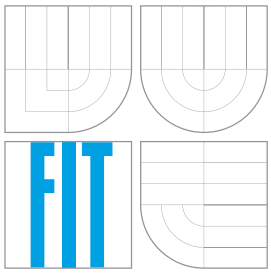
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ROBOT ŘÍZENÝ MIKROPROCESOROVOU JEDNOTKOU PIC

ROBOT CONTROLLED BY PIC MICROPROCESSOR UNIT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. PETR HEŘMAN

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN, Ph.D.

BRNO 2015

Abstrakt

Tato práce popisuje návrh robota z cenově dostupného materiálu. Zahrnuje implementaci firmware vlastní jednotky nižšího řízení založené na mikrokontroléru PIC, která má na starost řízení motorů, získávání dat ze senzorů a komunikaci s jednotkou vyššího řízení. Dále se věnuje realizaci napojení na robotický operační systém ROS a jeho standardní struktury umožňující využití existujících balíčků pro ovládání robota a zobrazování dat z jeho senzorů na jednotce vyššího řízení nebo na vzdáleném počítači. A nakonec popisuje experimenty s výsledným kompletem. Vyrobený prototyp je modelem robotické sekačky trávníku, ale celé řešení je univerzální včetně řídící a senzorové jednotky.

Abstract

This thesis describes design of a cheap robot. It includes implementation of firmware of low level control unit based on microcontroller PIC. The firmware drives motors, gains sensors data and communicates with the high level control unit. Furthermore the thesis presents realisation of connection to the robotic operation system ROS and its standard structures allowing usage of existing packages for the robot teleoperation and displaying sensor data on the remote computer. The thesis finally reports experiments with the robot. The constructed prototype is the model of the robotic lawn mower, however the whole solution has universal usage.

Klíčová slova

PIC mikrokontrolér, robot, zpětnovazebné řízení, PID regulace, senzory, akcelerometr, kompas, gyroskop, hallova sonda, enkodér, sonar, nižší řízení, vyšší řízení, ROS, vzdálené ovládání

Keywords

PIC microcontroller, robot, feedback control system, PID controller, sensors, acclerometer, compass, gyroscope, Hall effect sensor, encoder, sonar, low-level control, high-level control, ROS, remote control

Citace

Petr Heřman: Robot řízený mikroprocesorovou jednotkou PIC, diplomová práce, Brno, FIT VUT v Brně, 2015

Robot řízený mikroprocesorovou jednotkou PIC

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doktora Jaroslava Rozmana. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Heřman
27. května 2015

Poděkování

Děkuji mému vedoucímu diplomové práce, doktoru Jaroslavu Rozmanovi, za čas který věnoval mě a mé práci. Mé díky patří rovněž celé mé rodině, ale hlavně mému otci Milanu Heřmanovi, bez kterého by tato práce nejspíše ani nevznikla.

Obsah

Kapitola 1

Úvod

Robotika je obor úzce spjatý s mechanikou, elektronikou, mechatronikou, umělou inteligencí, počítačovým viděním a mnoha dalšími disciplínami. Jedná se o rychle se rozvíjející oblast lidského zájmu, jelikož jejím hlavním cílem je tvorba strojů, které usnadňují a často i nahrazují lidskou práci. Avšak robotů je více druhů. Jsou to hračky, domácí pomocníci, průmyslové stroje pro masovou výrobu, vojenská technika a v neposlední řadě také zařízení pro akademické i neakademické bádání.

Dalším důvodem rozmachu robotiky je klesající cena elektroniky, její zvyšující se výkon, zmenšující se rozměry a energetické nároky. Nasazení robotů už proto není výsadou velkovýrobního průmyslu, armády a významných vědeckých zařízení. Od jednoduchých průmyslových robotů, které pouze jednotvárně opakují jednoduchý program, se výzkum a vývoj posunul na tvorbu inteligentních autonomních robotů.

Mobilní roboti jsou ti, kteří se dokáží pohybovat v prostoru. Jejich míra autonomie může být různá (od teleoperovaných až po plně autonomní) stejně jako jejich inteligence. Všechny ale stojí na nějakém podvozku, mají nějaký pohon, mají senzory a jsou řízeny nějakou jednotkou. Bez těchto základních částí by žádná autonomie ani inteligence nebyla. Jsou základem mobilního robota.

Volba podvozku, pohonu a senzorů se odvíjí od prostředí, kde má robot operovat, a od činností, které má provádět. Zatímco letající robot bude mít nejspíše vrtule osazené na vysokootáčkových motorech, pozemní robot do vnitřního prostředí bude mít kolový podvozek se zpřevodovanými motory. Zatímco robot mapující neznámý terén bude mít kamery a laserové senzory, nejjednodušší robotický vysavač si vystačí s taktilními senzory. Možností je mnoho a špatná volba může znamenat, že i robot se sebelepší umělou inteligencí nebude schopen plnit své úkoly.

Vybranou kombinaci výše uvedených částí je nutné spojit dohromady a řídit. Řídící a senzorová jednotka bývá prostředníkem mezi vyšším a nižším řízením. Abstrahuje práci podvozku, pohonů a senzorů pro nadřazený systém, který pak může implementovat teleoperaci či autonomní pohyb prostředím. Dobře vytvořená jednotka pak může být využita ve více podobných nebo i různých robotech a tím klesají náklady na jejich výrobu. Často jsou také tvořeny jednotky, které je možné různě rozšiřovat například o další senzory. Druhým extrémem jsou čistě jednoúčelové jednotky, které jsou pevně vázány na jednoho robota. Universalita však bývá vykoupena nutností naddimenzovat výkon a tím dané řešení prodražit.

Práce navazuje na semestrální projekt stejného tématu, který byl vypracován v zimním semestru. Z něj jsou převzaty části kapitol věnující se teorii (řízení, senzory, další systémy

robota) a návrhu software robota.

1.1 Motivace a cíle práce

Cílem této diplomové práce je navrhnout a vyrobit mobilního robota včetně jeho řídící a senzorové jednotky založené na mikrokontroléru PIC od firmy Microchip Technology Inc[1]. Funkcionalita testovacího robota bude vycházet z předpokladu, že by se v dalších fázích vývoje (o kterých již tato práce nepojednává) mělo jednat o domácí robotickou sekačku. Takto zvolený předpoklad určuje několik vlastností, na které bude při návrhu řešení brán zřetel.

- Robot bude operovat ve venkovním zatravněném prostředí.
- Součástky použité pro výrobu - motory, senzory, elektrotechnické součástky - musí být v rozumné cenové hladině (výsledná cena robota v řádu tisíců Kč) a musí být běžně dostupné.
- Rychlost pohybu robota bude odpovídat pomalé chůzi člověka.
- Řídící a senzorová jednotka musí být připravena na komunikaci s nadřazeným systémem, který se bude starat o vyšší řízení.
- Řídící a senzorová jednotka se musí sama zotavit z chyb. Je vyžadován její dlouhodobý běh bez nutnosti zásahu uživatele.

Po úspěšném vytvoření robota je třeba k jeho řídící a senzorové jednotce připojit počítač zodpovědný za vyšší řízení a zobrazování dat ze senzorů. Tento počítač bude instalován na robotovi, přičemž součástí práce je navrhnout a implementovat jeho software, který bude tyto činnosti zajišťovat. Je třeba připravit i software pro vzdálené ovládání pohybu robota a zobrazování dat ze senzorů.

1.2 Existující podobná řešení

Protože mobilních robotů již existuje mnoho, bylo logickým krokem vyhledat řešení, která se zabývají podobnou problematikou jako tato práce. Vzhledem k velmi specifickým požadavkům vytyčeným v rámci podkapitoly 1.1 se však ukázalo, že zatím nebyl představen robot, který by jim všem vyhověl. Následující řešení jsou tedy výběrem těch, které se práci přibližují v jednom či více aspektů a z kterých tato práce nejvíce čerpá.

Roboti, které vyrábí firma Adept MobileRobots[2], jsou určení pro výzkum a vývoj. Jedná se o komplety – podvozek, jednotka řízení a často i integrovaná jednotka vyššího řízení – a jsou připraveny k montáži dalšího příslušenství. Převážně se jedná o roboty určené do vnitřních prostor nebo velmi mírného terénu a všechny je možné napojit na ROS (ROS je blíže popsán v kapitole 4). Asi nejblíže představovanému prototypu je Pioneer 3-AT. V základní konfiguraci ale neposkytuje žádné senzory. Mezi jeho nevýhody patří vysoká váha (12 kg) a výška (26 cm). Mezi výhody lze zařadit dlouhou dobu provozu na baterie (8 hod), velmi přesné enkodéry a možnost integrace vyšší jednotky řízení přímo v podvozku. Tento robot je velmi univerzální a používaný i na FIT VUT.

[1] http://www.microchip.com/
[2] http://www.mobilerobots.com

Další velkou skupinou robotů, operujících ve venkovním prostředí, jsou **profesionální robotické sekačky trávníků** (např. od firmy Husqarna[3]). Jedná se o komerční produkt pro běžného zákazníka. Podvozky těchto robotů jsou připraveny pro práci v obtížném terénu, jako jsou velmi prudké svahy nebo mokrá tráva. Mají kola s hroty a velkým poloměrem. Jejich podvozek byl inspirací pro vytvářeného robota. Problémem těchto robotů je ale takřka nulová senzorická výbava a hlavně nemožnost běhu jiného než integrovaného softwaru.

Co se týká samotných **řídících a senzorových jednotek**, tak jejich základem dnes nejčastěji bývá **Arduino či Raspberry PI a podobné mini-počítače**. Výhodou Raspberry PI je možnost provozovat nižší i vyšší řízení zároveň na jedné desce. Pro plnohodnotný provoz ROSu je ale výkon této platformy nedostatečný. Nevýhodou je nedostatek (nebo úplná absence) periferií – časovače, PWM, A/D převodníky atp. Arduino desky naopak obsahují mikrokontroléry ATmega, které mají periferií dostatek. Jedná se o vhodné řešení pro ty, kteří nemají možnost výroby vlastní desky. Navíc existuje mnoho knihoven a návodů, což usnadňuje vývoj.

Nakonec nelze nezmínit mnoho **robotických hraček a podobných vlastnoručně vyrobených robotů**. Jejich společným jmenovatelem je nízká cena, časté využití Arduino desek (či Raspberry PI) pro řízení a práci se senzory, využití modelářských komponent a bohužel křehkost konstrukce. Rozhodně je větší zastoupení těch co jsou určení do vnitřního prostředí.

1.3 Struktura práce

Práce je uvedena třemi teoretickými kapitolami. Kapitola 2 se věnuje teorii řízení robota, což pokrývá popis řízení různých podvozků a motorů, a také PID regulaci. Kapitola 3 se věnuje senzorům a kapitola 4 dalším částem robota – napájením, nižším a vyšším řízením.

V kapitole 5 je rozebrána výroba prototypů a finálního robota a jeho řídící a senzorové jednotky založené na mikrokontroléru PIC. Další dvě kapitoly se věnují návrhu (kapitola 6) a implementaci (kapitola 7) veškerého softwaru potřebného pro běh robota a jeho vzdáleného ovládání.

Předposlední kapitola 8 představuje experimenty, které byly s hotovým robotem provedeny, a zhodnocuje přesnost řízení a schopnosti robota. Poslední kapitola 9 je závěr, který shrnuje výsledky této práce.

Doplňkem práce je ještě příloha A obsahující elektrická schémata řídích a senzorických desek řízených mikrokontrolérem PIC.

[3] http://www.husqvarna.com/cz/products/robotic-mowers/

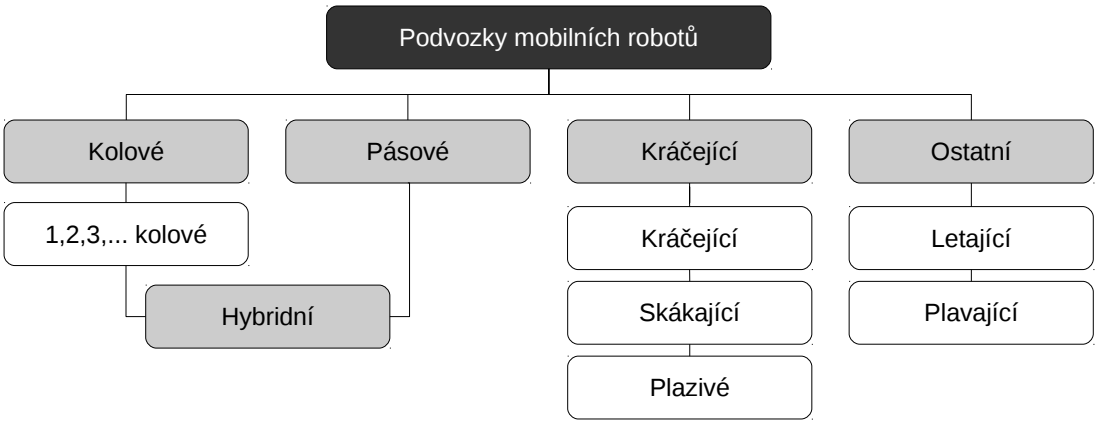
Kapitola 2

Teorie - řízení robota

Řízení robota znamená ovládání všech jeho pohonů. V oblasti mobilních robotů jsou to nejčastěji rotační elektromotory [1]. Těch je hned několik typů a každý se používá pro jiné potřeby. Jakmile se motor roztočí, je třeba jej regulovat - měnit počet jeho otáček a řídit směr jeho točení. Pohyb motorů je přenášen na podvozek a ten se pohybuje po zemi, v závislosti na svém kinematickém modelu. Výsledný pohyb robota je tedy kooperací několika subsystémů a tato kapitola se každému z nich věnuje.

2.1 Podvozky mobilních robotů

Podvozek mobilního robota může být konstruován různě. Obrázek 2.1 zachycuje základní kategorizaci podvozků [2]. Tato práce se zaměřuje hlavně na kolový podvozek, jelikož má dobrou průchodnost terénem a lze ho jednoduše a dostatečně přesně ovládat. Na rozdíl od pásového či kráčejícího podvozku, které mají sice ještě lepší průchodnost, je u něj ale podstatně snazší určování odometrie a jeho výroba je levnější a jednodušší.



Obrázek 2.1: Základní kategorizace podvozků mobilních robotů [2].

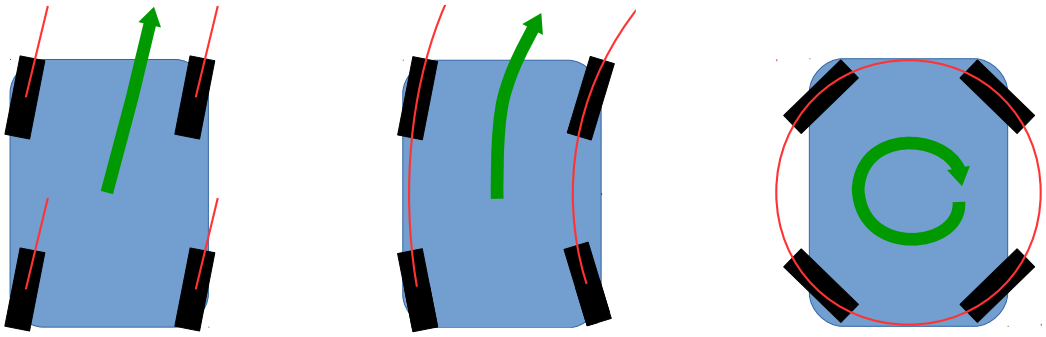
Ackermanův podvozek

Jedná se o častý typ podvozku, který je použit například i u osobních automobilů (v jeho čtyřkolové podobě) [1]. Používá se především pro roboty, kteří se pohybují vyšší rychlostí. Jeho hlavní nevýhodou je neschopnost otáčení na místě a tím snížená obratnost vozidla. Naopak výhodou je nižší energetická náročnost pohybu.

