, nejlepší tým totiž ujel necelých 8 kilometrů. Další rok byla cena pro vítěze ve výši 1 000 000 $ zdvojnásobena.

[2] Dodavatelem je opět společnost Boston dynamics a robot má označení LS3.

Úkolem této bakalářské práce je vytvořit a popsat řídicí subsystém na čtyřkolový robot Bender II, který má maximální užitnou hmotnost 20 Kg, rychlost kolem 5 km/h a podvozek, který umožňuje jízdu ve vnitřním i venkovním prostředí. Tento robot musí být schopný plnit příkazy nadřazeného počítače týkající se rychlosti a směru pohybu. Musí být schopný sbírat telemetrická data a data z jiných senzorů a posílat je zpátky nadřazenému počítači. Pro tento účel musí být navržen komunikační protokol, pomocí kterého budou data předávána. Robot by měl být schopný po dokončení účastnit se soutěže autonomních mobilních robotů Robotour.

Práce je rozdělena do dvou celků. V prvním krátce představím mobilní robot Bender II. Dále porovnám vybrané možnosti pro tvorbu řídicího subsystému. Srovnám několik platforem, které je možné pro řízení využít, kdy hlavním kritériem je cena, výkon a komunita, která se kolem platformy uskupila. Závěrem srovnám možnosti několika typů senzorů, které je možné využít k lokalizaci a mapování. V části druhé se pokusím předat praktické rady pro tvorbu řídicího subsystému robota, které jsem na tomto projektu získal.

Obrázek 1.1: Robot Case před městem Los Angeles

2. Motivace k práci

Mobilní robotika je stále v poměrně rané fázi vývoje, mnoho projektů probíhá na půdě technických univerzit různých států a jen málo produktů mobilní robotiky již dospělo do stádia komerčního využití. Díky tomu je v této oblasti množství produktů a projektů, které jsou vyvíjeny s důrazem na otevřenost, nejsou zatíženy licenčními omezeními a uskupila se kolem nich značná skupina odborníků a organizací zastřešující jejich vývoj. Jedním takovým projektem je ROS[1], velmi obsáhlý framework, který si klade za cíl co nejvíce usnadnit vývoj softwaru pro mobilní robot.

Tato práce vznikala s cílem vytvořit funkční a spolehlivou platformu, která by mohla být ovládána nadřazeným počítačem, na němž bude nainstalován právě ROS. Díky tomu by bylo možné otestovat možnosti tohoto frameworku, který si v mnoha zemích získal velkou oblibu, ovšem v České Republice se implementací tohoto systému dosud nikdo hlouběji nezabýval.

Pro tento účel byl zvolen robot Bender II. Protože robot byl osazen řídicím systémem instalovaným již v roce 2006, bylo rozhodnuto vytvořit řídicí systém zcela nový podle současných požadavků. Robot se vyznačuje nosností až 20 Kg a tuhým podvozkem, díky tomu je schopný jízdy ve vnitřním i venkovním prostředí a je schopný nést jako hlavní senzor průmyslový, 5 Kg těžký, laserový dálkoměr SICK, čímž se odlišuje od ostatních dostupných robotů.

Nový řídicí systém je vytvářen tak, aby umožňoval bezproblémovou spolupráci s frameworkem ROS, ale aby jej zároveň bylo možné ovládat i jiným způsobem, například mobilním telefonem s operačním systémem Android. Díky tomu by robot mohl sloužit pro více než jeden účel a bylo by na něm možné testovat i jiné algoritmy, například lokalizaci s použitím fázové korelace.

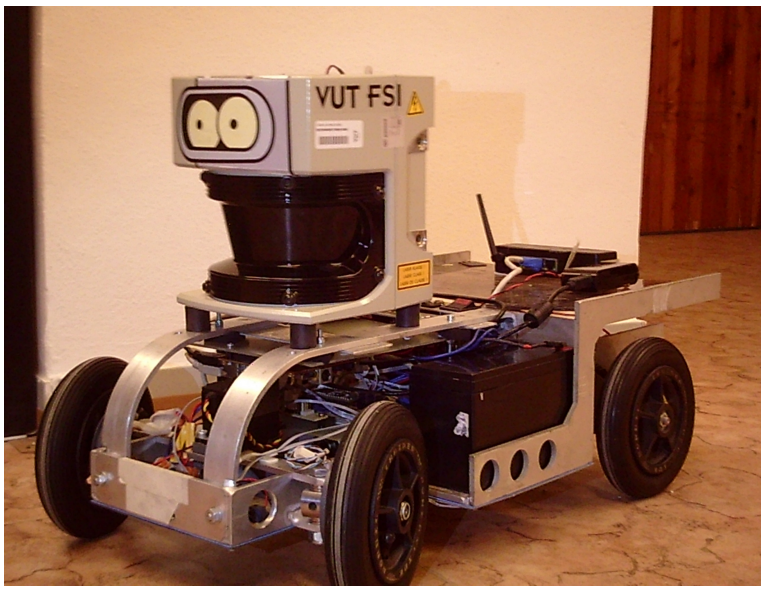
[1] Robot Operating System.

3. Rešerše tématu, možná řešení

V této části je v kapitole 3.1 popsán mobilní robot Bender II z pohledu konstrukce podvozku. Následuje kapitola 3.2 o frameworku ROS, který je využit v nadřazeném počítači k usnadnění navigace robotu. Kapitola 3.3 se věnuje požadavkům kladeným na řídicí systém a srovnáním použitelných platforem pro tento účel. V kapitole 3.4 jsou popsány některé senzory, které je možné na robotu použít k navigaci s uvedením jejich hlavních výhod a nevýhod.

3.1. Mobilní robot Bender II

Robot Bender II je mobilní čtyřkolový robot určený pro autonomní pohyb. Jedná se o projekt VUT FSI z roku 2007. Vnější rozměry jsou 600 x 330 mm, provozní váha robotu v se pohybuje kolem 14 Kg. Je postaven na podvozku ze svařovaných hliníkových profilů s ackermanovým řízením a kyvnou zadní nápravou. Pohon zajišťují dva stejnosměrné motory Maxon umístěné na zadní nápravě, o natočení předních kol se stará silné analogové modelářské servo[32] s tahem 11,0 kg/cm. Vývoj robota byl završen v roce 2009, kdy se účastnil mezinárodní soutěže autonomních outdoorových robotů Robotour a umístil se na pátém místě. Na obrázku 3.1 je robot v současné podobě.



Obrázek 3.1: Mobilní robot Bender II

3.1.1. Podvozek

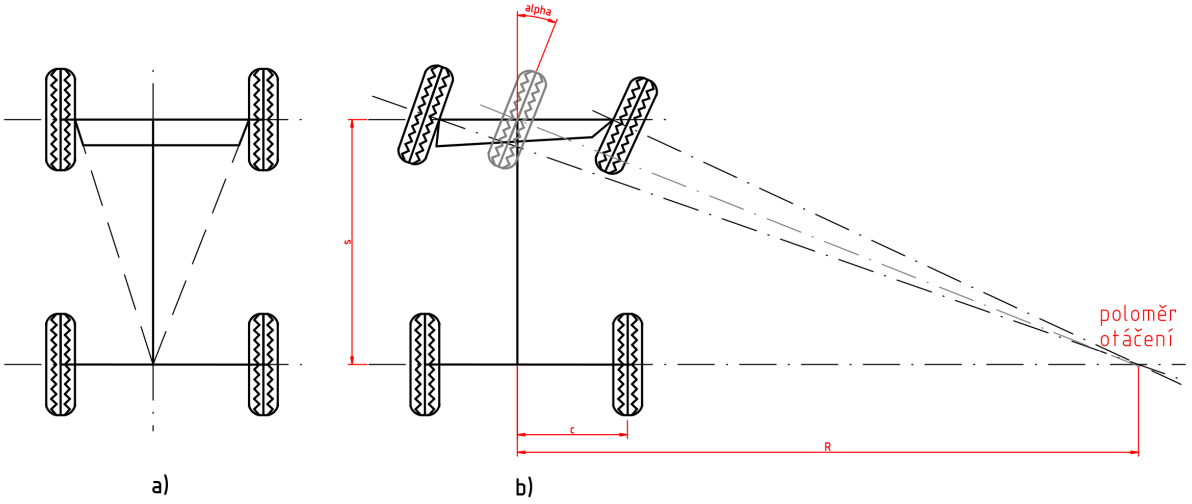
Podvozek je tvořen hliníkovými svařovanými profily. Přední náprava má stavitelnou geometrii a splňuje ackermanovu podmínku (obrázek 3.2), díky tomu osy všech kol směřují vždy do středu rotace, jak ukazuje tentýž obrázek. Tento systém je běžně používán u osobních automobilů.

Výpočet poloměru zatáčeni

Toto řešení s sebou přináší velkou výhodu ve snadném výpočtu poloměru zatáčení, který je nutný pro další řízení pohybu robota[13]. Pokud je splněna ackermanova podmínka, je možné do modelu vložit mezi přidní kola imaginární kolo, které je nakresleno na obrázku 3.3. Osa tohoto kola opět směřuje do středu rotace a úhel natočení kola α odpovídá natočení serva. Tím se celý model zjednodšil na model tříkolky, se kterým je možné počítat dál. Matematicky je možné dokázat, že úhel natočení kola α je shodný s úhlem proti straně c, jak je nakresleno na obrázku 3.3. Úhel β je potom doplněk α do 90°. Díky těmto znalostem již je možné vypočítat pomocí sinové věty (vzorec 3.1) aktuální poloměr zatáčení (vzorec 3.2).

$$\frac{s}{sin(\alpha)} = \frac{R}{sin(\beta)} \tag{3.1}$$

$$R = \frac{s}{sin(\alpha)} \cdot sin(\beta) \tag{3.2}$$



Obrázek 3.2: a) Ackermanova podmínka b) Geometrie podvozku s natočenými koly

Zadní náprava

Zadní náprava je kyvná, jak ukazuje obrázek 3.4. Toto je mechanismus, který je využíván například u někderých nákladních automobilů a který při relativní konstrukční jednoduchost poskytuje neustálý kontakt všech čtyř kol vozidla s povrchem. Díky tomu může robot jet i po náročnějším povrchu, jako je šotolinová cesta nebo dlažba.