Pro Ackermanovo řízení je typické, že se přední řízená kola nenatáčí stejně, ale vnitřní kolo se natáčí více pro dosažení stejného středu otáčení s vnějším kolem, aby nedocházelo ke tření. Plánování trasy je kvůli středu otáčení umístěnému mimo vozidlo obtížnější.

Synchronní podvozek

Tento typ podvozku vyžaduje kola s alespoň dvěma stupni volnosti [3]. Jak již říká název, všechna kola se točí zároveň stejnou rychlosti. Výsledný směr jízdy je ovlivněn pouze natočením jednotlivých kol. Způsoby pohybu demostruje obrázek 2.2.



Obrázek 2.2: Kinematický model čtyřkolového synchronního podvozku.

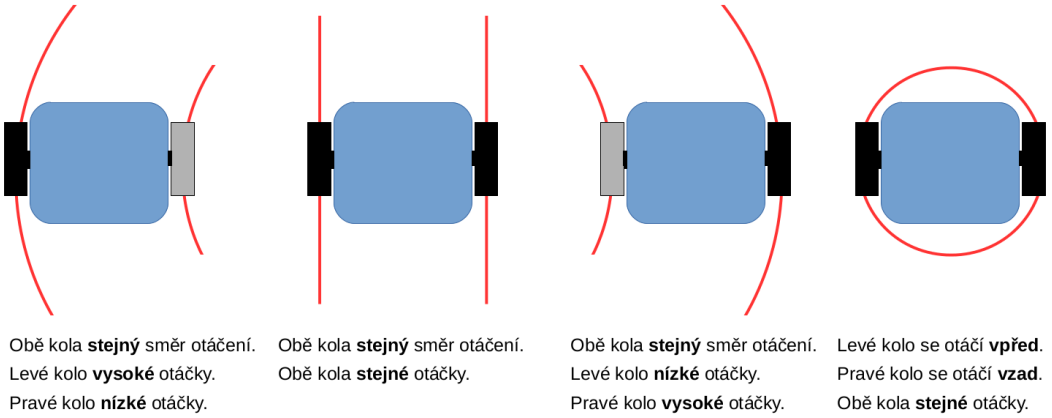
Nevýhodami synchronního podvozku jsou špatná prostupnost terénem, vyšší cena a náročnější výroba. Výhodami pak výborná manévrovatelnost, možnost otočení se na místě a vysoká stabilita.

Diferenciální podvozek

Konstrukce tohoto podvozku se zakládá na dvou nezávisle hnaných kolech [4]. Rozdíl jejich rychlosti otáčení určuje výsledný pohyb celého podvozku. Pokud se obě kola otáčejí stejně rychle, robot jede rovně. Pokud se točí stejnou rychlostí, ale opačným směrem, robot se otáčí kolem osy nápravy. Při různé rychlosti kol a stejném směru otáčení se robot pohybuje po kružnici, kdy vnitřní kolo je to s nižší rychlostí. Demonstrace způsobů pohybu je na obrázku 2.3.

Ideální diferenciální podvozek má kola umístěna tak, že střed otáčení je přímo mezi nimi. Nejen díky tomu se může robot otáčet na místě. Obecně má ale diferenciální podvozek velmi dobrou manévrovatelnost, i když není střed otáčení mezi koly. Dalšími výhodami jsou jeho jednoduchost konstrukce, robustnost, dobrá prostupnost terénem a nízká cena. Nevýhodou je jeho vyšší energetická náročnost pohybu způsobená zvýšeným třením při zatáčení.

Nejčastěji se používá u malých mobilních robotů určených do vnitřního prostředí. Není to však pravidlo a lze ho použít i pro venkovní roboty. Stačí použít větší kola s hrubším povrchem a dostatečně silné motory.



Obrázek 2.3: Způsoby pohybu diferenciálního podvozku.

2.2 Motory

Motory slouží k roztočení kol na podvozku, čímž ho uvedou do pohybu. Buď může být poháněno každé kolo zvlášť nebo může být použit jeden motor pro náhon více kol. Jak již bylo zmíněno výše, v oblasti mobilní robotiky se nejčastěji používají rotační elektromotory. Jejich nabídka je opravdu široká. Konkrétní volba se řídí podle požadavků na velikost, počet otáček, točivý moment, napětí, proud, cenu a další.

Stejnosměrné (DC)

Stejnosměrný motor s permanentním magnetem je často používaným typem motoru v mobilní robotice [1]. Je tomu tak díky jeho snadnému řízení, dostupnosti, široké nabídce a velice nízké ceně. To jsou zároveň jeho přednosti. Mezi nevýhody patří nízký točivý moment, složité polohové řízení a fakt, že motor je kvůli komutátoru zdrojem silného elektromagnetického rušení.

Stejnosměrné motory mají velmi vysoké pracovní otáčky. Pro některé, hlavně létající, roboty je to vhodné, ale pro kolové bývá nutností využít převodovku. Na rozdíl od motoru bývá převodovka drahé zařízení. Navíc výrazně zvyšuje hlučnost motoru. Zvýšením převodového poměru se zvyšuje moment motoru, což je pro kolové roboty žádoucí. Obzvláště pak pro, ty co se pohybují ve venkovním prostředí.

Většina stejnosměrných motorů má dva elektrické kontakty. Po připojení napětí se roztočí v jednom směru a při změně polarity se roztočí v opačném. Změnou napětí je možné regulovat počet otáček.

Krokové

Tento typ synchronního motoru je napájen stejnosměrnými impulzy [5]. Tyto impulzy ho posunují mezi jednotlivými přesně danými polohami – kroky. Takto nastavenou polohu si navíc dokáže držet. Kvůli těmto vlastnostem je široce využíván tam, kde je třeba přesného pohybu, v CNC obráběcích strojích, periferiích počítačů, robotických ramenech atd. Vzhledem k tomu, že se řídí impulzy, které lze počítat, není třeba další zpětné vazby pro určování odometrie.

U robotů se využívají právě kvůli snadnému a přesnému ovládání. Spíše ale u těch, co mají menší rozměry. Krokové motory se vyznačují mechanickou odolností a téměř bezúdržbovým provozem. Nevýhodami jsou vyšší cena, proti stejnosměrným motorům, a stálý odběr proudu, i když nedochází k otáčení motoru.

Střídavé (AC)

Rozdíl mezi stejnosměrným a střídavým motorem je ve způsobu generování jejich napětí [6]. U stejnosměrného je využito jednofázové, zatímco u střídavého se přenáší třífázové. Hlavní výhodou oproti stejnosměrnému je vyšší točivý moment, tichý provoz a nižší energetické nároky. U střídavých motorů se tak často nepoužívají převodovky, jelikož jsou samostatně dostatečně silné i pro provoz mobilního kolového robota. Zásadním problémem je ale podstatně obtížnější řízení. Z toho důvodu je často preferován stejnosměrný motor před střídavým.

Někdy bývají mezi střídavé motory řazeny i stejnosměrné bezkomutátorové motory. Díky absenci komutátoru je totiž jejich řízení podobné právě řízení střídavého motoru.

2.3 Regulace a řízení DC motorů

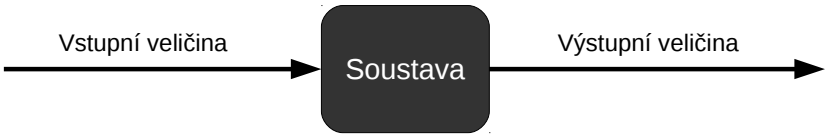
Prvním krokem řízení motoru je základní regulace jeho otáček. To je možné provádět hned několika způsoby: [7]

- **Změnou odporu** - realizuje se zapojením předřadného odporu do obvodu motoru. Nevýhodou je vznik trvalých ztrát přeměnou elektrické energie na teplo, a proto se téměř nevyužívá.
- **Změnou magnetického toku** - využití tohoto způsobu není možné u motorů s permanentními magnety, jelikož mají konstantní, tedy neregulovatelný magnetický tok.
- **Změnou napájecího napětí** - tento způsob má hned dvě varianty. První je přímé měnění napájecího napětí, což není z energetického hlediska vhodné. Druhá je dostatečně rychlé zapínání a vypínání napájecího napětí. Tento způsob je znám jako *PWM*, Pulse Width Modulation - pulzní šířková modulace. Aby bylo možné měnit i směr otáčení motoru, je navíc zapotřebí tzv. H-můstek, který se stará o změnu polarity napětí za využití čtyř spínacích tranzistorů. Tento způsob je realizovatelný pouze pro DC motory s permanentními magnety.

Pomocí výše uvedených způsobů je možné řídit otáčky motorů, které se však budou měnit v závislosti na jeho zatížení. Pokud se jedná o motor, u kterého není vyžadována přesnost a jeho zatížení není takové, aby způsobovalo zastavení motoru, lze využít řízení bez zpětné vazby. Pokud má být ale robot a jeho motory schopny samostatného pohybu, je třeba zavést zpětnou vazbu, a tím pádem dosáhnout požadovaných otáček při libovolném zatížení.

Řízení bez zpětné vazby

Tento přístup [8] nikterak nevyužívá informaci o aktuální rychlosti, jelikož k ní nemá přístup. Není proto třeba enkodéru, a tím pádem je řídící systém levnější. To je ale jediná výhoda tohoto přístupu.



Obrázek 2.4: Řízení bez zpětné vazby.

Pro řízení bez zpětné vazby postačuje znát pouze maximální rychlost motoru ($maxSpeed$), z které je možné vypočítat šířku PWM pulsu ($pulseWidth$) pro požadovanou rychlost ($reqSpeed$) podle následující rovnice 2.1. Hodnoty $minWidth$ a $maxWidth$ jsou minimální a maximální délka PWM pulsu.

$$pulseWidth = (minWidth + maxWidth)/2 + (reqSpeed/maxSpeed) \tag{2.1}$$

Zásadním nedostatkem rovnice je, že hodnota maximalání rychlosti motoru ($maxSpeed$) není konstantní. Je rozdílná pro každý motor a dokonce i směr jeho otáčení. Dále závisí na aktuálním napětí baterií, aktuálním zatížení motoru, okolní teplotě, aj. Tato hodnota je tedy pouhým odhadem.

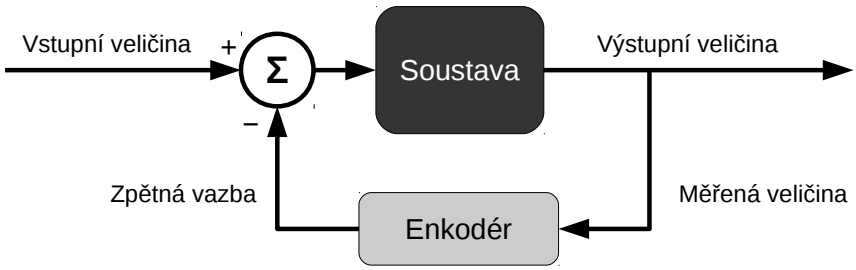
Dalším nedostatkem v rovnici je lineární závislost šířky pulsu na rychlosti. U reálného motoru je tato závislost povětšinu šířky nelineární. Ta část šířky, kde lze řídit otáčky lineárně, bývá velmi malá.

I v případě, že by se podařilo vytvořit křivku maximální rychlosti v závislosti na zátěži motoru, je zde spousta dalších faktorů, pro které by přestala platit, a proto je řízení bez zpětné vazby velmi omezeně použitelné.

Řízení se zpětnou vazbou

Aby bylo možné řídit motor s dostatečnou přesností, je třeba zavést zpětnou vazbu [8] - jeho skutečnou rychlost otáčení. U motoru ji zajišťují enkodéry, hallovy sondy, akcelerometry, gyroskopy, GPS senzory nebo jiné. V závislosti na ní se upravuje šířka pulzu a tím i rychlost otáčení motoru.

Následujcí obrázek 2.5 demonstruje jak řízení se zpětnou vazbou probíhá .



Obrázek 2.5: Řízení se zpětnou vazbou.

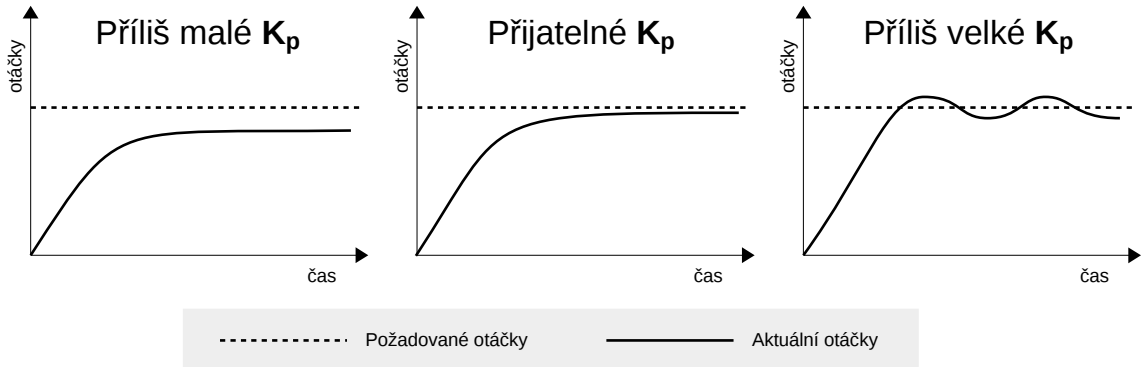
PID regulace

Zpětnovazební řízení pomocí PID regulátoru má jako vstup aktuální naměřenou chybu $e(t)$ a mění výstupní veličinu $y(t)$ tak, aby byla chyba v ideálním případě nulová [9]. Tento typ regulátoru pracuje s chybou ve třech složkách, z kterých plyne i jeho jméno – Proporcionální, Integrační a Derivační.

Výstup proporcionální složky P je přímo ovlivněn aktuální velikostí chyby $e(t)$, vstupující do systému, násobenou konstantou vlivu K_p.

$$P = K_p e(t) \tag{2.2}$$

Při použití samostatné proporcionální složky (P-regulátor) dochází často ke vzniku trvalé regulační odchylky nebo ke kmitání. Závislost výstupu P-regulátoru na nastavení konstanty K_p je zobrazena na obrázku 2.6.

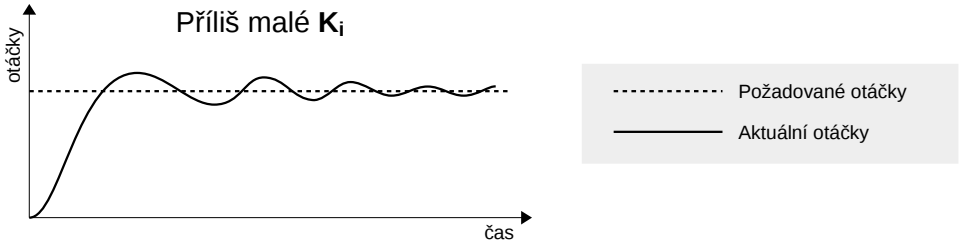


Obrázek 2.6: Vliv konstanty K_p na regulaci. [10]

Pro odstranění trvalé regulační odchylky se využívá integrační složky I. Ta provádí integraci chyby v průběhu času. Opět je zde konstanta K_i, ovlivňující její projev na výstupní hodnotu. Pro daný čas t a počáteční čas 0 je integrační složka vypočítána následující rovnicí, kde τ značí integrační proměnou.

$$I \ = K_i \int_0^t e(\tau)\, d\tau \tag{2.3}$$

Pokud budou uvažovány pouze složky P a I jedná se o PI-regulátor. Vliv konstanty K_i na tento typ regulace je ukázán na obrázku 2.7.



Obrázek 2.7: Vliv konstanty K_i na regulaci. [10]

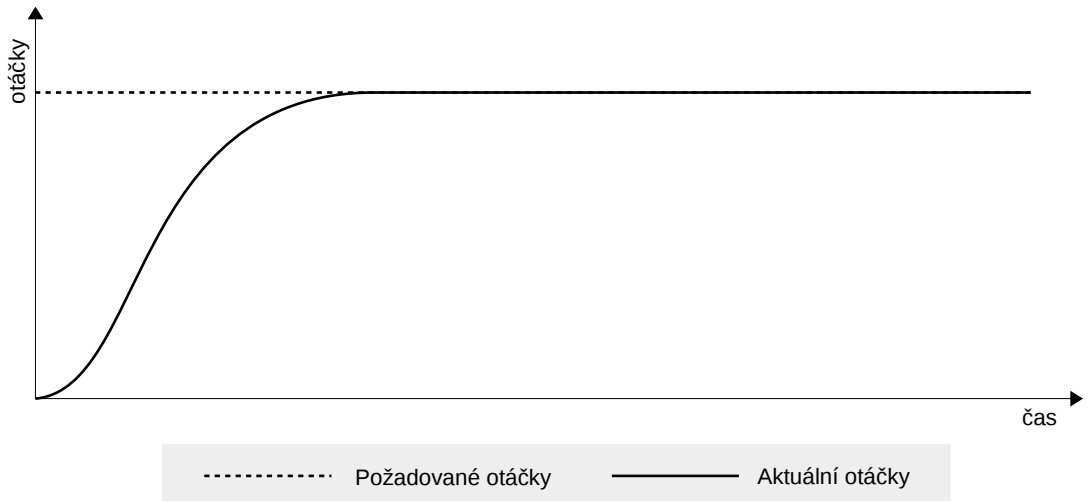
Poslední složka je derivační a používá se pro urychlení (nebo zpomalení) regulačního děje. Jejím smyslem je zmírnit kmitání pomocí předvídání budoucího chování systému na základě derivace chyby. Pokud je ale regulovaná veličina zatížena velkým šumem měření nebo systém obsahuje velké dopravní zpoždění, dochází k destabilizaci regulace. Pro nastavení vlivu derivační složky slouží konstanta K_d.

$$D = K_d \frac{de(t)}{dt} \tag{2.4}$$

Výsledný PID regulátor je tvořen součtem všech výše uvedených složek a jeho rovnice je následující:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt} \tag{2.5}$$

Pro každý regulovaný systém je třeba najít vhodné nastavení konstant K_p, K_i a K_d. Jejich prvotní odhad může být proveden například pomocí Ziegler-Nicholsovy metody [9], nicméně pro přesné nastavení je třeba upravit konstanty na základě experimentů. Některé projevy nastavení konstant jsou uvedeny výše. Výsledný požadovaný průběh regulace ukazuje následující obrázek 2.8.



Obrázek 2.8: Správný průběh PID regulace - správně zvolené konstanty K_p, K_i a K_d. [10]

Kapitola 3

Teorie - senzory

Zařízení, jímž robot vnímá okolní svět, se nazývá senzor a skupina těchto zařízení senzorický subsystém. Tento subsystém se řadí mezi jeden z nejdůležitějších, protože bez něj by se jednalo o jednoduchý automat bez schopnosti reakce.

Informace, které senzory poskytují, mohou sloužit k monitorování vlastního stavu robota, k rozpoznávání okolí, překážek v něm, atp. Pro tyto a další činnosti existuje široká škála zařízení [11], od jednoduchých tlačítek až po senzory, které poskytují obrovské množství dat o vnějším prostředí (například laserové senzory).

Jednotlivé senzory se mezi sebou liší citlivostí, rozsahem, rozlišením, spolehlivostí, energetickými nároky, cenou a mnoha dalšími. I dva zdánlivě stejné senzory proto mohou být naprosto rozdílné. Vybrat pro robota správné senzory je tedy netriviální úkol a ovlivňuje jeho hranice schopností. Výběr navíc může být komplikován vzájemným rušením senzorů.

Senzory dělíme na pasivní a aktivní. Pasivní senzory kvantizují energii, která do nich vstupuje, zatímco aktivní emitují záření do prostředí a měří jeho reakci. Mezi pasivní řadíme například hallovy sondy, akcelerometry, gyroskopy, kompasy a kamery. A mezi aktivní třeba laserové dálkoměry a radary.

3.1 Enkodéry

Hlavním smyslem enkodérů u robota je měření otáček kol [12]. Existují 2 typy:

- **Inkrementální** - u tohoto typu je detekován jeden čí více impulzů během jedné otáčky. Tyto impulsy jsou nerozlišitelné.
- **Absolutní** - jedna otáčka kola je rozdělena do několika částí, kde každá má svůj unikátní rozpoznávací kód. Enkodér přijímá tyto kódy a je schopen z nich určit aktuální polohu kola - v které z částí se nachází.

Princip enkodéru je jednoduchý a samotné měření může probíhat různými způsoby:

- **Hallova sonda** - využívá Hallova jevu, pomocí kterého měří magnetické pole. Tato sonda se používá tak, že se buď měří změny magnetického pole přímo v motoru a nebo se na kolo/osu připevní magnety, které jsou detekovány.
- **Fototranzistory, fotodiody** - měří změnu intenzity záření viditelného nebo neviditelného spektra (IR). Jako protikus optočidla je disk s průhlednými okénky, odrazivými ploškami nebo černo-bílými plochami.

- **Potenciometr** - měří změnu elektrického odporu. Pomocí A/D převodníku se snadno určí poloha. Typicky využíváno v levných modelářských servech.

Výstupní signál enkodéru má obdélníkový průběh. Rozlišení směru otáčení se řeší přidáním druhého segmentu s detekovanými body posunutými vůči detekovatelným bodům prvního segmentu. Signál je poté fázově posunut, což značí opačný směr otáčení.

Tento senzor se využívá jako zpětná vazba pro regulaci otáček motorů. Zároveň je možné ho použít pro měření ujeté vzdálenosti - odometrie. Zde ale dochází k chybě při prokluzu kol a chyba s časem výrazně narůstá, protože se akumuluje.

3.2 Kompasy

Směr vozidla patří mezi velmi důležité parametry v navigaci a u diferenciálních podvozků ho lze využít i pro účely řízení. Kompas [13] poskytuje absolutní měření, což významně pomáhá korigovat senzory s akumulační chybou.

Tento senzor měří magnetické pole v jeho okolí. Jeho užitečnost vychází z faktu, že země má své magnetické pole. Toto pole je sice silně ovlivňováno nerůznějšími vlivy, jako například střídáním dne a noci, bouřemi, elektricky nabitými slunečními částicemi, ale přesto je měřitelné a poměrně spolehlivé. Větší problém tvoří rušení způsobené elektromotory, kovovými strukturami nebo elektrickým vedení v okolí robota atp. Je proto důležité správné umístění kompasu na robotovi.

V dnešní době jsou velmi využívány tzv. 3D kompasy. Měří magnetické pole ve třech osách a díky tomu je možné odlišit magnetický sever a jih. U 2-osých to možné není.

3.3 Akcelerometry

Patří mezi tzv. inerciální senzory [14] (měření probíhá vůči nějakému inerciálnímu prostoru) a měří rozdíl mezi kynematickým zrychlením a gravitačním zrychlením . Většinou měření probíhá pro tři na sebe vzájemně kolmé osy (3D). Z těchto údajů je možné určovat polohu tělesa v prostoru. Na rozdíl od enkodérů netrpí chybou při protáčení kol.

Často je vyráběn MEMS technologií (Micro-Electro-Mechanical Systems - mikro-elektromechanický systém). To znamená že jsou tyto senzory miniaturní a energeticky nenáročné. Akcelerometry jsou navíc velice levné, protože se dnes masově vyrábějí pro použití ve spotřební elektronice.

Při práci s akcelerometrem je důležitá správné zpracování získávaných dat, jelikož to bývají zařízení s velmi vysokou citlivostí. Data je proto nutné filtrovat. Teprvé po filtraci je možné je využít pro, již zmíněné, určování polohy nebo pro detekci volného pádu, nárazu, smyku, náklonu aj.

3.4 Gyroskopy

Další inerciální senzor. Snímá úhlovou rychlost, z níž lze pomocí integrace dle času zjistit natočení. Dnes nejčastěji MEMS zařízení využívající principu Coriolisovy síly a stejně jako akcelerometr se dnes nejčastěji vyrábí v 3-osé variantě a bývá velice levný.

Pro snazší využitelnost dat z tohoto senzoru je výhodné umístit ho do středu otáčení robota. Kromě využití jako snímače natočení se používá pro monitorování a určování rovnovážné polohy robota.

3.5 IMU

IMU je anglická zkratka Inertial Measurement Unit - což znamená inerciální měřící jednotka. Jak již říká název, je složená z inerciálních senzorů, konkrétně 3-osého gyroskopu a 3-osého akcelerometru. Občas bývá doplněna pro zlepšení funkčnosti kompasem. Jedná se tedy pouze o kombinaci více jednoduchých senzorů dohromady.

Tato jednotka je obzvláště užitečná pro létající roboty, protože zajišťuje přesnou odometrií. Lze ji ale použít i pro pozemní roboty. Využívá se jako záložní jednotka GPS navigace. Když není signál GPS (například v tunelu), používá se pro odhad pozice na mapě právě IMU. Při časově delším běhu ale dochází k výrazné akumulační chybě.

3.6 Sonary

Sonary slouží k měření vzdálenosti překážek a jedná se o velmi často využívaný senzor kvůli své nízké hmotnosti, velikosti i ceně. Pracuje na principu měření doby letu vyslaného signálu, takže nevyžaduje přímý kontakt s překážkou. Signál je u sonaru představován ultrazvukovými vlnami a díky znalosti rychlosti šíření zvuku je možné vzdálenost snadno dopočítat. Vzhledem k použití ultrazvuku je ale omezen prostorový rozsah a dosah. Navíc senzor není schopen určit přesnou pozici překážky, ale pouze to, jestli v dané kuželové výseči je (nebo není) a v jaké vzdálenosti se nachází. Také nízká přesnost a malá rychlost nepatří u sonarů mezi žádoucí vlastnosti.

Využití nachází při detekci překážek v jízdě robota nebo při mapování prostředí a lokalizaci v něm. Pro tento učel se většinou využívá několika sonarů, aby pokryly širší okolí robota. Zároveň je tak možné přibližně určit i přesnější pozici nebo tvar překážky (v závislosti na tom, který sonar vrací jakou hodnotu).

3.7 Kamery

Dnes jeden z nejlevnějších senzorů, který se díky zvyšujícímu se výkonu počítačů dostává mezi nejpoužívanější a nejužitečnější senzory. Jedním z důvodů je i fakt, že funguje na podobném principu jako zrak, což je lidskému chápání blízké. Dalšími výhodami jsou nízká hmotnost, malá velikost, široká nabídka, rozlišovací schopnost. Nevýhodou je velké množství dat a jejich vysoká náročnost zpracování.

Kameru čí více kamer je možné použít pro širokou škálu účelů: odometrii, detekci překážek, lokalizaci, mapování a nepřeberné množství dalších. Nejčastěji jsou to ale úkony související s vyšším řízením. Nebývají proto součástí základní řídící jednotky, ale spíše se připojují k řídícímu počítači.

Kapitola 4

Teorie - další části a systémy robota

Mimo podvozku, pohonného subsystému a senzorického subsystému se robot skládá z dalších částí a subsystémů/systémů. Tato kapitola se jim krátce věnuje.

4.1 Napájení

Mobilní roboti nejčastěji bývají napájeni akumulátory. Existují i takoví, kteří jsou napájeni přímo z elektrické rozvodné sítě pomocí kabelu, ale jejich akční radius je tím velice omezen. Na druhou stranu napájení akumulátorem je omezeno nutností nabíjet ho. Dnešní akumulátory však již dosahují vysokých kapacit za rozumnou cenu.

Napájení robota bývá rozděleno na dvě části: napájení elektroniky a pohonu. Odběr energie pohonu totiž bývá řádově větší. Navíc při prudkém poklesu napájecího napětí pohonu nedojde ke kolapsu řídící elektroniky a ta může systém bezpečně ukončit.

Aktuálně nejlepší v poměru cena/výkon jsou **Li-Ion** a **Li-Pol** články [15]. Nemají paměťový efekt, jejich samovolné vybíjení je zanedbatelně malé, mají nízkou hmotnost, relativně vysokou kapacitu, dobrý teplotní rozsah fungování a jsou schopny dodávat vysoké proudy. Nominální napětí jednoho článku je 3,7 V.

U mobilních robotů je zvykem, že se nabíjí v dobíjecí stanici (připojené k elektrické síti), ke které je schopen se autonomně přiblížit a připojit ve chvíli, kdy mu dochází baterie. Tím je zajištěno nepřetržité fungování robota.

4.2 Řídící systém s mikrokontrolérem

Tento systém zajišťuje nižší řízení [16] a je připojen k systému vyššího řízení. Většinou se jedná o jednu desku, na kterou jsou připojeny senzory a pohony, které jsou přivedeny na vstupy a výstupy mikrokontroléru/mikropočítače. U menších jednotek pro mobilní roboty jde většinou o čipy firem STM, Atmel, Microchip a podobných. U takových jsou řídící systémy vytvářeny zcela na míru k robotovi. To vyžaduje přípravu tištěného spoje, jeho osazení a zprovoznění. Výhodou těchto systémů je jejich rychlost a naprostá přizpůsobenost konkrétnímu robotovi

Rozšířenou a často používanou alternativou jsou již hotové univerzální desky a malé počítače. Mezi takové se řadí třeba rodina desek Arduino. Poskytují programovatelnou platformu, ke které je možné připojit pomocí konektorů či nepájivého pole senzory, pohony

a další součástky. Taková platforma je vysoce univerzální. Toto řešení může být dražší nebo i levnější než řídící jednotky na míru (v závislosti na komplexnosti). Nevýhodami jsou větší velikost a nižší robustnost.