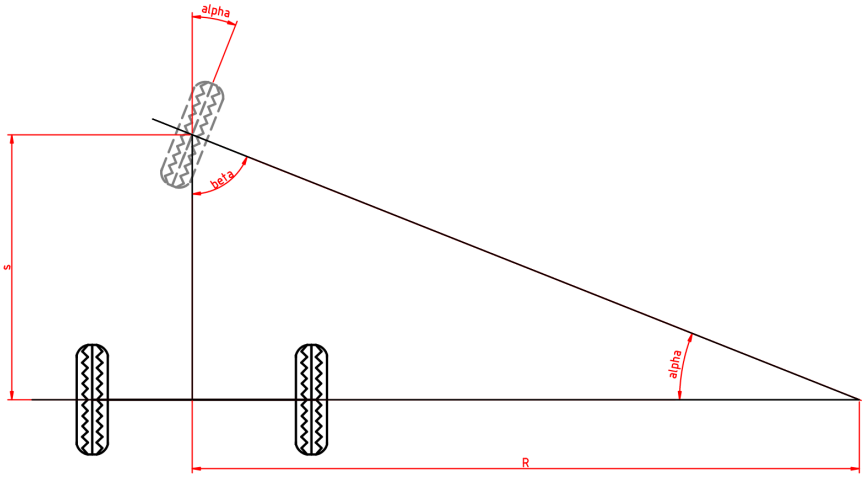
3.1.2. Pohon

Pohyb robota zajišťují dva stejnosměrné motory Maxon na 24 V s integrovanou planetovou převodovkou, každý o výkonu 150 W. Poskytují dostatečný výkon k tomu, aby robot byl schopný plně naložený vyjet po nakládací rampě. Protože každý motor pohání jedno kolo, je potřeba vytvořit softwarový diferenciál, díky kterému má při zatáčení vnější kolo vyšší rychlost než kolo vnitřní. Kdyby toto vyřešeno nebylo, jedno kolo by bylo neustále ve smyku. Více o tom v kapitole 4.4.2.

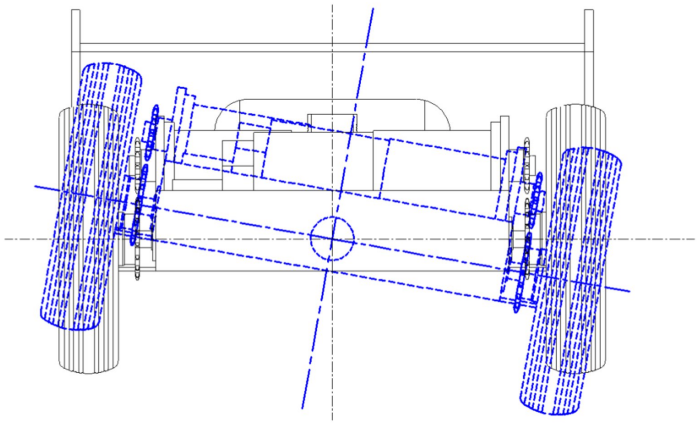
3.2. Robot operating system - ROS

V dnešní době bylo již velké množství lokalizačních a navigačních metod vymyšleno, velice dobře popsáno a byla ověřena jejich dobrá funkcionalita. Je tedy neproduktivní, aby každý vývojář psal celý robotický systém svůj.Vhodně jší by bylo, aby se vývojáři spojili v jednom úspěšném zastřešujícím projektu.

S takovou vizí začal v roce 2007 vznikat na Stanfordské univerzitě projekt nazvaný ROS - Robot Operating System, který o rok později převzal výzkumný inkubátor Willow Garage[1]. Z názvu se zdá, že se jedná o plnohodnotný operační systém. Není to ovšem pravda, jde „jen“ o velmi rozsáhlý framework pracující pod klasickým operačním systémem, nejčastěji pod jednou z linuxových distribucí. Plně podporováno je Ubuntu, v režimu „experimental“ je ale možné využít i jiné linuxové distribuce, Windows, OS X nebo Android.



Obrázek 3.3: Zjednodušený model čtyřkolového podvozku



Obrázek 3.4: Kyvná náprava robotu Bender II [14].

[1] V tomto inkubátoru světlo světa spatřil mimo jiné také velmi úspěšný nástroj OpenCV.

Nejedná se zdaleka o jediný robotický framework, jistě jde ale o jeden z nejpoplárnějších[2]. Spojuje několik velkých výhod. Jednak obsahuje velké množství knihoven pro práci s nejrůznějšími senzory a velmi kvalitními lokalizačními metodami. Dále jsou všechna data v ROSu členěna do tzv. topiců, v nichž jsou předávána mezi jednotlivými procesy. Stejná data tedy může používat naráz více různých programů, přičemž nemusí být spuštěny ani na stejném počítači. Poslední nezanedbatelnou výhodou je to, že systém ROS a všechny jeho součásti jsou vyvýjeny pod licencí BSD[3], díky čemuž je dostupný bez jakýchkoliv poplatků a to i pro komerční použití.

Řídicí systém pro robot Bender II byl konstruován s ohledem na to, aby na nadřazeném počítači mohl být použit právě framework ROS.

Obrázek 3.5: Logo projektu ROS[26]

3.3. Vestavěný řídicí systém

Řídicí systém tvoří vrstvu abstrakce mezi hardwarovými prostředky robotu a vyššími vrstvami řízení, které provádějí lokalizaci robotu a plánování pohybu[25].

3.3.1. High-level a low-level řízení

Řídicí systém je možné rozdělit na dva velké celky, kterými je tzv. *high-level* a *low-level* řízení.

Low-level stupeň má na starosti komunikaci s konkrétním hardware, tedy řízení rychlosti, natočení kol, shromažďování a modifikaci dat ze senzorů. Musí být schopen přijímat příkazy týkající se požadované rychlosti a směru pohybu. Nazpátek musí odesílat zpětnou vazbu o rychlosti a data ze senzorů.

High-level má za úkol celkovou strategii pohybu a plánování. Zpracovává objemnější a výpočetně náročnější data jako jsou snímky kamery či data z laserového dálkoměru.

Tyto dva stupně řízení jsou na sobě v podstatě nezávislé. *Low-level* nepotřebuje mít informace, jaký systém pracuje nad ním. *High-level* oproti tomu nepotřebuje „vědět“, jakým způsobem systém pod ním funguje. Jediné, co tyto dva systémy spojuje je komunikační protokol, kterým si mezi sebou předávají data. Z těchto potřeb plynou i velmi rozdílné požadavky kladené na tyto dva systémy. Vyšší část řízení (*high-level*) vyžaduje dostatečný výpočetní výkon a obvykle kvalitní operační systém na zpracování dat. Vestavěný řídicí

[2] V současné době se jen na tvorbě online dokumentace podílí více než 3 300 uživatelů[26].

[3] Jedná se o jednu z nejsvobodnějších licencí, která obnáší především povinnost zveřejnit autory softwaru, ale neomezuje nijak jeho modifikaci ani použití.

systém (*low-level*) oproti tomu potřebuje dostatečné množství vstupně výstupních pinů a dobrou podporu sběrnic, která je potřeba pro komunikaci jednotky s jednotlivými senzory.

3.3.2. Platforma pro vestavěný systém

Vybrat kvalitní a účelnou platformu pro řídicí systém je jedna z prvních věcí, kterou je třeba při tvorbě robotu vyřešit. Protože Bender 2 je vyvýjen s cílem publikovat projekt jako open-source, přibyly v našem případě další dva důležité požadavky. Prvním je otevřenost platformy a druhým jednoduchá implementace bez nutnosti vytvářet komplikované DPS[4].

I kvůli tomu bylo mým cílem celý vestavěný systém implementovat do jediného mikrokontroléru, místo abych použil větší množství mikrokontrolérů zabezpečujících dílčí činnosti robota a komunikujících mezi sebou po sběrnici. Toto řešení s sebou dále přináší menší hardwarovou složitost, vyžaduje méně datových vodičů uvnitř robota, zjednodušuje komunikaci mezi jednotlivými součástkami a odstraňuje nutnost zabezpečit komunikaci po sběrnici proti rušení. Do užšího výběru postoupily dvě platformy - ARM Mbed a Arduino. Obě platformy mají mnoho společného - jsou vyvýjeny jako open-source, mají velkou komunitu uživatelů a velké množství dostupných knihoven.

Pro naše účely byla vybrána vývojová deska Mbed NXP LPC1768[15]. Podrobnější a přehledné informace o této desce i projektu mbed lze získat v tomto článku[16].

Arduino

Arduino je open-source elektronická platforma založená na uživatelsky přívětivém hardware a software[17]. Hlavní částí je ve většině modelů 8-bitový mikrokontrolér Atmel z řady ATmega[18]. Pro programování je možné používat programovací jazyk C nebo pro účel Arduina vyvinutý jazyk Wiring, který je v podstatě jazyk C s množstvým knihoven, které zjednodušují práci s periferiemi procesoru.

Nezanedbatelnou výhodou je velké množství I/O pinů. Omezením je 8-bitová architekrura, která poskytuje nižší výkon projevující se především v aritmetických operacích s vícebytovými čísly a čísly s plovoucí desetinnou čárkou.

ARM Mbed

Mbed je mladší, ale v některých ohledech ambicioznější projekt než Arduino . Vedoucím projektu je anglická společnost ARM, která spolu s partnery projektu buduje nad 32-bitovými procesory ARM Cortex-M rozsáhlý *ekosystém* umožňující používat rozličné periferie jako sběrnice (I2C, SPI, CAN), USB host, Ethernet, Bluetooth, používání SD karty, AD převodníky a jiné. K dispozici je také real-time operační systém CMSIS RTOS podporující vícevláknové aplikace. Omezením je oproti Arduinu horší dostupnost pro Českou republiku a vyšší cena[5].

[4] Deska plošných spojů.

[5] Cena Arm Mbed, model LPC1768 se pohybuje kolem 75 $. Originální Arduino Micro stojí 18 $. Klon Arduino Micro je možné koupit již za 2 $.