Poslední variantou je řízení běžným počítačem (PC) nebo některým z mini-počítačů, který zajišťuje vyšší i nižší řízení. Počítač na to musí být připraven (konektory) a jeho operační systém musí poskytovat přístup na dostatečně nízké úrovní. Senzory jsou připojovány např. přes I2C či SPI, což nebývá pro kancelářské počítače běžně dostupné rozhraní. Další problém bývá s přesným časováním a periferiemi, jako jsou časovače, PWM generátory, A/D převodníky a další. Mezi zástupce těchto počítačů lze zařadit Raspberry Pi nebo BeagleBone.

4.3 Systém vyššího řízení

V případě autonomního robotu se stará o lokalizaci robota a zjišťování jeho stavu. Na základě těchto informací plánuje další pohyb v prostoru tak, aby dosáhl požadovaného cíle. Bere přitom v úvahu překážky, kterým se snaží vyhnout. Stará se o včasné navedení robota do nabíjecí stanice a po jeho nabití zpět na jeho původní kurz. V případě teleoperovaného robota zajišťuje převod pokynů operátora do jednotky nižšího řízení a naopak odesílání informací ze senzorů operátorovi. Mezi operátorem a jednotkou nižšího řízení může být klidně i více těchto systémů spojených fyzicky či bezdrátově a ty musí spolupracovat. Jedním ze systému vyššího řízení je například ROS, který bude popsán níže.

Pro běh systému vyššího řízení se používají běžné počítače (PC). V závislosti na požadované velikosti je možné použít například minipočítač Raspberry Pi, notebook (různé velikosti) či výkonné stolní počítače. S vyšším výkonem se ale značně zvyšují energetické nároky.

ROS - Robot Operating System

Robotický meta-operační systém poskytuje knihovny a nástroje sloužící k usnadnění práce při tvorbě software robotů a je distribuován pod otevřenou licencí BSD[1]. Systém se soustředí na abstrakci hardwarové vrstvy, podporu ovladačů různých zařízení (motorů, senzorů, ...), vizualizaci rozličných dat, komunikaci mezi podsystémy, implementaci často využívaných funkcionalit a další. Mezi důležité vlastnosti patří ještě vlastní prostředí pro sestavování a běh programů, které je multiplatformní (cíleno především na UNIXové systémy) [17].

Struktura ROS

Důležitou vlastností ROSu je jeho modularita. Jednotlivé moduly se nazývají **uzly (ang. node)** a jsou to libovolné programy, které vzájemně komunikují pomocí zpráv a služeb, které definuje ROS. Tyto programy tedy nejsou vázány programovacím jazykem. V době psaní této práce jsou však plně podporovány jen jazyky C++ a Python (další jsou ve vývoji). Termín uzel vychází z vizualizace běhu ROS systémů, kdy jsou jednotlivé komunikující programy vykresleny jako propojené uzly grafu.

[1] http://www.linfo.org/bsdlicense.html

Balíkovací systém ROS

Základní organizace systému ROS spočívá v hierarchickém dělení na menší celky. Jednotka tohoto systému je **balík (ang. package)**. Balík obsahuje jeden či více uzlů, knihovny, ovladače a další související komponenty. Jeho nedílnou součástí je seznam závislostí na dalších balících, ale pro dobrý návrh balíku platí, že by závislostí nemělo být mnoho, aby bylo možné balík použít v rozličných situacích. Myšlenkou balíkovacího systému je totiž rozdělení vývoje velmi složitého a komplikovaného projektu na menší kousky, které poté mohou implementovat různé týmy nezávisle na sobě.

Operační systém Ubuntu, který je hlavním operačním systémem, nad kterým je ROS vyvíjen, obsahuje v repozitářích mnoho již implementovaných balíků, které je možné volně využívat. To umožňuje velmi snadno zprovoznit robota a jeho systém vyššího řízení.

Sestavovací systém Catkin

Od ROS verze Fuerte je místo sestavovacího systému ROS build použit systém Catkin. Jedná se o kolekci maker do multiplatformního sestavovacího systému CMake, který slouží pro sestavení ROS balíků. Pomocí něj jsou rozgenerovány závislosti, definice zpráv, služeb atd.

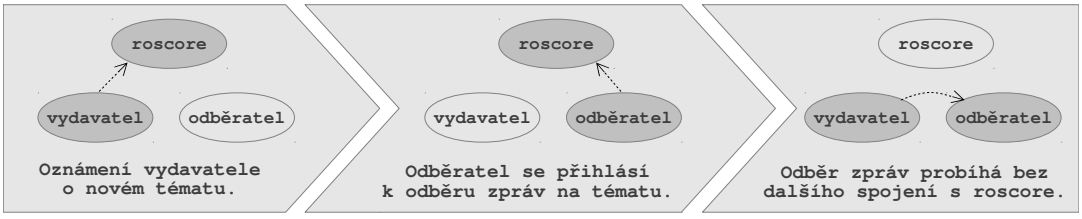
Komunikace v ROS

O zprostředkování komunikace se stará vždy `roscore`. Je to centrální uzel, který musí být spuštěn před ostatními uzly. Po spuštění udržuje přehled o ostatních uzlech a umožňuje jim nalezení protějšího uzlu ke komunikaci. Další komunikace pak probíhá bez nutnosti komunikace s `roscore`.

ROS implementuje dva způsoby komunikace:

1. **Asynchronní - pomocí zpráv** - sdíleným kanálem pro tuto komunikaci jsou **témata (ang. topics)**, přes která jsou distribuovány **zprávy (ang. messages)**. Uzel, který odesílá zprávy na určité téma, je nazýván **vydavatel (ang. publisher)** a jiný uzel, který na něm zprávy přijímá, je **odběratel (ang. subscriber)**. Pro jedno téma platí, že na něj může posílat zprávy více vydavatelů a ty pak z něj může přijímat zaráz více odběratelů. Obrázek 4.1 ilustruje inicializaci asynchronní komunikace.

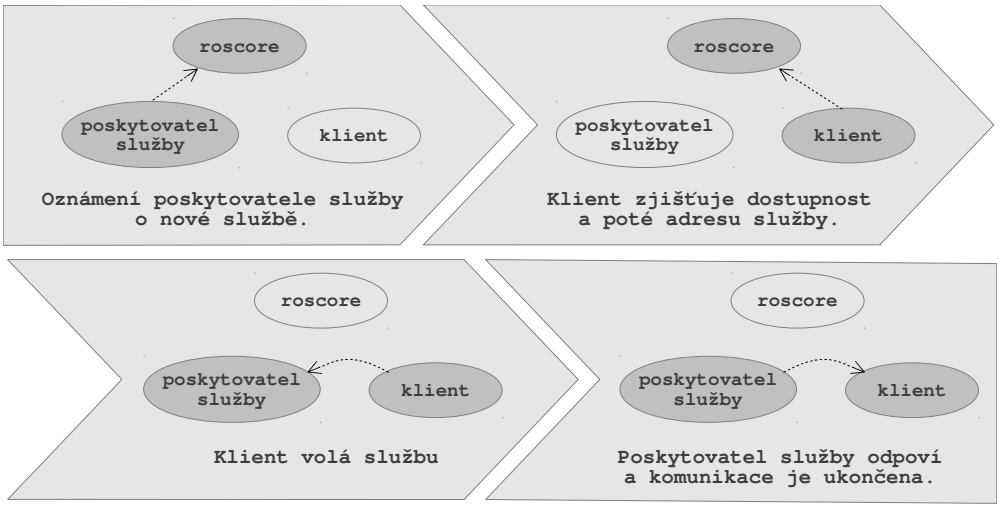
   Komunikace mezi uzly a `roscore` a poté mezi vydavatelem a odběratelem je realizována pomocí protokolu vycházejícího z XML-RPC.



Obrázek 4.1: Schéma asynchronní komunikace pomocí zpráv [17].

2. **Synchronní - služby a klienti - služba (ang. service)** je implementována v uzlu a ten oznámí `roscore`, že je k dispozici. **Klient (ang. client)** pak může od `roscore`

zjistit adresu služby a volat ji přímo (s požadovanými parametry). Uzel poskytující službu na toto volání odpovídá přímo klientskému uzlu a spojení je ukončeno. Toto chování je zachyceno na obrázku 4.2



Obrázek 4.2: Schéma synchronní komunikace mezi poskytovatelem služby a klientem [17].

Pro definici zpráv a služeb zavádí ROS jednoduchý textový formát, který zajišťuje nezávislost na programovacím jazyce (soubory v tomto textovém formátu se při sestavení pomocí `catkin_make` rozgenerují pro všechny jazyky).

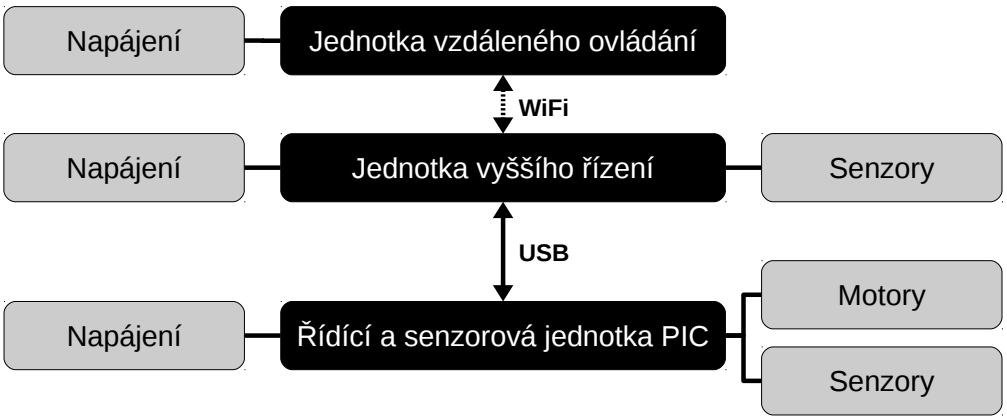
Všechny zprávy i služby jsou silně typované. Lze využít několik předdefinovaných typů nebo definovat vlastní, které mohou být následně použity v definici komplexnějších zpráv. Všechny strany, které spolu chtějí komunikovat, musí mít stejné definice zpráv a služeb, jinak je komunikace neúspěšná. Shodnost definic je ověřena z MD5 otisků definičních souborů zpráv/služeb.

Kapitola 5

Konstrukce robota

Tato kapitola se zabývá návrhem a výrobou robota vhodného pro otestování řídící a senzorové jednotky PIC, který bude možné vzdáleně ovládat. Základní požadavky, které musí splňovat jsou se sepsány v úvodní kapitole v sekci 1.1.

5.1 Základní schéma systémů robota



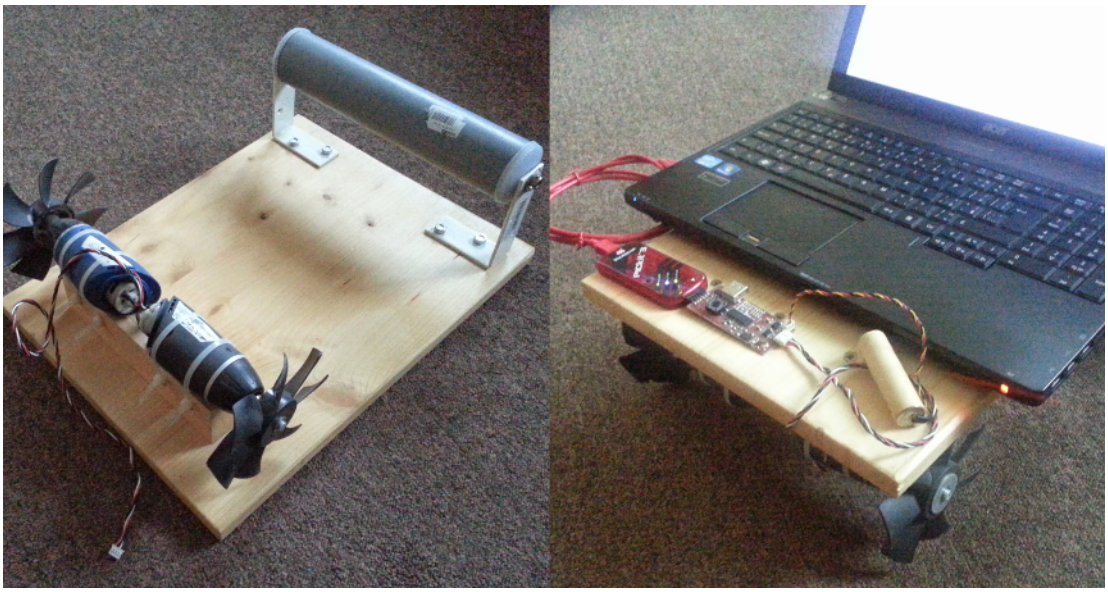
Obrázek 5.1: Schéma robota a jeho systémů.

Schéma ukazuje, že robot bude složen ze tří samostatných, vzájemně komunikujících prvků. Jednotka založená na čipu PIC se bude starat o nižší řízení a vyčítání dat z jednoduchých senzorů (kompas, akcelerometr, gyroskop, enkodéry). Počítač zodpovědný za vyšší řízení s ní bude fyzicky propojen a navíc bude obsahovat náročnější senzory (prozatím jen kameru). A vzdálený ovladač, který k němu bude připojen bezdrátově.

5.2 První prototyp

První konstrukce robota byla vytvořena na základě znalostí získaných z nastudovaných existujících řešení uvedených v kapitole 1.2. Tento prototyp byl snahou získat počáteční testovací platformu, která bude dále rozšiřována a upravována.

Podvozek tohoto prototypu byl zvolen jako diferenciálně řízený, protože bylo snadné ho vytvořit a zároveň bylo možné ověřit jeho vhodnost pro určený záměr, vytvořit robotickou sekačku trávníku. K ploché dřevěné desce, která slouží jako základ robota, byly připevněny dva malé aku-šroubováky *CMI 3,6 V Li*. Tyto šroubováky nebylo nutné nijak výrazně upravovat, jelikož obsahují motor, převodovku a nástavec na bity, který lze využít pro uchycení kol. Odstraněny byly pouze rukojeti. Baterie, které byly původně v rukojetích, byly vyvedeny přes konektor spolu s vývody k motorům. Zbytek vrtačky byl upevněn pomocí stahovacích pásků k dřevěnému hranolu připevněnému k základové desce. Jako přední podpůrné kolo nebylo zvoleno otáčecí kolečko, jak bývá u diferenciálních podvozků zvykem, nýbrž válec, který je vhodnou simulací sekacího vřetena. Kola byla vyrobena, s ohledem na záměr provozovat robota ve venkovním prostředí, z větráků, které by neměly při jízdě prokluzovat. Obrázek 5.2 ukazuje popsanou konstrukci.



Obrázek 5.2: První prototyp podvozku robota.

Kromě podvozku je důležitou součástí prvního prototypu i první verze řídící a senzorové jednotky založené na mikrokontroléru PIC. Tato jednotka musí disponovat konektorem pro programátor, přes který je nahráván firmware a dále USB konektorem pro napojení na jednotku vyššího řízení, a konektorem pro připojení baterií a motorů. Navíc pro lepší ladění firmware je součástí jednotky i 8 led diod, které jsou vhodné pro zobrazování binárních dat (zobrazení jednoho byte) či jinou signalizaci. Detailní schéma elektrického zapojení této jednotky se nachází v příloze na obrázku A.1. Jednotlivé použité integrované obvody jsou následující:

Kompas HMC5883L

Magneto-rezistivní čidlo [18], které komunikuje přes protokol I^2C. Jeho základní parametry jsou:

- 3 osy měření (3D)
- nízké operační napětí (2,16 – 3.6V) a malý odběr proudu (100μA)

- rychlá výstupní frekvence dat - maximálně 160Hz
- přesnost mezi 1° až 2°
- dva módy měření - průběžné a měření jedné hodnoty

Akcelerometr + gyroskop LSM330DL

Jedná se o dva senzory v jednom pouzdře [19]. Oba dokáží komunikovat přes I^2C (každý má svůj vlastní identifikátor). Základní parametry:

- měření lineárního a úhlového zrychlení ve třech osách (3D)
- volitelný rozsah přesnosti akcelerometru ±2g/±4g/±8g/±16g
- volitelný rozsah přesnosti gyroskopu ±250/±500/±2000 dps
- integrovaný teploměr pro korekce měření
- integrované filtry

Obvod pro řízení motorů DRV8833

Slouží pro řízení stejnosměrného motoru pomocí duálního H-můstku [20]. Tato součástka přijímá jako jeden ze vstupů PWM, které řídí výkon motoru. Dokáže detekovat přehřátí obvodu a příliš vysoký proud. Provozní rozsah napětí je 2,7 – 10,8 V. Pro každý motor je nutné použít jeden tento integrovaný obvod.

Mikrokontrolér PIC24FJ64GB002

Jako řídící mikrokontrolér byl zvolen PIC24FJ64GB002 [21]. Jde o 28-pinový 16-bitový RISC kontrolér založený na Harvardské architektuře (paměť programu oddělená od dat), obsahující širokou paletu periferií. Jeho nejdůležitější parametry jsou:

- operační frekvence až 32 MHz
- velikost flash paměti 64 KB
- integrace technologie USB On-The-Go (OTG)
- jednotka reálného času (RTCC)
- patnáct přemapovatelných pinů, pět 16-bitových časovačů, pět PWM výstupů, devět kanálů 10-bitových A/D převodníků, tři komparátory, dva UART moduly
- podporované komunikační protokoly: I^2C, SPI

5.3 Druhý prototyp

Při experimentech s první prototypem se projevilo několik více či méně závažných problémů. Asi největším nedostatkem byly nedostatečně výkonné motory. Jejich kroutící moment byl vhodný pouze pro pohyb po hladkém vnitřním povrhu. Na zatravněné venkovní ploše motory nedokázaly překonat odpor prostředí. Tento problém ještě více umocnilo použití válce místo předního kolečka a větráků jako kol. Díky větrákům se zvýšil odpor kol a také se snížila přesnost pohybu, jelikož minimální pohyb „kola“ je určen roztečí mezi dvěma lopatkami. Pohyb byl kvůli tomu velmi trhaný.

V rámci druhého prototypu tedy byly větráky nahrazeny běžnými koly s větším poloměrem (7,5 cm) a hrubým gumovým vzorkem. Stejně tak byly nahrazeny původní akušroubováky za výkonější – *CMI 10,8 V Li*. Motor a převodovka, které obsahuje tento šroubovák mají kroutící moment 18 Nm, což je více než nabízí běžně dostupné motory s převodovkami v této cenové a velikostní skupině. Výsledné maximální otáčky soustavy (motor + převodovka) jsou 520 ot/min. Motor *RS-550*, který je součástí aku-šroubováku, má 19300 ot/min při nulové zátěži a převodovka má převodový poměr 36:1. Provozní napětí se pohybuje v rozmezí 6 - 14,4 V a odběr proudu bez zátěže je 1,4 A.

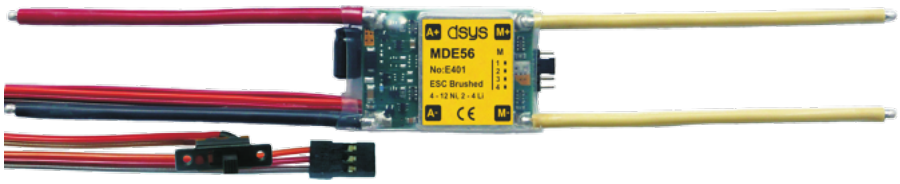
Uchycení kol je řešeno přes sklíčidla, kterými jsou použité aku-šroubováky vybaveny. Toto uchycení je dostatečně pevné a snadno odnímatelné. Nejen díky použití sklíčidla jsou však rozměry soustavy motor, převodovka a sklíčidlo o mnoho větší. Na obrázku 5.3 je dobře vidět, že původní desku tyto nové pohony značně přesahují. Proti původním použitým akušroubovákům byl u těchto nových odstraněn ochranný kryt, který zbytečně zvětšoval již tak dost velké rozměry.

Obrázek 5.3: Fotky druhého prototypu robota.

Při použití výkonnějších motorů je nutné použít baterii s vyšším napětím a kapacitou. Vhodné jsou ty z aku-šroubováku, jelikož byly pro použité motory přímo dimenzovány a navíc obsahují ochrannou a nabíjecí elektroniku, kterou lze snadno využít. Dalším, a mnohem

větším problémem je vyšší proud, který motor odebírá a který prochází přes integrovaný obvod DRV8833. Tento obvod je schopný pracovat s proudem do 4 A, což je nedostatečné, jelikož trvalý odběr proudu u jednoho motoru se pohybude kolem 15 A a ve špičkách až 20 A. Obvod DRV8833 proto v druhém prototypu nahradily dva modelářské stejnosměrné regulátory **MDE32** od firmy *dsys* (na obrázku 5.4). Jeho parametry jsou následující [22]:

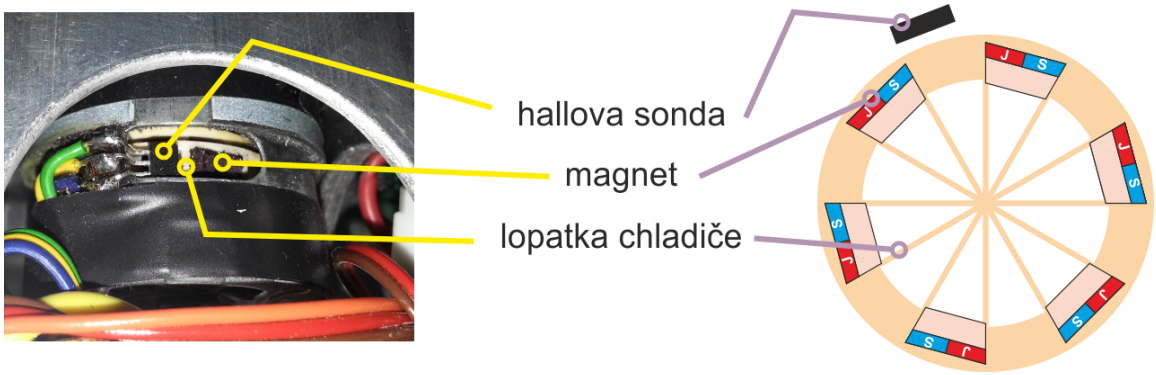
- plně programovatelný pomocí zasíláním PWM impulsů různé délky
- velmi jemné řízení a plynulý rozběh motoru, nastavitelný volnoběh
- tepelná, proudová, napájecí, podpěťová, přepěťová a signálová ochrana regulátoru
- akustická a optická signalizace stavu zařízení
- počet článků 4 - 12Ni-xx, 2 - 4Li-Pol / Li-Ion / A123, Pb
- maximální trvalý proud 32 A a špičkový 100 A
- provozní teplota -10 až 40 $°C$



Obrázek 5.4: Modelářský stejnosměrný regulátor MDE32.

Tento prototyp byl dále rozšířen o enkodéry. Protože použité motory nemají vzadu prodlouženou osu a není možné k nim upevnit standardně dodávené enkodéry, bylo nutné vyrobit enkodér vlastní. K tomu posloužil chladič, který je součástí osy motoru. Mezi lopatky chladiče se vlepí kusy linolea, na který se přilepí magnety (viz obrázek 5.5). Tyto magnety musí být dostatečně malé a mít právě jeden přechod polarity magnetického pólu. Tento přechod je detekován **hallovou sondou TLE4906L**. Hlavní předností tohoto konkrétního typu sondy je vysoká citlivost a stabilita měření [23]. Operační teplota se pohybuje v rozmezí -40°C až +150°C, což znamená že, není problém umístit ji k motoru. Takto je snímáno šest magnetů, což garantuje 6 impulsů na jednu otáčku motoru. Teoreticky je možné zvýšit počet magnetů na 12, nicméně poté by nezůstaly u chladiče motoru žádné mezery na odvětrávání. Šest impulsů na otáčku znamená jeden impuls na pohyb robota o 2 mm, což je pro zvolený záměr dostatečné. Nevýhodou takto vyrobeného enkodéru je nemožnost detekovat směr otáčení. To je nutné řešit v rámci firmware.

Kromě náhrady integrovaných obvodů DRV8833 za modelářské regulátory a přidání enkodérů bylo ještě z baterie přivedeno napětí na A/D převodník mikrokontroléru (přes napěťovou děličku) za účelem monitorování stavu baterie. Také bylo řešeno nabíjení baterií pomocí elektroniky převzaté z aku-šroubováku. Všechny tyto změny vedly k přepracování prvního prototypu řídící a senzorové jednotky založené na PIC mikrokontroléru. Elektrické schéma tohoto upraveného zapojení je v příloze A.2.



Obrázek 5.5: Vzhled a schéma vyrobeného enkodéru.

5.4 Výsledný robot

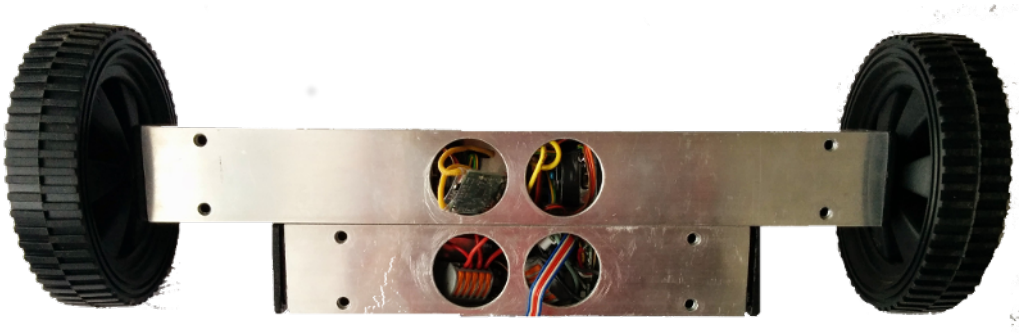
Pohonný subsystém druhého prototypu při testovacích jízdách prokázal schopnost pohybu v náročných podmínkách ve venkovních prostorech, takže nebylo třeba jej nijak měnit. Bezproblémový byl i napájecí subsystém a stejně tak i řídící a senzorová jednotka. Výsledný robot proto všechny tyto části zachovává a změněna je pouze konstrukce robota, která byla u prototypů řešena pouze provizorně. Z provedených jízd vyplynulo následující:

- těžiště robota by mělo být co nejníže, aby při jízdě do kopce či z kopce nedošlo k převrácení robota
- poháněná kola by měla být co nejvíce zatížena, aby docházelo k co nejmenšímu prokluzu kol
- váha robota musí být větší než u prototypů, čímž se sníží prokluz kol a zároveň se zlepší schopnost robota překonávat vyšší stoupání (a klesání)
- řídící a senzorová jednotka musí být umístěna v co největší vzdálenosti od motorů, aby nedocházelo k rušení senzorů

V souladu s těmito znalostmi se jako materiál pro základovou desku výsledného robota použila dřevotříska, která je o něco těžší než smrkové dřevo (použité u prototypů). Pro uložení motorů (včetně sklíčidla a převodovky), baterií, regulátorů a další elektroniky je využito dvou kovových jeklů. V delším z nich jsou umístěny motorové komplety a v kratším baterie a elektronika. Do stěn jeklů jsou vyvrtány otvory na přepínače a nabíjecí konektor, takže není třeba elektroniku vytahovat a je zároveň chráněna. Pro přístup do ní slouží čtyři kruhové otvory. Na následujícím obrázku 5.6 jsou zobrazeny osazené jekly včetně kol.

Ze strany těchto otvorů se přímo připevňuje základová deska pomocí šroubů. Výška takto sestaveného kompletu je nižší než horní bod kola. To znamená, že těžiště leží pod úrovní tohoto bodu, což je dostatečně nízko, a robot je proto velmi stabilní. Vepředu je ještě upravený válec, simulující sekací vřeteno, který je širší než původní na prototypech a stabilitu robota ještě zvyšuje.

Nahoře, v přední části základové desky, je umístěna plastová krabička, do níž je vyvedena řídící a senzorová jednotka založená na mikrokontroléru PIC. Dráty vedoucí z této jednotky do jeklů s elektronikou slouží k přenosu signálu enkodérů (2x), napětí baterií a signálu PWM řídícího regulátory (2x). Dále je z krabičky vyvedeno:



Obrázek 5.6: Základ podvozku tvořený jekly, obsahující motorové sestavy a elektroniku.

- konektor USB sloužící k propojení jednotky nižšího a vyššího řízení)
- konektor pro programátor mikrokontroléru
- vývod I2C a napájení pro sonary

Před touto plastovu krabičkou je umístěn sonar, pokrývající okolí před robotem. Jedná se o rozšíření proti prototypům. Záměrně je umístěn pouze v čelním směru, jelikož se robot bude pohybovat pouze vpřed (kvůli sekacímu vřetenu). Pokud chce robot couvat, stačí se otočit na místě a jet vpřed – to je umožněno díky diferenciálnímu podvozku.

Na horní část desky jsou ještě umístěny držáky, do kterých je možné usadit jednotku vyššího řízení představovanou notebookem. Tyto držáky zamezují pohybu notebooku. Vzhledem k tomu, že se jedná o poměrně těžký HW, je umístěn nad kola, čímž je zatíží, což je žádoucí efekt. Výsledný robot je na obrázku 5.7.

Obrázek 5.7: Výsledný robot.