3.3.3. Nadřazený počítač

Následně je potřeba vybrat počítač, který bude mít na starost plánovací algoritmy. V dnešní době jsou možné dva nejjednodušší způsoby. Jedním je použít malý notebook nebo tablet. Druhým je využít jeden z jednodeskových počítačů určených pro prototypování a vývoj.

Raspberry Pi

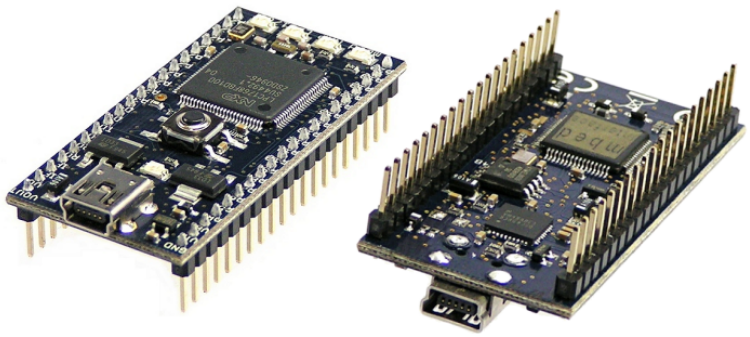
Velkou popularitu získal během posledních let projekt Rasperry Pi. Jedná se o jednodeskový počítač o velikosti přibližně platební karty postavený na architektuře ARM. Obsahuje procesor ARM Cortex-A6 s taktem 700 Mhz, grafický procesor VideoCore IV a 512 Mb paměti RAM. Jako úložiště slouží paměťová MicroSD karta. V lednu 2015 vyšla nová verze, která při stejné ceně a velikosti poskytuje 6x vyšší výkon než verze původní[27].

Beagleboard xM

Podobný jako Raspberry Pi, ale starší projekt je BeagleBoard. Opět se jedná o jednodeskový počítač podobných rozměrů i parametrů. Poháněn je procesorem ARM Cortex-A8 o taktu 1 Ghz, disponuje 512 Mb paměti RAM a jako úložný prostor opět slouží microSD karta[28]. Na robot Bender II byl kvůli vyhovujícím parametrům a dobré dostupnosti vybrán tento jednodeskový počítač.

Nvidia Jetson TK1

Oproti předchozím dvěma platformám je deska Jetson TK1 od společnosti Nvidia zcela odlišná. Při stále kompaktních rozměrech[6] nabízí mnohonásobně vyšší výkon. Obsahuje SoC Nvidia Tegra K1 s 4+1 jádrovým procesorem ARM Cortex-A15 a Kepler GPU se 192 CUDA jádry, které podporují OpenGL 4.4 a OpenCV. Jedná se o shodnou technologii, která je používána v mnoha superpočítačích. Dále je deska vybavena 2 Gb paměti RAM a 16 Gb eMMC úložného prostoru. Toto jsou parametry, které jsou zcela dostatečné ke zpracování obrazu ve vysokém rozlišení v reálném čase. Bohužel adekvátně vyšší je i cena této platformy[7][31].



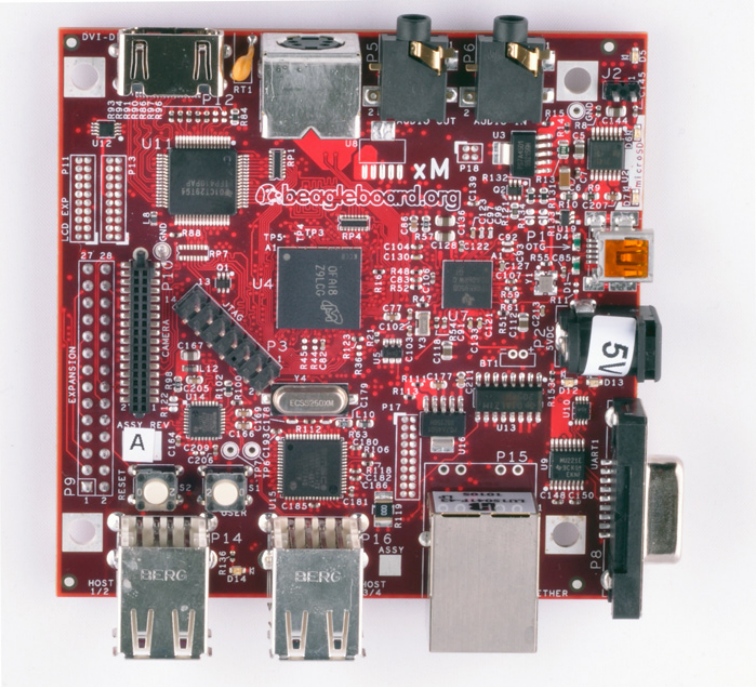
Obrázek 3.6: ARM Mbed NXP LPC1768[30]

[6] Deska má rozměr 127x127 mm.

[7] V USA je cena stanovana na 192 $.

3.3.4. Použitá platforma

Pro nadřazený počítač byl z důvodu dostupnosti a ceny zvolen Beaglebord-xM. Platformou pro vestavěný řídicí systém byl vybrán ARM Mbed, především díky většímu výkonu ve srovnání s Arduinem. Po prvních experimentech bylo zjištěno, že Mbed kvůli použití operačního systému není schopný poskytnout dostatečnou vzorkovací frekvenci pro optické snímače polohy předních kol a pro tento účel bylo do projektu přidáno Arduino Nano v režimu I^2C slave.



Obrázek 3.7: Jednodeskový počítač BeagleBoard-xM[28]

3.4. Mapování prostoru

Pro mobilní robot je velmi důležitá schopnost vytvořit si mapu prostředí, ve kterém se nachází a následně se v tomto prostředí zlokalizovat. Senzorů, které je pro tento účel možné využít je mnoho typů. Několik možných je popsáno v této kapitole.

3.4.1. Stereo kamera

Jedná se o jednu z pasivních metod měření, tedy takových, kdy z robota nic nevyzařuje. Principem těchto metod je obvykle triangulace - stejný signál, který přichází zvenčí, je zachycen více než jedním čidlem a podle drobných rozdílů mezi jejich signály je vypočten směr a vzdálenost cíle[19]. Toto je princip často využívaný v přírodě (zrak, sluch).

V případě stereo kamery je obraz snímán dvěmi kamerami. Z drobných rozdílů a známé vzdálenosti mezi kamerami je potom možné vypočítat vzdálenost k překážce.

Výhodou je velmi komplexní informace o okolí, která lze využít mnoha způsoby. Nevýhodou je náročné zpracování takových dat jak po stránce hardwaru (výpočetní náročnost), tak softwaru (komplikované algoritmy pro zpracování obrazu).

3.4.2. Microsoft Kinect

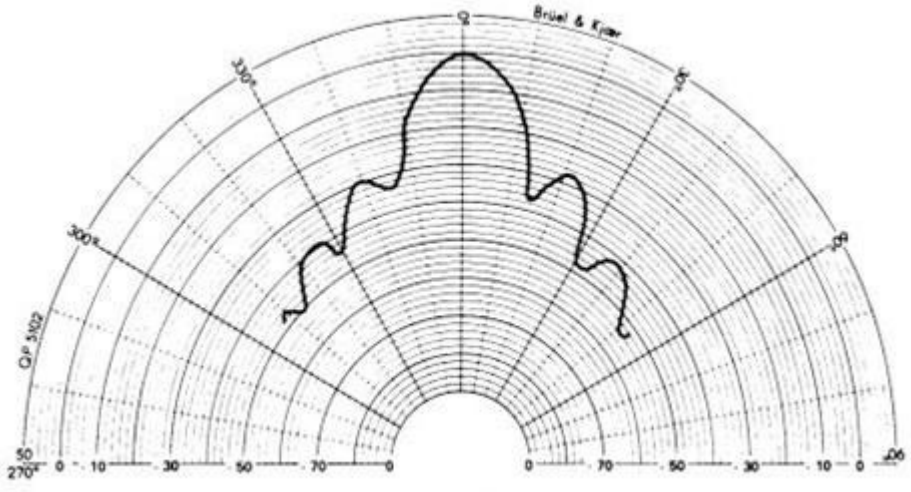
Zajímavým zdrojem trojrozměrných snímků vhodných pro počítačové zpracování je tzv. dvourozměrný senzor hloubky. Senzor hloubky je obvykle založen na principu vyzařování specifického záření, sledování jeho odrazu a následného zpracování získané informace. Snímek vytvořený tímto zařízením je maticí pixelů, kde každý jednotlivý bod představuje jeho vzdálenost od jednoznačně daného referenčního místa (obvykle objektiv či zdroj emitovaného záření). Příkladem takového zařízení je senzor Microsoft Kinect, který hloubkovou mapu vytváří na základě pozorovaného odrazu emitovaného infračerveného vzoru[21].

Taková možnost je téměř ideální, má ovšem několik limitujících omezení. Jednak zpracování probíhá až na počítači a je stále poměrně výpočetně náročné. Druhak infračervený vzor je náchylný k rušení jinými zdroji infračerveného záření a na přímém slunci je dokonce nepoužitelný. Robot Bender 2 se bude pohybovat jak v budovách, tak mimo ně a tento senzor pro něj tedy postrádá smysl.

3.4.3. SONAR

Název SONAR vznikl jako zkratka SOund Navigation And Ranging (zvuková navigace a zaměřování). Jak název napovídá, jedná se o využití vlastností šíření zvuku pro měření vzdálenosti.