5.5 Jednotka vyššího řízení

Vzhledem k velké nosnosti robota je možné využít běžný přenosný počítač – konkrétně je použit notebook Acer Travelmate P653-mg. Byl využit u obou prototypů a je i součástí finálního robota. Jeho vlastnosti jsou následující:

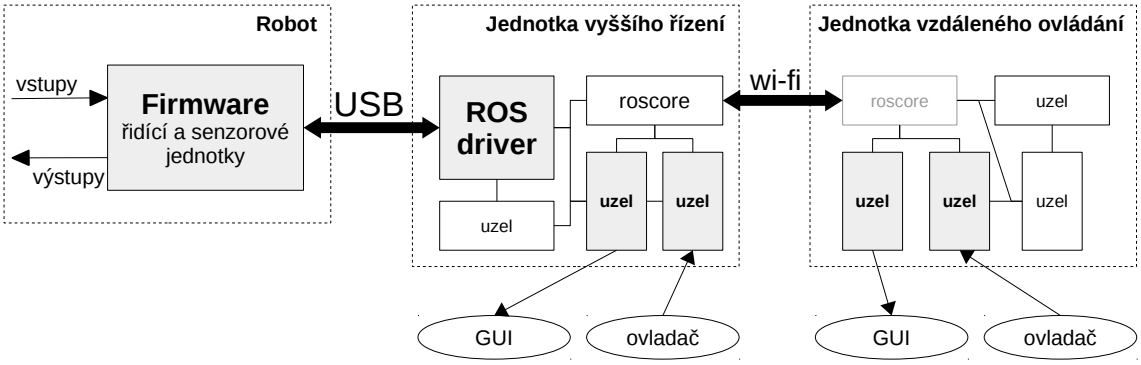
- má více než dostatečný výkon (procesor Intel core i5, 8 GB RAM)
- součástí je WiFi adaptér
- má dostatek USB portů – pro připojení řídící jednotky i dalších senzorů a jiných periferií
- je vybaven integrovanou kamerou
- běží na něm operační systém Ubuntu, který je oficiálně jediným podporovaným systémem pro robotický systém ROS.

Pro konkrétní operační nasazení je využití takto robustního systému většinou nevhodné. Robot a jeho řídící jednotka ale není na použité vyšší jednotce závislá. Dokonce je schopná samostatně provádět autonomně jednodušší programy (pohyb definovaný sekvencí řídících vektorů). Při vývoji a prototypování robotického softwaru, pro nějž je tento robot navržen, robustní platforma značně usnadňuje a urychluje činnost. Není nutné se omezovat výkonem a optimalizovat programy na rychlost a co nejmenší využívání zdrojů, což je velmi náročná a nákladná činnost. Takové řešení je plně univerzální, a to je jedním z definovaných cílů této práce.

Kapitola 6

Návrh softwaru robota

Tato část práce se zabývá návrhem tří programů. Prvním je firmware mikrokontroléru, zajišťující řízení pohonného subsystému a vyčítající data ze senzorů. Druhý zajišťuje komunikaci mezi jednotkou PIC a počítačem, jenž má na starost vyšší řízení. Třetí slouží pro vzdálené ovládání tohoto počítače.



Obrázek 6.1: Souhrnné schéma jednotlivých navrhovaných programů a jejich vzájemné propojení.

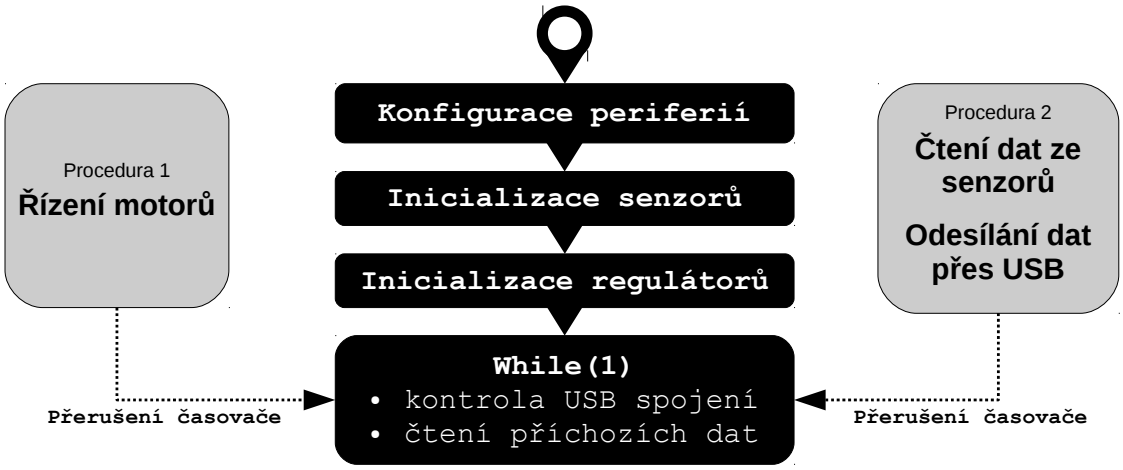
Na schematickém obrázku 6.1 jsou tři čárkované bloky, reprezentující jednotlivé systémy. První blok reprezentuje robota a jeho řídící a senzorovou jednotku, jejíž firmware zpracovává vstupy – data ze senzorů – a výstupy – řízení motorů, rozsvěcení LED diod. Mezi firmwarem a jednotkou vyššího řízení, která je reprezentována dalším blokem, probíhá obousměrná komunikace přes USB. Konkrétně data od jednotky PIC zpracovává ROS driver, který zajišťuje abstrakci robota a jeho systémů pro robotický operační systém ROS. Součástí jednotky vyššího řízení mohou být ještě další uzly. Jejich vzájemnou komunikaci zajišťuje uzel roscore. Tento uzel může být zviditelněn přes síťové rozhraní (např. bezdrátově přes wi-fi) a můžou se k němu, a tím pádem i dalším uzlů jednotky vyššího řízení, připojit uzly spuštěné na vzdáleném ovladači. Ten je představen třetím čárkovaným blokem.

V návrhu, který je zachycen na obrázku, je ještě vidět, že ovladač by mělo být možné připojit buď přímo na jednotku vyššího řízení a nebo na jednotku vzdáleného ovládání. Stejná logika by měla platit i pro grafické uživatelské rozhraní (GUI), jehož hlavním úkolem je zobrazovat data ze senzorů robota. Takto připravené rozhraní umožňuje použít robota i bez jednotky vzdáleného ovládání, například v prostředí, kde není možné vzdálené ovládání

připojit (rušení bezdrátového spojení).

6.1 Firmware mikrokontroléru

Jedná se o nejdůležitější program ze tří uvedených. Robot by totiž teoreticky měl být schopen operovat bez dalších připojených jednotek. Nemusí zvládat komplexní chování, ale měl by být schopen základních pohybů v případě, že by došlo k odpojení vyšší jednotky (například výpadek jejího napájení nebo jiná chyba). Samozřejmě by bylo možné použít řídící jednotku jen pro základní práci nad hardware a nechat řešit veškeré procedury ROS. Veřejné repozitáře ROS dokonce disponují i balíčky pro takovéto nízkoúrovňové řízení. Avšak mikrokontrolér je pro tuto činnost lépe uzpůsoben – je mnohem přesnější a rychlejší. Výkon jednotky vyššího řízení může být ušetřen pro operace, které naopak mikrokontrolér nezvládá.



Obrázek 6.2: Schéma programu mikrokontroléru.

Ze schématu na obrázku 6.2 je patrné, že po konfiguraci periferií mikrokontroléru, inicializaci senzorů a inicializaci regulátorů běží program v nekonečné smyčce provádějící kontrolu a udržování USB spojení a čtení příchozích dat z USB. Běh této smyčky je pravidelně přerušován vypršením časovačů a na to navázaným voláním procedur, které zajišťují plnění funkcí řídící a senzorové jednotky. Intervaly volání mají různou délku, odpovídající kritičnosti procedury.

Konfigurace periferií

Inicializační fáze, při které jsou konfigurovány všechny periferie mikrokontroléru, musí zajišťovat jejich správné nastavení. To je z velké části závislé na použitém mikrokontroléru a zapojení jeho pinů. Při tomto kroku je nejdůležitější připravit časovače, kvůli kterým jsou v pravidelných intervalech volány výše uvedené procedury. Samotné spuštění časovačů by nemělo proběhnout dříve, než jsou ukončeny všechny inicializace. Stejným případem jsou i časovače s externím zdrojem (čítače), použité pro enkodéry. Krom časovačů se ještě připravují A/D převodníky, PWM výstupy a I^2C.

Inicializace senzorů

Pro každý z použitých senzorů je nezbytné provést jeho inicializační proceduru, kterou je nutné dohledat v jeho dokumentaci (angl. *datasheet*). Vzhledem k tomu, že jsou všechny použité senzory připojeny přes sběrnici I^2C, je třeba mít tuto sběrnici již inicializovanou. Inicializační procedury mají pro různé senzory různou obtížnost. V rámci tohoto kroku se tedy musí inicializovat následující senzory:

- 3D kompas (HMC5883L) - datasheet [18]
- 3D gyroskop (LSM330DL) - datasheet [19]
- 3D akcelerometr (LSM330DL) - datasheet [19]
- všechny sonary (SRF08) - datasheet [24]

Snímání napětí na bateriích není třeba inicializovat, jelikož využívá pouze A/D převodníku, který je nakonfigurován už během konfigurace periferií. Stejně tak není třeba se v rámci této fáze starat o enkodéry.

Inicializace regulátorů

Na rozdíl od senzorů, které jsou připojeny přes I^2C, je třeba zajistit počáteční konfiguraci regulátorů přímo za pomocí PWM. Vzhledem k použití modelářských regulátorů, které se běžně inicializují pomocí pákového vysílače, je nutné simulovat pohyb pák vysílače pomocí změn PWM. Definice této úvodní sekvence je uvedena v dokumentaci regulátoru [22].

Nekonečná smyčka - čtení příchozích dat z USB

Smyčka realizovaná cyklem `while` s vždy pravdivou podmínkou musí zajišťovat dvě činnosti:

1. **Údržbu USB spojení** definovanou v protokolu USB.
2. **Čtení příchozích dat na sběrnici USB**, která mohou přicházet kdykoliv a obsahují řídící vektory. Na rozdíl od odesílání dat (přes USB) je příjem dat kritický. Pokud není obdržena zpráva po předem určený čas, je nutné, aby robot okamžitě zastavil svůj pohyb. Tyto zprávy totiž reflektují případné problémy na USB lince. Z toho plyne, že vyšší jednotka je zodpovědná za pravidelné odesílání řídících vektorů, které mohou být i nulové (značící stop pro robota).

Procedura řízení motorů

Tuto proceduru je třeba volat v pravidelných intervalech, což lze zajistit v obsluze přerušení, vyvolaného vypršením časovače. Vzhledem k tomu, že se jedná o nejdůležitější proceduru, která musí probíhat v přesně určeném intervalu, je potřeba tomuto přerušení zajistit vysokou prioritu zpracování. Tento přístup, jak provádět části programu závislé na přesném čase volání, je běžně využíván při programování mikrokontrolérů.

Délka intervalu volání by měla být implementována jako nastavitelná konstanta, jejíž hodnota bude nastavena experimentálně. Hodnotu lze určit na základě dostupného výkonu mikrokontroléru, rychlosti přibývání impulzů od enkodérů, průběhu PID regulace a dalších faktorů. V rámci jednoho provedení této procedury musí být nejprve ověřeno, zda-li máme aktuální vektory řízení. Pokud ne, tak se všechny motory okamžitě zataví. Pokud je vektor

řízení aktuální (což značí funkční USB spojení s jednotkou vyššího řízení), pak je pro každé řízené kolo provedena následující posloupnost kroků:

1. **Aktualizace odometrie kola** - úprava proměnné čítající/odečítající počet impulsů enkodéru. Kladná hodnota odpovídá točení kola v jednom směru a záporná v opačném. Takto připravené hodnoty je třeba při odesílání senzorických dat jednotce vyššího řízení převést na pozici v prostoru.

2. **Kontrola změny otáčení kola** - jelikož vyrobený enkodér, který robot využívá, neumí detekovat směr otáčení kola, je nezbytné směr zjišťovat z řídících vektorů a uchovávat si ho v proměnné. Při změně směru je pak nutné tuto proměnnou změnit, re-inicializovat PID regulaci a upravit stav odometrie.

3. **PID regulace** - výpočet výstupního výkonu probíhá na základě vzorců uvedených v kapitole 2.3. Cílový výstupní výkon vychází z řídících vektorů, které jsou po přijetí transformovány na otáčky za sekundu. Ty tvoří hodnotu, ke které PID regulace konverguje. Jako zpětná vazba se použijí údaje z enkodéru pro daný motor, získané z registru čítače impulsů.

4. **Řízení motoru** - získanou hodnotu z PID regulátoru je nutno normalizovat na rozsah PWM, které je odesíláno do modelářského regulátoru, který řídí motor. Tato hodnota musí reflektovat hodnotu proměnné určující směr otáčení kola.

Procedura čtení dat ze senzorů a jejich odesílání

Druhá procedura, která je volána v pravidelných intervalech pomocí časovače, se skládá z volání funkcí zajišťujících čtení dat z jednotlivých senzorů. Způsob, jakým lze ze senzorů získat data přes I^2C, je opět popsán v technické dokumentaci každého z nich. Většina z nich má definovánu minimální dobu mezi dvěma měřeními. Interval volání celé této procedury je proto závislý na čase nejdelšího z intervalů.

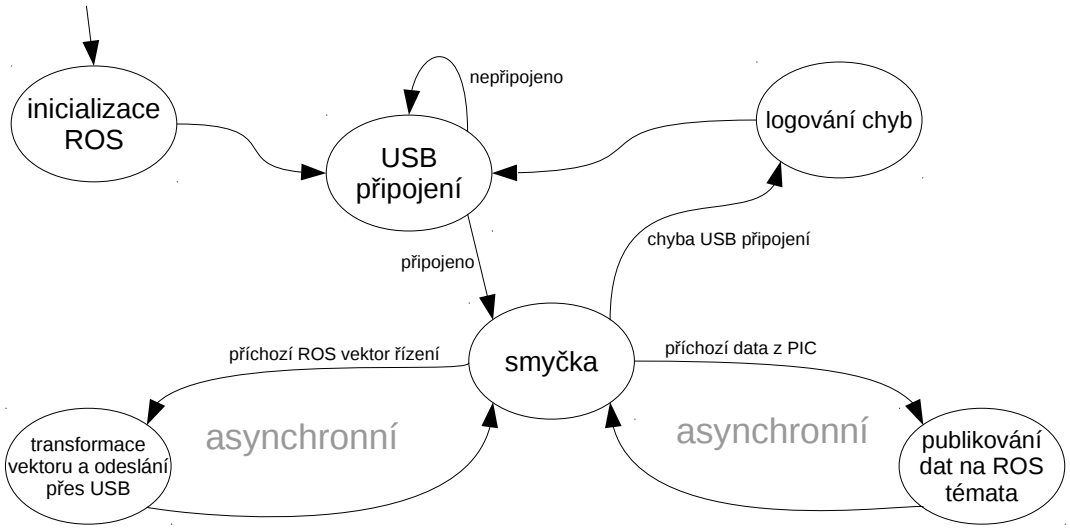
Aby nebylo nutné odesílat data přes USB po každém měření, je příhodné data ukládat a odesílat po větších celcích. Počet měření, po kterých budou vždy všechna uložená data odeslány, je opět třeba implementovat jako nastavitelnou konstantu. V rámci testování je poté možné ověřit výhodnost tohoto dávkového odesílání.

Při posílání dat je rozumné ověřovat, zda je informace obsažená v nich úplná a neporušená přenosem. Součástí datové struktury, která bude odesílána jednotce vyššího řízení (přes USB), by měl být i kontrolní součet informací v ní obsažené.

6.2 ROS driver

ROS driver slouží jako prostředník mezi řídící a senzorovou jednotkou PIC a uzly robotického operačního systému ROS. Tento uzel/program abstrahuje robota a jeho systémy tak aby, všechny další uzly/balíčky mohly být na konkrétním hardwaru nezávislé. Je možné si vytvořit kompletně vlastní architekturu uzlů, zpráv a služeb, ale je to nepraktické a nevhodné. Smyslem ROS driveru je naopak co největší využití již existující infrastruktury. Tím je myšleno hlavně využití předdefinovaných typů zpráv pro data ze senzorů i pro řízení.

Protože USB je plug & play rozhraní, musí být běžící program schopen řešit situaci, kdy bude řídící jednotka připojena nebo odpojena během jeho vykonávání. Nelze spoléhat na to, že uživatel nejprve robota připojí a poté program spustí. Navíc může systém zařízení odpojit a zase připojit, a program by poté skončil v chybovém stavu.



Obrázek 6.3: Schéma ROS driveru, jeho stavů a přechodů mezi nimi.

Na obrázku 6.3 jsou vidět stavy a přechody mezi nimi, které jsou definovány příchozími signály. Program začíná inicializací ROS prostředí a po jejím dokončení se dostává do stavu, kdy je kontrolováno USB připojení. Při neúspěchu je pokus o připojení opakován, dokud se připojení řídící a senzorové jednotky nezdaří. Poté, co je jednotka PIC úspěšně připojena, dochází k asynchronnímu přijímání zpráv pomocí signálů. Můžou nastat 3 případy:

- Pokud se jedná o příchozí zprávu od robota (jednotky PIC), jedná se o senzorická data, která je nutné rozdělit a publikovat na správná ROS témata.
- Jde-li o řídící vektor od některého nadřazeného ROS uzlu, pak je tento vektor transformován do interní struktury – vlastního typu vektoru řízení. Transformace by měla co nejméně měnit strukturu či obsah dat. Slouží spíše k převodu na datové typy, které jsou stejné pro architekturu x86 i použitý mikrokontrolér PIC. Po provedení transformace jsou upravená data odeslána přes USB jednotce PIC.

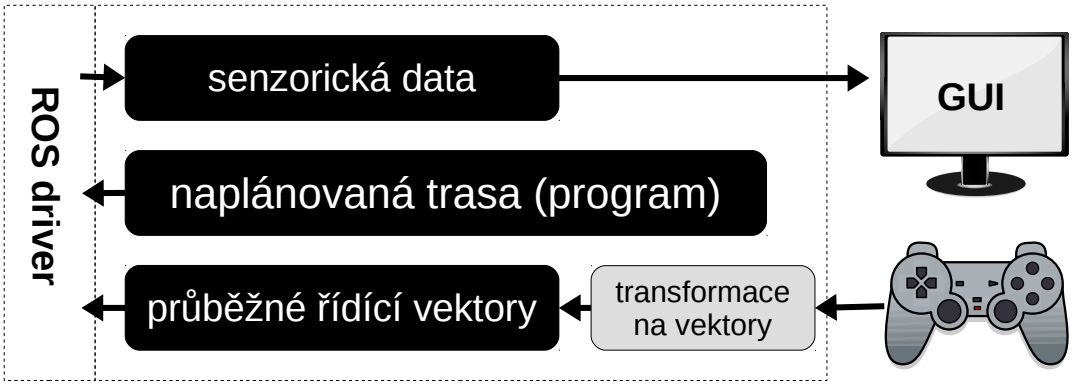
  Je důležité zajistit, aby řídící data buď přicházela v pravidelných intervalech, a nebo aby byla v ROS driveru opakovaně odesílána v pravidelných intervalech, dokud nedorazí nový řídící vektor. Firmware mikrokontroléru totiž na základě toho, jestli mu data přichází nebo ne, vyhodnocuje funkčnost komunikace.
- Dojde-li k výpadku USB připojení nebo jakékoliv jeho chybě, je tato chyba zpracována a poté je program vrácen do stavu, kdy je kontrolováno USB připojení. To znamená, že není možné dále přijímat zprávy od řídící a senzorové jednotky, ani jí přeposílat transformované řídící vektory.

6.3 Vzdálené ovládání

Pro vzdálené ovládání je možné využít již existujících balíků z repozitáře ROS. Tyto balíky je potřeba upravit a nastavit, aby vyhovovaly záměrům této práce. Konkrétní balíky, které budou vybrány v rámci implementace řešení musí splňovat následující:

- Balík pro grafické zobrazování dat senzorů ve formě grafů.
- Balík pro zachycování a zobrazování obrazu z integrované kamery.
- Program pro zpracování výstupů z joystick ovladače.
- Balík transformující surová data z joysticku do řídících vektorů.

Výhodou použití takových balíků je možnost jejich instalace na počítač, který bude sloužit jako vzdálený ovladač, a nebo na notebook, který je součástí robota a představuje vyšší řídící jednotku. Případně na oba najednou. Na obrázku 6.4, reprezentujícím komponenty vzdáleného ovládání je tato možnost ilustrována dotýkajícími se čárkovanými bloky.



Obrázek 6.4: Diagram komponent vzdáleného ovládání.

Poslední částí vzdáleného ovládání je ještě jednoduchý uzel, který bude definovat předem danou sekvenci řídících vektorů oddělených časem. Takto připravený program vlastně specifikuje jednoduchou trasu robota. Použití tohoto uzlu je cíleno na provádění experimentů ověřujících přesnost a opakovatelnost pohybů robota.

Kapitola 7

Realizace softwaru

Postup realizace všech programů navržených v předchozí kapitole je obsahem této kapitoly. Jsou zde detailně rozebrány jednotlivé kroky implementace a jejich zdůvodnění. U vzdáleného ovládání je popsáno, jaké ROS balíky byly vybrány, jak jsou vzájemně propojeny, nakonfigurovány a vůbec zprovozněny.

7.1 Firmware mikrokontroléru

Pro implementace firmware mikrokontroléru PIC24FJ64GB002 je nutné využít nástrojů, které připravila firma Microchip, jež je výrobcem těchto mikročipů. Jako vývojové prostředí je bezplatně poskytováno studio *MPLAB X IDE*. Konkrétně byla při implementaci využita verze 2.26.

MPLAB X je postaven nad prostředím NetBeans od firmy Oracle, což je multiplatformní aplikace napsaná v Javě. K tomuto IDE jsou navíc zdarma distribuovány i kompilátory jazyka C a assembleru, schopné překládat programy na instrukce z instrukční sady podporované architekturou mikrokontrolérů řady PIC24F (pro použitý 16-bitový mikročip je využit kompilátor *MPLAB XC16*). Kromě zdarma dostupné verze překladače je k dispozici i placená, obsahující mnohem lepší optimalizátor, avšak vzhledem k dostatečnému výkonu čipu nebylo třeba této možnosti využít.

Krom vývojového prostředí a překladačů ještě Microchip implementuje základní knihovny v balíku *Microchip Libraries for Applications* (zkráceně *MLA*). Součástí *MLA* jsou hlavně hlavičkové soubory pro každou vyráběnou verzi mikrokontrolérů PIC. Ty obsahují definici všech dostupných pojmenovaných konfiguračních bitů, registrů a makra abstrahující konfigurační direktivy. Dále *MLA* obsahují knihovny pro práci s nejrůznějšími nízkoúrovňovými periferiemi (časovače, oscilátory, A/D převodníky, ...) i vysokoúrovňovými periferiemi (I^2C, USART, USB, ...). Velkým problémem těchto knihoven jsou chyby, které obsahují. Během implementace jich bylo několik nalezeno a nahlášeno, nicméně nové verze *MLA* jsou vydávány v delších časových intervalech, takže zatím nebyly opraveny.

Poslední důležitou komponentou pro vývoj firmware je hardwarový programátor, sloužící k přenosu přeloženého kódu do flash paměti mikrokontroléru. Pro tento účel je využito programátoru *PICkit 3* (na obrázku 7.1), připojitelného k PC pomocí USB. Na straně mikrokontroléru, ke kterému je PICkit připojován, je třeba, aby součástí desky řídící a senzorové jednotky byl konektor a potřebné elektrické zapojení, které umožní programování čipu. Detailní zapojení je zakresleno ve všech elektrických schématech v příloze A. Mimo samotného programování slouží *PICkit 3* i k ladění programu.

Obrázek 7.1: Použitý programátor/debugger PICkit 3 od firmy Microchip.

Z možných programovacích jazyků byl pro implementaci zvolen jazyk C. Protože tento jazyk není objektový, je strukturování funkcionalit rozděleno do jednotlivých souborů, které implementují menší logické celky a zajišťují modularitu a znovupoužitelnost napsaného kódu. Veškerý kód je hojně komentován, jelikož práce na úrovni bitů a registrů nemusí být vždy dostatečně samovypovídající.

7.1.1 Konfigurace mikrokontroléru a jeho periferií

Prvotním úkolem při programování firmware mikrokontroléru je jeho nastavení, které výrazně ovlivňuje jeho funkcionalitu. Jeho špatné nastavení může vést i k nefunkčnosti jinak správného programu. Toto nastavení je určeno jednak požadovanými vlastnostmi, ale také konkrétním zapojením mikročipu na desce s elektronikou.

První část konfigurace zajišťují konfigurační registry `config1` až `config4`. V prostředí jazyka C se využívá bitového součinu (operátor `&`) předem definovaných konstant, pojmenovaných podle technické specifikace [21], které jsou předány jako parametry funcím `_CONFIG1(param)` až `_CONFIG4(param)`.

Následující použité konstanty jsou pro nastavení významné (zbylé jsou komentovány přímo ve zdrojovém kódu):

- `FWDTEN_OFF` a `DSWDTEN_OFF`- vypíná watchdog, který běžně chrání systém před uváznutím pomocí restartu po vypršení `WDT` časovače. Tento časovač musí být pravidelně nulován speciální instrukcí, čímž je signalizováno, že program běží v pořádku. Během ladění byl watchdog zapnut, ale ve výsledné implementaci ho již není potřeba, jelikož bylo ověřeno, že k uváznutí nedochází.
- `PLLDIV_DIV2` a `PLL96MHZ_ON` - základní nastavení hodin/oscilátoru potřebných pro správný běh USB.
- `FNOSC_FRCPLL` - výběr rychlého RC oscilátoru s využitím postscaleru a PLL modulu.
- `OSCIOFNC_ON` - nastavení pinu RP15 jako výstup oscilátoru - použit pro generování PWM řidícího modelářský regulátor
- `I2C1SEL_PRI` - určuje, které piny budou využity pro I^2C1. Tato hodnota zajišťuje využití přednastavených pinů `SCL1/SDA1`.

- `SOSCSEL_IO` - piny `RA4`, `RB4` lze využít pro digitální vstupně/výstupní operace

Druhou částí konfigurace je nastavení periferií. Před tím je ještě třeba u každého pinu nastavit, zda-li bude digitální či nikoliv a také jestli bude použit jako vstupní nebo výstupní. Pak už ale lze za pomoci funkcí z knihovny *MLA*, konfigurovat periferie následovně:

- **Časovače** - u obyčejných časovačů se opět pomocí logického součinu konstant, odpovídajících konfiguračním bitům, nejprve vypnou přerušení, které časovač spouští. Poté se nastaví priorita přerušení vyvolaného časovačem, hodinový vstup, předdělička hodin a čas vypršení, po kterém je spuštěno přerušení. Takto připravený časovač je spuštěn, ale dokud nejsou povolena jeho přerušení, tak nemá žádný vliv. Pro nastavení jsou použity funkce `ConfigIntTimer1(param)` a `OpenTimer1(param1, param2)` (číslo na konci značí, který časovač funkce ovlivňuje). Pro povolení přerušení slouží makro `EnableIntT1`.

  U dvou časovačů, které jsou použity jeko enkodéry, tedy čítače externích impulsů, je součástí jejich nastavení výběr pinů, které budou sloužit jako zdroje impulsů místo interního oscilátoru. K tomu slouží funkce `iPPSInput(param)`. U těchto pinů `RA0` a `RB13` je potřeba ještě zprovoznit integrované pull-up rezistory, protože nejsou součástí elektrického zapojení.

- **A/D převodník** - před konfigurováním je nutné nejprve převodník uzavřít pomocí funkce `CloseADC10()`. Pak lze nastavit vstupní pin (musí být analogový) a kanál funkcí `SetChanADC10()`. A nakonec ho konfigurovat a opět otevřít funkcí `OpenADC10(param)`. Detailní nastavení parametrů otevření lze nalézt v kódu.

  Jedním z výše vzpomenutých problémů knihovny *MLA* je to, že při nastavování vstupního pinu jsou přepnuty všechny piny `AN0` až `AN12` do analogového módu. Po zavolání je tedy potřeba znovu obnovit původní nastavení registru `AD1PCFG`;

- **PWM** - postup je obdobný jako u A/D převodníku, jen jsou využity funkce `CloseOC1()`, `iPPSOutput(param)`, `ConfigIntOC1(param)`, `OpenOC1(param1, param2, param3)`. Po konfiguraci mají PWM nastavena periodu i střídu na 0, což znamená, že jejich počáteční výstupní napětí je nulové.

- **I^2C** - pro správnou funkčnost se nejprvé nastaví registr `CLKDIV`, určující hodinový signál a z něho plynoucí komunikační rychlost I^2C sběrnice. Pro spuštění už potom stačí použít knihovní funkci `OpenI2C1(param1, param2)`, jejíž druhý parametr určuje rychlost komunikace v baudech za sekundu.

  Během testů při implementaci se ukázalo, že po spuštění I^2C dochází k občasné nefunkčnosti k ní připojených zařízení. Bylo zjištěno, že je třeba nechat po zavolání funkce `OpenI2C1` mikroprocesor chvíli „nečinný“ (pomocí `delay(500)`), aby se stabilizovalo napětí na sběrnici.

Popsaná konfigurace mikrokontroléru a jeho periferií se nachází v souboru *main.c* a *system.c*

7.1.2 Senzory

S výjimkou enkodérů a měření napětí na bateriích jsou všechny použité senzory připojené přes i^2C. To je multi-master seriová sběrnice, která se využívá pro připojení nízkorychlostních zařízení, což použité senzory jsou. V rámci řešené řídící a senzorové jednotky je použit

pouze jeden master, a tím je mikrokontrolér. Senzory jsou připojeny jako slave zařízení. Tím odpadá několik obtíží, které mohou vznikat kvůli výskytu více master zařízení. Jelikož je I^2C nakonfigurováno na začátku běhu programu, stačí již pouze posílat na sběrnici data.

Pro I^2C komunikaci jsou využity knihovny *MLA*. I v této části knihovny byly nalezeny chyby, kvůli kterým nebylo možné komunikaci zprovoznit. Problémy způsobují funkce:

- `MasterWrite(unsigned char byte)`
- `MasterGetByte(unsigned char addr, unsigned char subaddr)`
- `MastergetsI2C1(unsigned int length, unsigned char * rdptr, unsigned int i2c1_data_wait)`

Tyto funkce bylo zapotřebí pro zprovoznění I^2C komunikace reimplementovat. Aby nekolidovaly s knihovními funkcemi, je jejich deklarace umístěna v hlavičkovém souboru *i2c_modifications.h* a implementace v souboru *i2c_modifications.c*. Navíc jsou jménům funkcí přidány předpony `i2c_`.