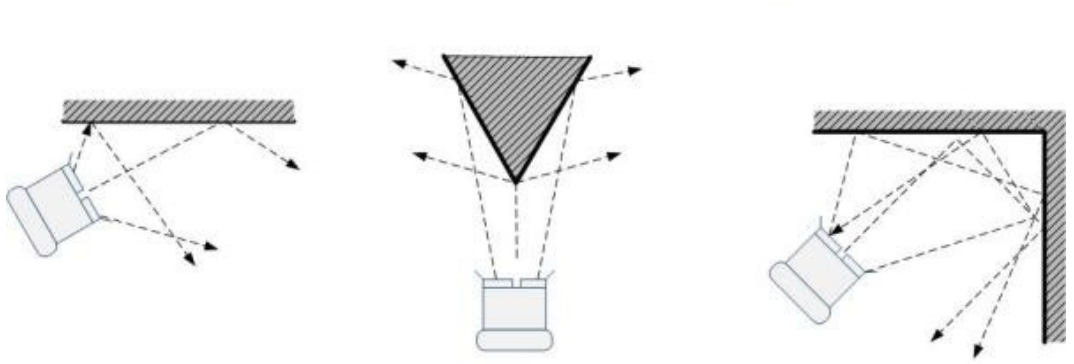
Nejčastější provedení pro účely robotiky je vysílání ultrazvukového pulzu o frekvenci 40kHz a měření času, za který se odražený vrátí. To s sebou přináší několik výhod i nevýhod. Jedná se, obzvláště ve srovnání s jinými metodami, o poměrně jednoduchý princip. Senzory pracující na tomto principu jsou proto cenově dostupné a často připojitelné rovnou na sběrnici, například I^2C [22]. Nevýhodou je především nejistota správného výsledku, protože SONARy jsou náchylné na celou řadu chyb. Zvukový pulz nemá zdaleka podobu paprsku, ale spíše kuželu, jak ukazuje obrázek 3.8 [22]. Nikdy potom není jistota, v jakém úhlu se přesně překážka nachází.



Obrázek 3.8: Vyzařovací charakteristika SONARu SRF08 [22]

Druhou oblastí problému jsou situace, kdy překážka vyskytující se před robotem není detekována, přestože se před senzorem nachází, případně změřená vzdálenost neodpovídá skutečnosti. Takové případy ukazuje obr. 3.9 [23]. Případy, kdy překážka je pro sonar

„neviditelná“ nastává při překročení kritického úhlu k ose sonaru (vlevo a uprostřed). Nutno podotknout, že situace nakreslená na prostředním obrázku je spíše teorietická a s použitím lepších sonarů při běžném použití nenastává. Zato případ, kdy senzor dostane chybná data je poměrně častý a vzniká v případě odrazu zvuku v kolmém rohu.



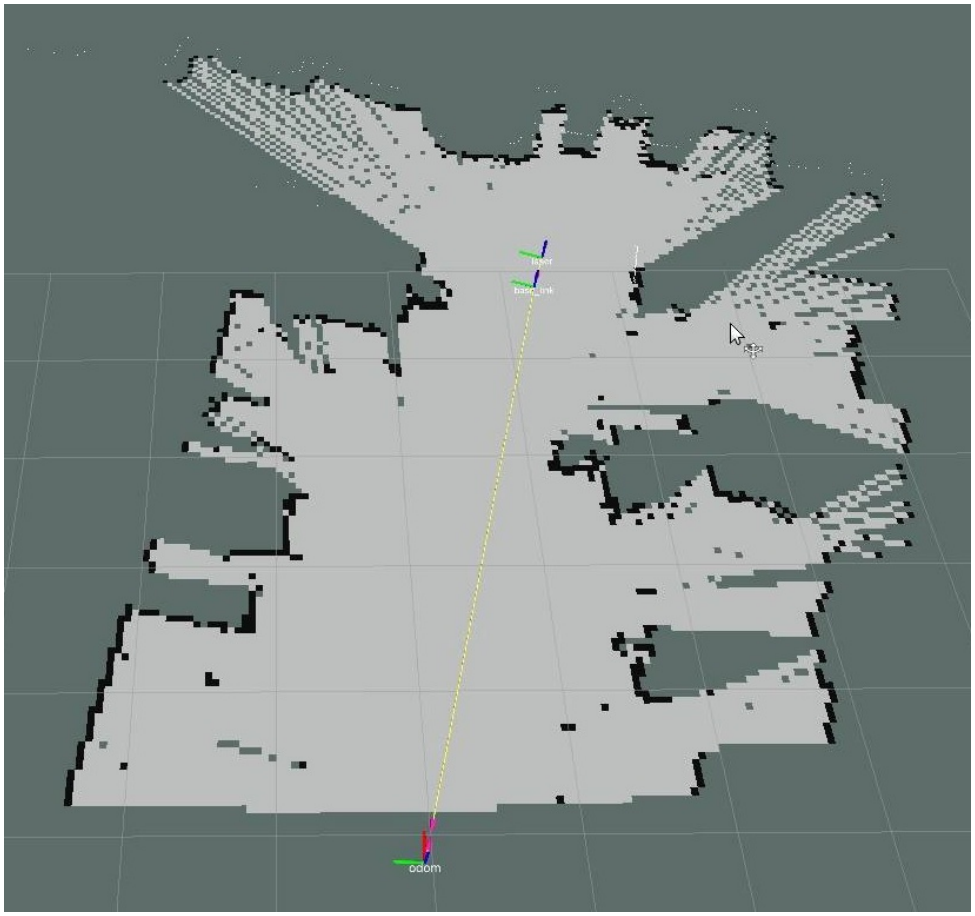
Obrázek 3.9: Problematické případy při měření vzdálenosti SONARem [22]

Tyto problémy celkem spolehlivě řeší použití více senzorů, které jsou natočeny pod různými úhly. Tyto senzory jsem použil na robotu Bender 2 pro snímání prostoru za robotem.

3.4.4. LIDAR

Zkratka z LIght Detection And Ranging (světelná detekce a měření) je zařízení obvykle známé jako laserový dálkoměr. Podobně jako SONAR využívá princip odraženého vlnění, v tomto případě ale světelného. Konstrukce je obvykle řešená tak, že laserová dioda vyšle paprsek na rychle rotující zrcátko a měří, za jakou dobu se vrátí. Díky tomu zařízení vytváří point cloud (mračno bodů), ze kterého je možné zkonstruovat 2D model okolí, ve kterém se nacházíme. Příklad takové mapy vytvořené robotem Bender II je na obrázku 3.10.

Problém je, že zařízení měří čas odrazu světelného paprsku, který má rychlost přibližně 300 000 km/s, za dobu 1 nanosekundy urazí tedy vzdálenost 30 cm. Pokud je potřeba přesnost v jednotkách centimetrů nebo dokonce milimetrů, je potřeba měřit čas s přesností nejvýše desítek pikosekund (10^{-11} s). Laserová dioda také musí vysílat jen velmi krátké pulzy (desítky nanosekund) a aby toho bylo dosaženo, tečou do ní po tuto dobu značné proudy. Pro detekci odrazu je potřeba použít mimořádně citlivé a nízkošumové tzv. lavinové fotodiody[20]. Na robotu Bender 2 je nainstalován LIDAR od firmy SICK, konkrétně model LMS200[24]. Jedná se o průmyslový laserový 2D dálkoměr určený pro vnitřní použití a s možností nastavit mnoho parametrů. Úhel snímání je 100°, respektive 180°. Podle toho nejvyšší rozlišení je 0.25°, respektive 0.5° a z toho vychází počet měřených hodnot buď 401 nebo 361. S dosahem 80 metrů je přesnost ± 15 mm. Tento dálkoměr tvoří hlavní senzor robota, podle kterého detekuje jak dynamické, tak statické překážky a zároveň data z něj jsou použita k lokalizaci robota ve vnitřním prostředí.



Obrázek 3.10: Mapa místnosti vytvořená laserovým dálkoměrem SICK.

4. Realizace

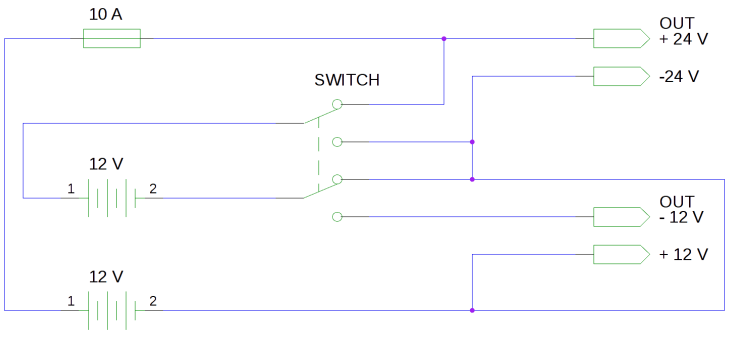
Tato část pojednává o konkrétních problémech, které bylo potřeba na robotu Bender II vyřešit. V kapitole 4.1 jsou popsána vybraná hardwarová řešení. Kapitoly 4.2, 4.3 a 4.4 se zabývají vývojem softwaru řídicího systému.

4.1. Hardwarové řešení

V kapitole 4.1.1 je popsán výběr a zapojení baterií. Kapitola 4.1.2 je věnována výrobě zdroje stabilizovaného napětí a nakonec v kapitole 4.1.3 je probrána celková topologie řídicího systému navrženého pro mobilní robot Bender II.

4.1.1. Baterie

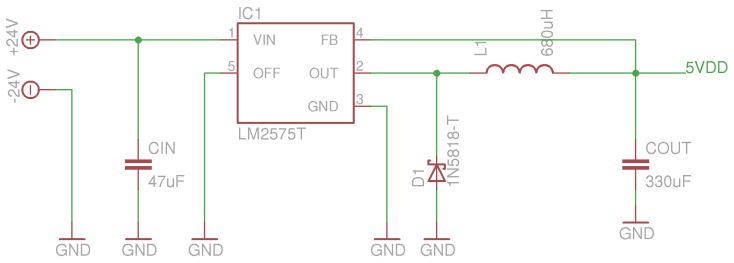
Jako zdroj energie pro robot Bender byly zvoleny dvě gelové olověné baterie, každá o napětí 12 V a kapacitě 7 Ah. Toto řešení má nevýhodu ve zvýšené hmotnosti robotu, ta je ovšem vyvážena tím, že baterie jsou bezúdržbové a pro jejich spolehlivý chod jim není potřeba věnovat zvláštní péči. Protože laserový dálkoměr SICK a regulátor motorů pracují na napětí 24 V, jsou za provozu baterie spojeny sériově. Po vypnutí jsou spojeny paralelně a je možné je tedy dobíjet libovolnou nabíječkou na olověné akumulátory. Toto řešení bylo vytvořeno pomocí dvojitého přepínače, jak ukazuje schéma na obrázku 4.1.



Obrázek 4.1: Zapojení 2 olověných baterií

4.1.2. Napájení

Na robotu je větší množství zařízení, které pracují na různých napětích, konkrétně 5 V (mikrokontrolér Mbed, servo, jednodeskový počítač BeagleBoard), 12 V (WiFi router) a 24 V (regulátor motorů, laserový dálkoměr SICK). 24 V je výstupní napětí bateterií. 5 V a 12 V je řešeno shodně pomocí spínaného regulátoru napětí. Jednodušší řešení by bylo využít lineární stabilizaci napětí, ta ovšem funguje na principu převedení nadbytečného napětí do tepla. Pokud je převáděno napětí 24 V na napětí 5 V, má takový stabilizátor účinnost kolem 20% a zbytek je převeden na teplo, zatímco spínaný zdroj má účinnost kolem 80% a není potřeba ho tedy aktivně chladit. Nejjednodušší zapojení takového stabilizátoru na obrázku 4.2 přitom není o mnoho komplikovanější než zapojení lineárního stabilizátoru.

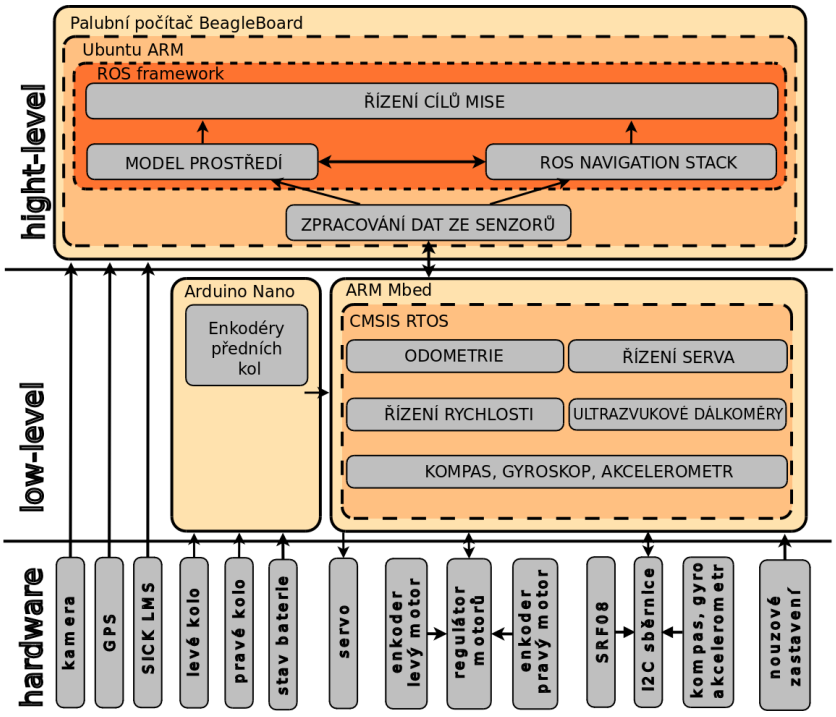


Obrázek 4.2: Zapojení půlzního stabilizátoru napětí

4.1.3. Topologie řídicí části

Na následujícím obrázku 4.3 je zjednodušeně naznačeno, jak celý systém pracuje.

Ve vrchní části se nachází palubní jednodeskový počítač, který se stará o *high-level* řízení, tedy o plánování a navigaci[29]. Na něm je nainstalován operační systém Ubuntu ARM s frameworkem ROS. Přes USB rozhraní přijímá data z laserového scanneru SICK a informace o poloze pomocí GPS. V dalším kroku přibude USB kamera. Nakonec komunikuje, také přes USB rozhraní, s mikrokontrolérem Mbed. Tato komunikace probíhá paketově s kontrolním součtem pro detekci chyb.



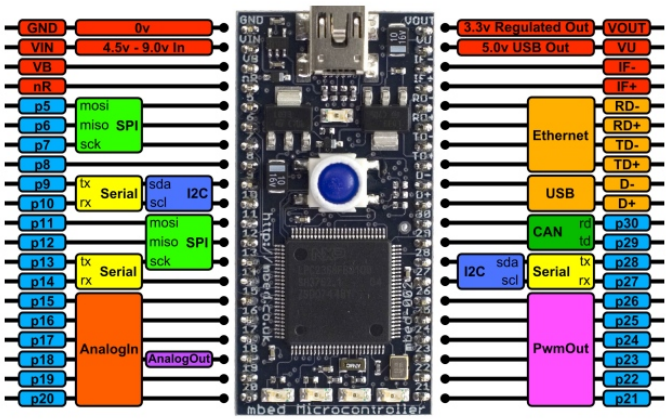
Obrázek 4.3: Topologie zapojení řídicí části

Na tomto mikrokontroléru probíhá zpracování všech zbývajících dat. Po I^2C sběrnici komunikuje s ultrazvukovými dálkoměry SRF08 a čte data ze senzoru, který kombinuje akcelerometr, gyroskop a kompas. Přes UART paketovou komunikací řídí regulátor motorů, na který jsou připojeny optické snímače polohy hřídele a pomocí PI regulátoru jsou řízeny požadované otáčky. PWM signálem dále Mbed ovládá servo. Do systému musel být přidán ještě jeden mikrokontrolér, který slouží jako čítač pro optické snímače polohy

předních kol a měření stavu baterií[1]. Pro tento účel bylo zvoleno v rešerši zmiňované Arduino[2]. Toto zařízení je připojeno na I^{2}C sběrnici v režimu slave.

4.2. Platforma Mbed

V rešeršní části jsem se teoreticky zabýval tím, jaké platformy je možné využít pro řídicí subsystém. Protože systém má být jen s malými úpravami použitelný i pro jiné projekty, je důležitým parametrem otevřenost platformy a podpora komunitou. Je vyžadována snadná implementace a rychlý vývoj. Platforma musí mít dostatek vstupně výstupních pinů a musí podporovat sběrnici I^{2}C. Tyto parametry splňuje ARM Mbed, který má navíc ve své kategorii vysoký výpočetní výkon, proto byl pro naši aplikaci vybrán. Utvořit si představu, jaké možnosti platforma nabízí je možné na obrázku 4.4, kde jsou znázorněny vývody vývojové desky.



Obrázek 4.4: Schéma vývodů platformy Mbed NXP LPC1768[30]

4.2.1. Vývojové prostředí

Obrovskou výhodou a zároveň určitým omezením je originálně pojaté vývojové prostředí určené k programování mikrokontroléru. To totiž probíhá primárně online v okně prohlížeče. Na stránkách výrobce Mbed.org je možné se zaregistrovat, každý uživatel dostane svůj profil a diskový prostor na vzdáleném úložišti a může začít programovat. Kompilace probíhá také na vzdáleném serveru výrobce, který odešle zpátky uživateli hotový binární kód, který je možné stáhnout do počítače. Po připojení se vývojová deska nahlásí jako mass storage zařízení [3] s kapacitou 2 MB. Na toto zařízení je nahrán stažený binární soubor a při restartu zařízení zabudovaný programátor nahraje kód do paměti mikrokontroléru. Není tedy potřeba stahovat žádná data do počítače, konfigurovat kompilátor, stačí podporovaný prohlížeč. Nutno podotknout, že jak flash paměť, tak USB rozhraní má umožněn používat i mikrokontrolér za běhu.

[1] Tento úkol by zvládl i Mbed, ovšem kvůli jeho ochraně byl signál, po kterém může být v případě poruchy přivedeno vyšší napětí, připojen na levnější a méně kritický obvod.

[2] Nebo lépe řečeno jeho levnější klon nazvaný Sanduino, model Nano.

[3] Komunikační protokol využívaný externími zařízeními, jakými jsou například USB flash disky nebo digitální fotoaparáty. Po připojení zařízení je umožněn přenos dat mezi hostitelským počítačem a připojeným zařízením bez instalace ovladačů.

Další výhodou je, že každý uživatel může svůj kód nebo knihovnu zveřejnit online, kde ji mohou používat ostatní. Pro využití cizího software stačí vybrat příslušnou knihovnu, importovat ji do svého projektu a protože vývoj probíhá obvykle pod svobodnou licencí[4], dle potřeby upravovat a případně využít i v proprietárním software[5]. Vše je centralizované na jednom místě a uživatelé nejsou roztroušeni na velkém množství diskuzních fórech, ale jsou na jednom velkém serveru, spravovaném uznávanou autoritou.

Tento systém má dva velké nedostatky. Pro plnohodnotné použití je potřeba mít přístup na internet[6]. Dále zařízení ze své podstaty není možné za běhu ladit. Není žádná možnost, jak nahlédnout do běžícího programu, jak často umožňují konkurenční výrobci.

4.2.2. Rosserial

Jak bylo zmíněno v úvodu a rešeršní části, vyšší řízení bude postaveno nad robotickým operačním systémem ROS, který umožňuje implementovat velké množství algoritmů řešících lokalizaci a plánování. Tento systém má vytvořen svůj komunikační protokol, pomocí kterého jsou předávána data ze senzorů ve formátu, ve kterém je ROS vyžaduje. Tento protokol je možné využít například v mikrokontrolérech Atmel nebo právě na platformě Mbed. Knihovna psaná pro Mbed je ovšem značně zastaralá a na nové verzi firmware ji bohužel není možné použít. V době psaní této bakalářské práce jsem nebyl schopen tento problém ani s podporou komunity vyřešit a proto bylo přistoupeno k řešení napsat komunikační protokol vlastní.

4.2.3. CMSIS RTOS

Při tvorbě vestavěného řídicího systému je možné zvolit dva směry. Jedním je vytvořit decentralizovaný systém složený z dílčích mikrokontrolérů, kdy každý má za úkol svou část řízení a mezi sebou komunikují po sběrnici. To má za následek menší softwarovou složitost, zato složitější hardwarovou konstrukci. Druhou možností je implementovat všechny funkce do jednoho mikrokontroléru, který obstarává vše potřebné a komunikuje jen s nadřazeným počítačem.

V druhém uvedeném případě vyvstává potřeba poměrně komplikovaného softwaru s dobrým časováním. Krokem, který mnoho problémů zjednoduší a zabezpečí bezpečnější běh programu je použít real-time operační systém. Mbed poskytuje řešení v použití knihovny CMSIS RTOS, což je operační systém na bázi Free RTOS přepsaného pro potřeby procesorů ARM Cortex řady M. V systému se jednotlivé procesy dělí na tzv. „Vlákna“, která nabývají různých stavů[7]. Systém obsahuje mnoho synchronizačních algoritmů jako mutex, semafory, signály, fronty, časovače a jiné[8]. Na příkladu 4.1 je ukázka programu, který za použití knihovny rtos čte stav baterie a podle napětí mění periodu blikání LED

[4] Obvykle se lze setkat s licencemi MIT nebo Apache.

[5] Za podmínek daných licenčním ujednáním. To obvykle předpokladá, že svobodná licence bude zachována a text licence spolu se jménem původního autora (autorů) bude dodáván spolu s daným softwarem.

[6] Výrobce deklaruje možnost offline kompilace, mně se ji ovšem i přes značné úsilí nepodařilo úspěšně nakonfigurovat. Proto tuto možnost v současné chvíli označuji za nepříliš vhodnou.

[7] Stavy jsou Ready, Waiting, Running, Inactive.

[8] Více informací poskytují stránky developer.mbed.org/handbook

diody. Ulehčení práce oproti programování mikrokontroléru v čistém jazyce C je zřejmý. Naopak operační systém komplikuje kvůli mutexu [9] komunikaci po sériovém rozhraní[10].

Listing 4.1: Program s použitím knihovny rtos.h

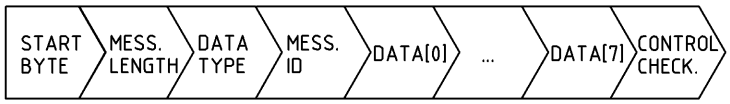
```
1  /*  Ukazkovy program s pouzitim knihovny CMSIS RTOS
2  *
3  *  Podle toho, jake napeti je privadeno na pin 17 blika
4  *  integrovana LED dioda s periodou 1000 ms nebo 200 ms.*/
5
6  #include "mbed.h"          // oficialni knihovna mbed
7  #include "rtos.h"          // knihovna real-time OS
8
9  AnalogIn     battery(p17);    // definice analogoveho vstupu na portu 17
10 DigitalOut   led(LED1);       // digitalni vystup - integrovana LED dioda
11
12 // definice promennych; perioda = led_on + led_off
13 int led_on     = 100;
14 int led_off    = 900;
15
16 void measure(void const *args) {  // proces measure
17   while(true) {
18   // precteni hodnoty na anagovem vstupu a prevod na realne napeti
19   battery_voltage = (26.6)*battery.read();
20   // pokud napeti poklesne pod 20 V, perioda 100 ms, jinak 900 ms
21   led_off = (battery_voltage < 20) ? 200 : 1000;
22   Thread::wait(1000);       // hodnota v ms, po kterou je proces
23                             // v rezimu Wait
24   }
25 }
26
27 void blink(void const *args) {  // proces blinking
28   while(true) {
29   led = 1;                  // nastavi hodnotu digitalniho vystupu na 1
30   Thread::wait(led_on);     // po dobu led_on
31   led = 0;                  // nastavi hodnotu digitalniho vystupu na 0
32   Thread::wait(led_off);    // po dobu led_off
33   }
34 }
35
36 int main() {
37   Thread th1(measure);      // definice procesu measure
38   Thread th2(blink);        // definice procesu blink
39
40   while(1)                  // nekonecna smycka
41   {
42
43   }
44   return 0;
45 }
```

[9] Mutual exclusion - algoritmus zabraňující tomu, aby byly současně vykonávány dva (nebo více) kritické kódy nad stejným sdíleným prostředkem.

[10] Ovšem s přínosem podstatného zlepšení stability systému.

4.3. Paketová komunikace

Vestavěný řídicí systém musí být schopný v pořádku předat nadřazenému počítači data o stavu a poloze robota a na druhou stranu musí být schopný od tohoto počítače přijímat příkazy. Pro komunikaci na robotu Bender jsem navrhl komunikační rámec, který má 13 bytů, jak naznačuje obrázek 4.5. První byte je identifikační pro detekci začátku zprávy a je nastavený na hodnotu 0xF0. Následuje informace o tom, kolik bytů bude v datové části. Maximum je 8. Třetí byte je typ dat (celé číslo, číslo s plovoucí desetinnou čárkou...). Čtvrtý byte nese jednoznačnou identifikaci zprávy. Následuje 8 datových bytů a rámec je uzavřen kontrolním součtem 2. - 12. bytu.



Obrázek 4.5: Komunikační rámec

4.3.1. Softwarové řešení

Hodnoty v bytu *Data type* mohou nabývat hodnot uvedených v tabulce 4.1. Hodnoty int a float jsou posílány jako pole znaků ASCII tabulky. Takové řešení způsobuje velkou redundanci kanálu, ale zaručuje spolehlivé fungování s jakýmkoliv nadřazeným systémem. Možnosti komunikace směřující z nadřazeného počítače do mikrokontroléru Mbed je zaznamenána v tabulce 4.2, komunikace z mikrokontroléru do počítače potom v tabulce 4.3. Mbed posílá data o směru a rychlosti robotu 20x za vteřinu bez ohledu na to, zda jsou přijata a v jakém stavu. Data ze SONARU jsou posílána 10x za vteřinu. Více o hodnotách linear-x, linear-y a angular-z v kapitole 4.4.3.

hodnota	**typ proměnné**
0x01	char
0x02	int
0x03	float

Tabulka 4.1: Hodnoty v bytu *Data type*

ID	**význam**	**data type**	**hodnota**	**poznámka**
0x01	move	char	<001; 202>	001 - 100: jízda vzad 101: stop 102 - 202: jízda vpřed
0x02	stop	char		
0x03	turn	char	<001; 202>	001: plně vpravo 101: střed 202: plně vlevo
0x05	linear speed-x	float		$m*s^{-1}$
0x06	linear speed-y	float		$m*s^{-1}$
0x07	angular speed-z	float		$rad*s^{-1}$

Tabulka 4.2: Možnosti komunikace ve směru počítač → mikrokontrolér

ID	význam	data type	hodnota	poznámka
0x05	linear speed-x	float		$m*s^{-1}$
0x06	linear speed-y	float		$m*s^{-1}$
0x07	angular speed-z	float		$rad*s^{-1}$
0xA0	SRF08_1 data	int	0-600	cm

Tabulka 4.3: Možnosti komunikace ve směru mikrokontrolér → počítač

Možné softwarové řešení pro příjem zprávy je naznačeno ve výpisu 4.2. Odesílání zprávy potom může probíhat podle vvýpisu 4.3. Obě funkce využívají třídu Message, která je ve výpisu 4.4.

Listing 4.2: Funkce pro příjem zprávy

```
/*  Pro prijem zprav je vytvoren 13-ti mistny buffer, ktery po svem
 *  naplneni preda data dal funkci receive    */
int MySerial::receive(char *buff) {
  // kontrola, ze na prvnim miste prisel spravny identifikator
  if (buff[0] == 0xf0 ) {
    checksum = 0;                       // vynulovani kontrolniho souctu
    for (int i = 1; i < 12; i++)  // provedeni kontrolniho souctu
      checksum += (int)buff[i];
    checksum = checksum % 256;
    if (checksum == buff[12]) { // pokud se kontrolni soucet shoduje
      mess.len         = buff[1];  // postupne naplneni hodnot
      mess.data_type   = buff[2];
      mess.id          = buff[3];
      // pokud je zprava typu char, jednoduse se naplni
      if ( mess.type == 1 )
        mess.data_char = buff[4];
      // pokud je promenna typu float, vytvori se pomocne pole znaku
      // delky "len", naplni se a nasledne se prevede na float
      else if (mess.type == 3) { // pokud je hodnota typu float
        char data[mess.len];
        for (int j = 0; j < mess_len; j++) {
            data[j] = buff[4+j];
        }
        mess.data_float = atof(data);
      }
      // pokud vse probehlo v poradku, potom je navratova hodnota 1
      return 1;
      }
    }
    else
      // pokud ve zpracovani nastala chyba, navratova hodnota je 0
      return 0;
}
```

Listing 4.3: Funkce pro odesílání zprávy

```
/*  Analogicky k prijmu se zprava tvori a odesila  */
void MySerial::send(Message mess) {
    int check = mess.type + mess.id;     // kontrolni soucet
    // pomocne promenne
    int size = 0;
    char buff[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
    // podle typu zpravy se cislo rozdeli na prislusne pole znaku
```

```
 8       // hodnota size urcuje, na kolik znaku se cislo rozdelilo
 9       switch (mess.type) {
10           case 1:
11               break;
12           case 2:
13               size = sprintf(buff, "%d", mess.data_int);
14               break;
15           case 3:
16               size = sprintf(buff, "%f", mess.data_float);
17               break;
18       }
19       for (int i = 0; i < 8; i++)   // dopocitani kontrolniho souctu
20           check += buff[i];
21       if(size > 8)     // orez delky, pokud by presahl velikosti 8
22           size = 8;
23       check += size;  // pripocteni velikosti ke kontrolnimu souctu
24       check %= 256;   // prevedeni kontrolniho souctu na char
25       // zde nasleduje funkce, ktera vysledek vezme a odesle ho
26   }
```

Listing 4.4: Třída použitá funkcemi Receive a Send

```
1  class Message
2  {
3      public:
4          int         id;
5          int         len;
6          int         type;
7          int         data_char;
8          int         data_int;
9          float    data_float;
10         Message();
11         ~Message();
12 };
```

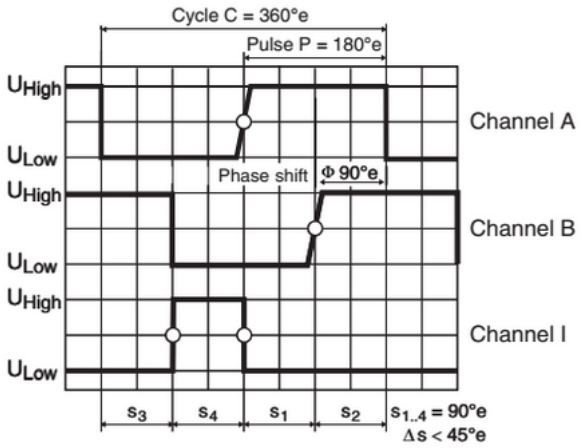
4.4. Řízení rychlosti

4.4.1. Měření otáček motoru

Na hřídeli motorů jsou optické snímače polohy s rozlišením 500 pulzů na otáčku. Každý obsahuje dva kanály, které jsou vůči sobě otočené o $\frac{1}{2}$ pulzu, jak ukazuje obrázek 4.6 a díky tomu lze vyhodnotit směr otáčení. Na robotu Bender II je použit regulátor motorů RoboClaw od firmy Orion robotics, který obsahuje kvadraturní dekodér a umožňuje připojit snímače polohy a řídit otáčky pomocí předprogramovaného PID regulátoru s možností měnit jeho parametry.

4.4.2. Softwarový diferenciál

V případě dvou hnaných kol má v zatáčce vnější kolo větší poloměr otáčení než kolo vnitřní. Je tedy potřeba, aby vnější kolo mělo i vyšší rychlost než kolo vnitřní. Kdyby tomu tak nebylo, jedno kolo by bylo bylo během zatáčení ve smyku a nebylo by možné držet ideální stopu. K tomu slouží diferenciál. Je možné použít buď jeden centrální motor a



Obrázek 4.6: Výstup optického snímače polohy

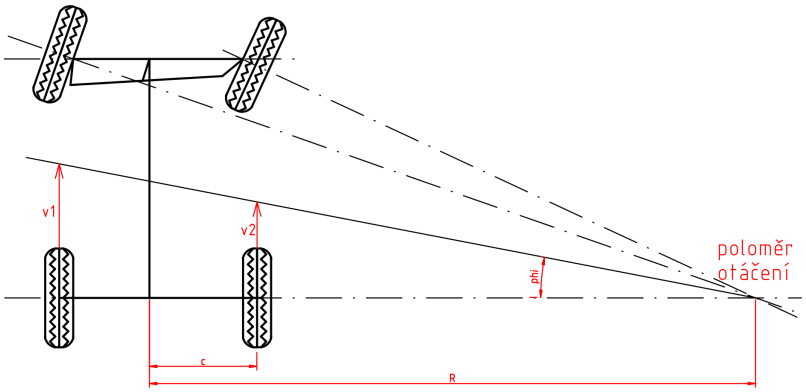
točivý moment na kola rozdělit hardwarovým diferenciálem. Druhou možností je pohánět každé kolo vlastním motorem a poměr rychlostí řídit softwarově. Toto řešení bylo použito u robota Bender II a v této kapitole bude blíže popsáno.

Pokud je známý poloměr zatáčení (kapitola 3.1.1) a požadovaná rychlost robota, potom není velký problém vypočítat rychlosti jednotlivých kol. Řekněme, že požadovaná rychlost je v a nás zajímá rychlost pravého a levého kola l_{speed} a r_{speed}. Z podobnosti trojúhelníků vyplývá vztah 4.1 pro levé kolo a 4.2 pro pravé kolo, kde R je poloměr zatáčení a c vzdálenost středu kola k ose robota (obrázek 4.7). Nezáleží na tom, zda počítáme s lineární rychlost nebo úhlovou, poměr je stejný. V případě úhlové rychlosti je jen rychlost podělena poloměrem kola, jak ukazuje vzorec 4.3. V jazyku C++ je možné daný problém vyřešit podle výpisu 4.5.

$$v_1 = v \times (1 + \frac{c}{R}); \tag{4.1}$$

$$v_2 = v \times (1 - \frac{c}{R}); \tag{4.2}$$

$$\omega = \frac{v}{r} \tag{4.3}$$



Obrázek 4.7: Poměr rychlostí jednotlivých kol v závislosti na poloměru otáčení

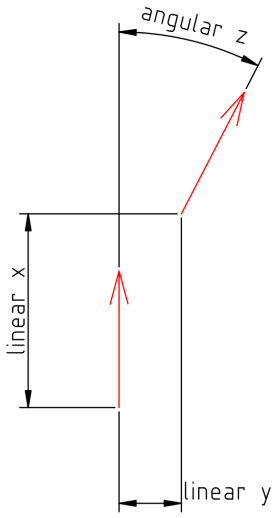
Listing 4.5: Výpočet poloměru zatáčení a rychlosti levého a pravého kola

```
#include  "math.h"
#include  <vector>
#define     M_PI           3.14159265358979323846    // pi

// definice geometrie
float a = 0.093, b = 0.0028, c = 0.1, d = 0.002886, s = 0.4;

// deklarace promennych
float p2, p;
float beta, gama, delta;

// do funkce vstupuje natoceni serva ve stupnich
// funkce vraci polomer zataceni v metrech
float radius(float sigma)
  {
  //doplnek k uhlu a prevod na radiany
  alfa = (M_PI * (sigma + 90)) / 180;
  delta = acos(b/d);
  //pokud se uhel blizi nule, nastav radius na preddefinovanou hodnotu
  if( sigma > -2 && sigma < 2)
    R = 999999.9;
  else {
    // vypocet podle vztahu v kapitole 3.1.1.
    p2 = (float) powf(b, 2) + powf(c, 2) - 2*b*c*cos(alfa);
    p = sqrt(p2);

    beta = acos((p2 + powf(d, 2) - powf(a, 2))/(2*p*d));
    gamma = asin(b*sin(alfa)/p);
    omega = beta + gamma + delta;
    R = (s*sin(\delta))/(sin((M_PI/2) - delta)) [m]
  }
}

// do funkce vstupuje polomer zatoceni v metrech
// vystup je zadana rychlost na leve a prave kolo
std::vector<float> differential(float speed, float R)
{
  // nastavi navratovy typ jako vektor o velikosti 2
  std::vector<float> sp(2);

  sp[0] = speed * (1 + (c/R));     // hodnota pro leve kolo
  sp[1] = speed * (1 - (c/R));     // hodnota pro prave kolo

  return sp;
}
```

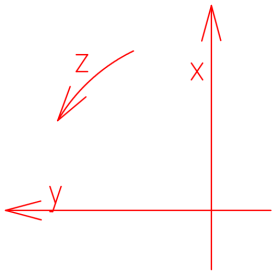
4.4.3. Odometrie

Poloha robota a vyžádaná rychlost je posílána ve tvaru *linear-x*, *linear-y* a *angular-z*, přičemž *linear-x* a *linear-y* je rychlost robota v osách x a y s jednoutkou ms^{-1} a *angular-z* je rychlost rotace robota kolem své osy v $rad*s^{-1}$. Pro lepší přehlednost jsou tyto hodnoty

zakresleny do obrázku 4.8. Červená šipka je orientace robota ve výchozím a koncovém bodu[11]. Orientace os ukazuje obrázek 4.9.



Obrázek 4.8: Význam hodnot linear-x, linear-y a anglar-z



Obrázek 4.9: Orientace os linear-x, linear-y a anglar-z

Výpočet těchto hodnot je ukázán na následujících vzorcích. Rychlost robota ve své ose je průměrem rychlostí levého a pravého kola (vzorec 4.4). Z poměru rychlosti lze vypočítat i rádius, jak ukazuje vzorec 4.5 (který vychází z obrázku 4.7).Nicméně takto získaný radius je značně nepřesný a lepší je použít radius vypočítaný v předchozí kapitole.

$$v = \frac{v_l + v_r}{2} \tag{4.4}$$

$$R = \frac{c \times (v_r + v_l)}{2 \times (v_r - v_l)} \tag{4.5}$$

Z obrázku 4.7 vychází i vzorec 4.6. Je možné matematicky snadno dokázat, že úhel ϕ je stejný, jako otočení robotu kolem své osy. Z úhlu už je snadné dopočítat rychlost v ose x 4.7 a y 4.8. Pokud jsou tyto vztahy napsány v programu C++, dostáváme kód ve výpisu 4.6.

$$\phi = \frac{v}{R} \tag{4.6}$$

[11] Je potřeba upozornit, že se nejedná o vzdálenost, ale o rychlost.

$$x = R \times sin(\phi) \tag{4.7}$$

$$y = R \times (1 - cos(\phi)) \tag{4.8}$$

Listing 4.6: Výpočet hodnot linear-x, linear-y a anglar-z

```
#include <vector>

// do funkce vstupuje vektor o velikosti 2 s informaci o rychlosti
// smer pohybu robota a znamy radius
std::vector<float> getStandardSpeed(std::vector<int> speed, int dir,
                                    float radius) {
  // deklarace vektoru o velikosti 3, ktery bude pouzit jako navratova
  // hodnota funkce
  std::vector<float> data(3);
  // deklarace promennych
  int dir;
  float v, v1, v2, fi;

  // vypocet podle vyprcu v kapitole 4.4.3
  v = (v1+v2)/2;       // rychlost robota v ose
  //radius. Kvuli kladnemu smeru os je potreba otocit znamenko
  R = -radius;
  fi = v/R;                 // natoceni
  x = R*sin(fi);            // linear-x
  y = R*(1-cos(fi));        // linera-y
  z = fi;                   // angular-z

  if(dir == 0) {            // pokud by robot jel vzad
      x = -(x);
      z = -(z);
  }

  data[0] = _x;             // naplneni promenne
  data[1] = _y;
  data[2] = _z;

  return data;              // odevzdani vysledku
}
```

5. Závěr

Tvorba řídicího systému pro mobilní robot je komplexní mezioborová problematika zahrnující oblast elektrotechniky, mechaniky a programování.

V rešeršní části jsem se v kapitole 3.1 zabýval obecnými parametry robota Bender II se zaměřením na kinematiku čtyřkolového podvozku a možnostmi jeho řízení. V kapitole 3.2 jsem krátce popsal framework ROS, který je použit v nadřazeném počítači pro plánování pohybu robota a zmínil jeho základní vlastnosti a výhody, které poskytuje. Nakonec byly v rámci rešeršní části provedeny studie tvorby vestavěného řídicího systému pro mobilní robot. Kapitola 3.3 pojednává o platformách použitelných pro vestavěný řídicí systém, kapitola 3.4 popisuje vybrané senzorické vybavení pro mapování prostředí, ve kterém se robot pohybuje.

V praktické části byly popsány vybrané problémy stavby řídicího systému. Kapitola 4.1 je věnována hardwaru robota, konkrétně výběru a zapojení použitých baterií, stavbě zdroje stabilizovaného napětí a celkové topologii systému. V kapitole 4.2 je více do hloubky rozebrána platforma, která byla vybrána pro řídicí systém. Následuje kapitola 4.3 s popisem vytvořené paketové komunikace mezi vestavěným systémem a nadřazeným počítačem, po které jsou posílána data ze senzorů a příkazy týkající se pohybu robota. Celá část je uzavřena řešením řízení směru a rychlosti použitým na robotu Bender II.

Vytvořený řídicí systém ovládá směr a rychlost pohybu robota. Čte data ze senzorů, kterými jsou především optické snímače polohy, ultrazvukové dálkoměry a měření stavu baterie. Cíle této bakalářské práce tedy byly splněny, nicméně stále zůstává velký prostor pro další vývoj.

Do budoucna by bylo vhodné na mikrokontrolér instalovat displej, který by zobrazoval klíčové stavy, ve kterých se robot nachází. Na robotu je instalováno tlačítko nouzového zastavení, ovšem bylo by vhodné přidat nárazníky, které by v případě srážky odstavily motor, neboť robot má značnou sílu a hrozí reálné riziko zranění člověka nebo zvířete či poškození majetku. V této verzi řídicí systém komunikuje pomocí rozhraní USB. Protože ale mikrokontrolér Mbed má přímou podporu rozhraní Ethernet, bylo by pro řízení mnohem vhodnější využít tohoto rozhraní a to z důvodu menší latence, lepší stability, ve výsledku vyšší přenosové rychlosti a paketové komunikace specifikované modelem ISO/OSI.

Literatura

[1] Darpa Robotics Challenge. [online]. 7.5.2015 [cit. 2015-05-07]. Dostupné z: http://www.theroboticschallenge.org/

[2] NVIDIA Launches Tegra X1 Mobile Super Chip. Nvidia. [online]. 4.1.2015 [cit. 2015-05-07]. Dostupné z: http://nvidianews.nvidia.com/news/nvidia-launches--tegra-x1-mobile-super-chip

[3] KOLÁŘOVÁ, Petra. Karel Čapek - R.U.R.. Knihy Omega. [online]. 3.1.2014 [cit. 2015-05-07]. Dostupné z: http://www.knihyomega.cz/clanky-recenze-karel--capek-r.u.r..html

[4] IŠA, Jiří a Tomáš SEVERÝN. Grand Challenge 2005. Robotika.cz. [online]. 15.10.2005 [cit. 2015-05-07]. Dostupné z: http://robotika.cz/competitions/grandchallenge/2005/cs

[5] Photos. CaseWesternReserve University. [online]. 2006 [cit. 2015-05-07]. Dostupné z: http://urbanchallenge.case.edu/photos.html

[6] Urban Challenge. Darpa. [online]. 2005 [cit. 2015-05-07]. Dostupné z: http://archive.darpa.mil/grandchallenge/

[7] Contracts. U.S. Department of Defence. [online]. 13.9.2012 [cit. 2015-05-07]. Dostupné z: http://www.defense.gov/contracts/contract.aspx?contractid=4853

[8] LEGGED SQUAD SUPPORT SYSTEM (LS3). Darpa. [online]. 2012 [cit. 2015-05-07]. Dostupné z: http://www.darpa.mil/Our_Work/TTO/Programs/Legged_Squad_Support_System_%28LS3%29.aspx

[9] The latest chapter for the self-driving car: mastering city street driving. Google Official Blog. [online]. 28.4.2014 [cit. 2015-05-07]. Dostupné z: http://googleblog.blogspot.cz/2014/04/the-latest-chapter-for-self-driving-car.html

[10] O'BRIEN, Matt. Google's 'goofy' new self-driving car a sign of things to come. Mercury news. [online]. 22.12.2014 [cit. 2015-05-07]. Dostupné z: http://www.mercurynews.com/business/ci_27190285/googles-goofy-new-self-driving-car-sign-things

[11] Subaru models with EyeSight® get the highest possible score in IIHS front crash prevention tests. Subaru. [online]. 2014 [cit. 2015-05-07]. Dostupné z: http://www.subaru.com/engineering/eyesight.html

[12] KURYLKO, Diana. Volvo to unleash self-driving cars on Swedish roads. Automotive news. [online]. 2014 [cit. 2015-05-07]. Dostupné z: http://www.autonews.com/article/20150301/OEM06/303029948/volvo-to-unleash-self-driving-cars-on-swedish-roads

[13] CHOSET, Howie, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents). The MIT Press, June 2005. ISBN: 9780262033275.

[14] HRBÁČEK, J., RIPEL, T. and KREJSA, J. Ackermann mobile robot chassis with independent rear wheel drives. [online]. 2010 [cit. 27.5.2015]. Dostupné z: http://www.umt.fme.vutbr.cz/ ruja/vystupy/Bender2/EPE-PEMC2010paper.pdf

[15] mbed LPC1768. ARM Mbed. [online]. 2014 [cit. 2015-05-07]. Dostupné z: http://developer.mbed.org/platforms/mbed-LPC1768/

[16] BURGESS, Phil. Review: mbed NXP LPC1768 microcontroller. HackaDay. [online]. 2014 [cit. 2015-05-07]. Dostupné z: http://hackaday.com/2009/11/21/review-mbed--nxp-lpc1768-microcontroller/

[17] Co je Arduino. CzechDUINO. [online]. 2014 [cit. 2015-05-07]. Dostupné z: http://czechduino.cz/?co-je-to-arduino,29

[18] Products. Arduino. [online]. 2014 [cit. 2015-05-07]. Dostupné z: http://arduino.cc/en/Main/Products

[19] KUBAC, Petr. Z-kamera 1. Robodoupě. [online]. 23.6.2013 [cit. 2015-05-07]. Dostupné z: http://robodoupe.cz/2013/z-kamera-1/

[20] KUBAC, Petr. Z-kamera 2. Robodoupě. [online]. 1.7.2013 [cit. 2015-05-07]. Dostupné z: http://robodoupe.cz/2013/z-kamera-2/

[21] Review: Pokročilé počítačové vidění (1). ČVUT, fakulta biomedicínského inženýrství. [online]. 2014 [cit. 2015-05-07]. Dostupné z: http://popular.fbmi.cvut.cz/optoel/Stranky/Pokrocile-pocitacove-videni-1-Co-to--vlastne-je.aspx

[22] SRF08 Ultra sonic range finder. Robot electronics UK. [online]. 2014 [cit. 2015-05-07]. Dostupné z: http://www.robot-electronics.co.uk/htm/srf08tech.shtml

[23] Ultrasound sensor. Generation robots. [online]. 2014 [cit. 2015-05-07]. Dostupné z: http://www.generationrobots.com/en/content/65-ultrasonic-sonar-sensors--for-robots

[24] LMS200 Laser Measurement Systems. Sicktoolbox. [online]. 15.12.2004 [cit. 2015-05-07]. Dostupné z: http://sicktoolbox.sourceforge.net/docs/sick-lms-technical--description.pdf

[25] HRBÁČEK, Jan. Embedded control system for an autonomous mobile robot: master's thesis. Brno: Brno University of Technology, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky, 2011. 69 p. Supervised by Ing. Jiří Krejsa, PhD

[26] Why ROS?. ROS. [online]. 2014 [cit. 2015-05-07]. Dostupné z: http://www.ros.org/is--ros-for-me/

[27] Raspberry Pi 2 Model B. Raspberry Pi. [online]. 2015 [cit. 2015-05-07]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[28] BeagleBoard-xM. BeagleBoard. [online]. 14.11.2014 [cit. 2015-05-07]. Dostupné z: http://beagleboard.org/beagleboard-xm

[29] THRUN, Sebastian, Wolfram Burgard, and Dieter Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series). Intelligent robotics and autonomous agents. The MIT Press, August 2005. ISBN 9780262201629.

[30] mbed NXP LPC1768. Elektronic Laden. [online]. 26.2.2015 [cit. 2015-05-07]. Dostupné z: http://elmicro.com/de/mbed-nxp-lpc1768.html

[31] The world's first embedded supercomputer. Nvidia. [online]. 2015 [cit. 2015-05-07]. Dostupné z: http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

[32] RIPEL, T. Návrh a realizace konstrukce kolového mobilního robotu. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2008. 35 s. Vedoucí bakalářské práce Ing. Jiří Krejsa, Ph.D.

6. Seznam použitých zkratek a symbolů

CPU	Central Processing Unit - Centrální procesorová jednotka.
CUDA	Compute Unified Device Architecture - hardwarová a softwarová architektura společnosti Nvidia umožňující na GPU spouštět programy psané v C/C++ nebo postavené na technologiích OpenCL.
DARPA	Defense Advanced Research Projects Agency, americká vojenská organizace pro výzkum pokročilých obranných systémů.
DPS	Deska plošných spojů.
GPU	Graphic Processing Unit - Grafická procesorová jednotka.
Kepler	Kódové označení pro architekturu grafických procesorů od společnosti Nvidia vyznačující se vysokým výpočetním výkonem a energetickou úsporností.
mutex	Mutual exclusion - algoritmus zabraňující tomu, aby byly současně vykonávány dva (nebo více) kritické kódy nad stejným sdíleným prostředkem.
PCB	Printed circuit board, viz DPS.
ROS	Robot Operating System. Rozsáhlý framework usnadňující vývoj softwaru pro mobilní robot.
SoC	System on chip - integrovaný obvod obsahující výpočetní jádro a další elektronické obvody.

7. Seznam příloh

Obsah CD

Příloha A

Obsah CD

Přiložené CD obsahuje elektronickou verzi bakalářské práce a ukázky zdrojových kódů v následující adresářové struktuře:

- */src* - ukázky zdrojových kódů
- */* - domácí adresář obsahující bakalářskou práci v elektronické podobě