Za využití těchto upravených funkcí spolu s originálními z *MLA* vypadá odeslání jednoho bytu následovně:

```
StartI2C1();
IdleI2C1();
i2c_masterWrite( SLAVE_ADRESA_ZARIZENI );
i2c_masterWrite( ADRESA_CILOVEHO_REGISTRU );
i2c_masterWrite( DATA );
StopI2C1();
IdleI2C1();
```

Z tohoto kusu kódu je patrné, že při začátku práce se sběrnicí je potřeba nejprve iniciovat komunikaci a čekat dokud není potvrzeno, že je sběrnice volná. Poté je odeslána adresa zařízení následovaná adresou registru zařízení a nakonec jsou odeslána data. Z toho plyne, že každé I^2C zařízení je definováno unikátní adresou velikosti 1 byte a sadou registrů, které jsou u něj dostupné.

Čtení dat odeslaných slave zařízením po I^2C sběrnici vychází z předchozího kódu a dále ho rozšiřuje:

```
StartI2C1();
IdleI2C1();
i2c_masterWrite( SLAVE_ADRESA_ZARIZENI );
i2c_masterWrite( ADRESA_CILOVEHO_REGISTRU );
RestartI2C1();
while(I2C1CONbits.RSEN);
i2c_masterWrite( SLAVE_ADRESA_ZARIZENI + 1 );
if(i2c_mastergetsI2C1(6, rdptr, 100) != 0)
    // zpracovani chyby cteni
StopI2C1();
IdleI2C1();
```

Nejprve je proveden standardní zápis do cíloveho registru a poté místo ukončení I^2C je provedena funkce `RestartI2C1()`, která spustí restartování `SDA` a `SCL` pinů. Po dokončení

restartu, který je signalizován snulováním bitu `I2C1CONbits.RSEN` pošle master zápisovou adresu, což značí, že je připraven ke čtení. Je zvykem, že zápisová adresa je adresa zařízení, ke které je přičtena jednička. Poté je již přijat předem stanovený počet bytů a ukončena komunikace na sběrnici.

Kompas HMC5883L

Kompas je inicializován do módu průběžného měření a poskytuje data s frekvencí 15 Hz. Počet měřených vzorků, ze kterých je vypočtena výsledná hodnota, je 8. Zisk měření je nastaven na hodnotu 230 $LSb/Gauss$, což odpovídá nejmenší možné přesnosti kompasu - 4,35 mG/LSb. Toto nastavení reflektuje tabulka 7.1.

Adresa zařízení:	`0x3C`
Inicializační procedura:	– odeslání `0x3C 0x00 0x70` – odeslání `0x3C 0x01 0xE0` – odeslání `0x3C 0x02 0x00` – čekání 6 ms
Vyčítání dat:	– odeslání `0x3C 0x03` – odeslání `0x3D` – čtení 6 bytů
frekvence měření:	15 Hz

Tabulka 7.1: Shrnutí implementovaného I^{2}C protokolu kompasu.

Akcelerometr LSM330DL

Za pomoci registru `CTRL_REG1_A` je spuštěno měření akcelerometru v normálním módu (frekvence měření 1,344 kHz), probíhající pro všechny 3 osy - X, Y, Z. Hodnota zapsaná do `CTRL_REG4_A` zajišťuje, že při čtení dat z registru nebude tento registr přepisován novým měřením. Určuje, že data budou ukládána ve stylu little endian a bude použit maximální rozsah ± 16 G. Ostatní registry jsou ponechány v přednastavených hodnotách, které znamenají, že nebude využito žádného filtrování dat, stejně jako nebudou využita žádná přerušení. Nebude využita sběrnice SPI.

Adresa zařízení:	`0x32`
Inicializační procedura:	– odeslání `0x32 0x20 0x97` – odeslání `0x32 0x23 0xF8`
Vyčítání dat:	– odeslání `0x32 0xA8` – odeslání `0x33` – čtení 6 bytů
frekvence měření:	1,344 kHz

Tabulka 7.2: Shrnutí implementovaného I^{2}C protokolu akcelerometru.

Gyroskop LSM330DL

Pro gyroskop je zvolena nejmenší možná frekvence, ale nejvyšší možná přesnost 2000 dps. Hodnoty měření jsou filtrovány horní propustí. Při čtení není čtecí registr měněn a data

jsou poskytována ve formátu little endian, stejně jako u akcelerometru. Také u gyroskopu nejsou povolena přerušení a není využíváno SPI sběrnice.

Adresa zařízení:	`0xD0`
Inicializační procedura:	– odeslání `0xD0 0x20 0x0F` – odeslání `0xD0 0x22 0x10` – odeslání `0xD0 0x23 0xF0` – odeslání `0xD0 0x24 0x10`
Vyčítání dat:	– odeslání `0xD0 0xA8` – odeslání `0xD1` – čtení 6 bytů
frekvence měření:	100 Hz

Tabulka 7.3: Shrnutí implementovaného I^2C protokolu gyroskopu.

Sonary SRF08

Na rozdíl od ostatních senzorů, které mají pevně dané I^2C adresy, je u sonaru možné adresu nastavit. Sonarů totiž může být na robotovi více a adresy musí být unikátní. Změna adresy je záležitost, pro kterou je nutno napsat jednoduchý program, který není součástí firmware řídící a senzorové jednotky. Každý sonar je potom potřeba samostatně zapojit a spustit tento program, pomocí nějž je mu nastavena nová adresa.

Připojené sonary není třeba nijak inicializovat, protože je využito přednastavené konfigurace, což odpovídá dosahu 11 metrů a 67 ms mezi jednotlivými měřeními. Výsledky měření vzdálenosti jsou vraceny v centimetrech (druhé dva byty) a v prvním bytu je výsledek měření světelného senzoru (který je součástí sonaru).

Adresa zařízení:	– počáteční: `0xE0` – nastavitelné: `0xE0` až `0xFE`
Změna adresy:	– odeslání `0xE0 0x00 0xA0` – odeslání `0xE0 0x00 0xAA` – odeslání `0xE0 0x00 0xA5` – odeslání `0xE0 0x00 0xF0`
Inicializační procedura:	- - -
Vyčítání dat:	– odeslání `0xE0 0x00 0x51` – čekání 67 ms – odeslání `0xE0 0x01` – odeslání `0xE1` – čtení 3 bytů
frekvence měření:	15 Hz

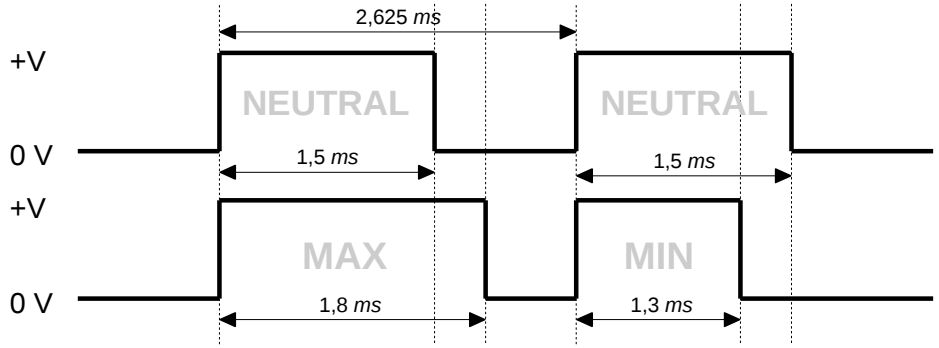
Tabulka 7.4: Shrnutí implementovaného I^2C protokolu sonaru.

7.1.3 Řízení modelářských regulátorů

Proti řízení motorů pomocí integrovaných obvodů *DRV8833* (H-můstků), které byly na prvním prototypu a ovládaly se PWM signálem s libovolnou periodou a délkou pulsu, je

řízení modelářských regulátorů dáno jednoduchým „protokolem“. Ten je velmi podobný například i pro ovládání serva či jiných pozičně ovládaných součástek.

Základem řízení je délka kladného impulsu, který pro neutrál (středový stav) odpovídá 1,5 ms, pro maximální výkon je 2 ms a pro minimální 1 ms. Perioda signálu bývá v rozmezí 20 ms – 2,5 ms, což odpovídá frekvenci 50 Hz – 400 Hz. Na periodě je závislá rychlost reakce řízení. Čím výšší, tím rychlejší reakce. Pro řízení robota byla zvolena frekvence 381 Hz, tedy perioda 2,625 ms. V dokumentaci regulátoru [22] je doporučeno využívat pouze rozmezí od 1,2 ms do 1,8 ms délky impulsu a firmware toto doporučení dodržuje. Graf PWM signálu, vycházející z reálně využitých hodnot, vypadá jako na obrázku 7.2.



Obrázek 7.2: Podoba požadovaného PWM, který řídí modelářský regulátor.

Protože funkce `SetDCOC1PWM(param1, param2)` a `SetDCOC2PWM(param1, param2)`, nastavující hodnoty definující PWM signál, nejsou v milisekundách, je potřeba vypočítat příslušné hodnoty podle dokumentace mikrokontroléru ([21]). Vypočítané hodnoty uvádí tabulka 7.5. První parametr zmíněných funkcí je délka kladného impulsu a druhý perioda.

Perioda	31999
1,5 ms	24000
1,2 ms	19200
1,8 ms	28800

Tabulka 7.5: Vypočítané hodnoty parametrů funkcí řídící PWM.

Jelikož se jedná o modelářský regulátor, který je většinou řízen pákovými vysílači, jsou pokyny pro inicializaci a nastavování uváděny jako kombinace pohybů pákového vysílače. Tyto pohyby jsou minimální, maximální výchylka a neutrál a odpovídají výše uvedeným hodnotám v milisekundách. Není problém tedy pomocí PWM tyto pohyby simulovat.

Inicializace regulátoru je jednoduchá. Nejprve je nutné spustit firmware řídící a senzorové jednotky PIC, který nakonfiguruje PWM a nastaví ho tak, aby signál odpovídal neutrálu (1,5 ms). Jakmile je PWM připraveno, je možné spustit připínačem regulátor a vyčkat na zvukovou a světelnou signalizaci, oznamující úspěch inicializace (dlouhé pípnutí následované bliknutím LED).

Pro programování regulátoru, které je řádově komplexnější, bylo nutné vytvořit jednoduchý program mikrokontroléru, který pomocí USB přijímá symboly `a`, `s`, `d` a na jejich základě mění nastavení PWM na min, neutrál a max. Tímto je možné nastavovat regulátory stejně jako při použití pákového vysílače. Po naprogramování si regulátor hodnoty uloží do

perzistentní paměti, takže je možné provozovat firmware jednotky PIC bez nutnosti pokaždé regulátory programovat. Postup programování je popsán v dokumentaci regulátoru [22].

Funkce pro ovládání motorů využitím regulátorů se nachází v souborech *motor_regulator.c* a *motor_regulator.h*.

7.1.4 USB komunikace - sériový port

USB protokol definuje několik tříd zařízení, přičemž každá má svá pozitiva i negativa. Součástí knihovny *MLA* je i USB stack, který implementuje některé z těchto tříd. Pro tuto práci bylo nejvýhodnější použít třídu CDC device emulující RS-232 sériový adaptér. Výhody jsou následující:

- dostatečná datová propustnost v obou směrech komunikace (možnost provádět i *bulk* přenosy)
- není nutné implementovat ovladač do jádra systému (používá se generický) - multiplatformní
- univerzalita RS-232 protokolu (standard pro roboty)
- mnoho knihoven pro práci s RS-232

Pro využití USB stacku při implementaci je třeba do projektu vložit 4 soubory z *MLA*: *usb_config.h*, *usb_descriptors.c*, *usb_device.c*, *usb_device_cdc.c*. Tyto soubory obsahují implementaci komunikační zvolené třídy. Zbývá tedy jen v nich pomocí konstant zvolit možnosti komunikace:

- zapnout volbu `USB_PING_PONG__FULL_PING_PONG`
- zvolit `USB_INTERRUPT` pro využívání přerušení (místo aktivního čekání)
- využití interního vysílače - `USB_INTERNAL_TRANSCEIVER`
- zapnout plnou rychlost USB – 12 MHz – přepínačem `USB_FULL_SPEED`
- specifikovat vlastní název výrobce v USB deskriptoru na `"Herman"`
- specifikovat vlastní označení zařízení v USB deskriptoru na `"Robot controller"`

Obzvláště podstatné je nastavení vlastního názvu výrobce a označení produktu, jelikož díky tomu je možné lépe detekovat zařízení na jednotce vyššího řízení.

Základní struktura programu celého firmware se odvíjí právě od USB komunikace. Tu lze rozdělit do dvou částí. První je nekonečná smyčka, v níž je kontrolováno a udržováno spojení. Druhá je představována obsluhou přerušení, které USB vyvolává. Před důkladnějším rozborem první části je vhodné si uvědomit, že komunikace je základní činností řídící a senzorové jednotky, bez které je robot nečinný.

Nekonečná smyčka mikrokontroléru vypadá takto:

```
  while(1) {
    if(USBGetDeviceState() < CONFIGURED_STATE)
      continue;

    if(USBIsDeviceSuspended() == true)
      continue;

    APP_DeviceCDCTasks();
  }
```

Je z ní patrné, že pokud není USB nakonfigurováno nebo je uspané, pak nedochází k provádění funkce `APP_DeviceCDCTasks()`, která obsahuje čtení příchozích zpráv, na jejichž základě robot provádí svou činnost. Protože není použito uspávání USB, je jeho kontrola spíš formalitou. Zato stav konfigurace USB je zcela zásadní. Tento stav je měněn asynchronně pomocí přerušení USB. Přerušení jsou řešena pomocí callback funkce:

```
bool callback(USB_EVENT event, void *pdata, uint16_t size) {
  switch((int) event) {
    . . .

    case EVENT_CONFIGURED:
      APP_DeviceCDCInitialize();
      EnableIntT4;
      EnableIntT5;
      break;

    . . .
  }

  return true;
}
```

Právě přerušení obsahující událost `EVENT_CONFIGURED` zajišťuje změnu stavu na `CONFIGURED_STATE`. Při tom je provedena inicializace sériové linky a jsou povoleny přerušení časovačů, spouštějící proceduru řízení a proceduru získávání dat ze senzorů. Vzhledem k tomu, že napájení mikrokontroléru je zprostředkováno pomocí USB, dochází ke konfiguraci vždy právě jednou. Není proto potřeba řešit situaci, kdy by bylo USB odpojeno a mikrokontrolér by musel dále běžet. Nejedná se ale o příliš obtížnou věc na implementaci.

Ihned po konfiguraci začne v nekonečné smyčce probíhat funkce `APP_DeviceCDCTasks()`. Ta obstarává udržbu USB komunikace sériové linky pomocí funkce `CDCTxService()`. Jakmile smyčka běží, je možné asynchronně i synchronně volat funkce `getsUSBUSART(data, dataSize)` a `putUSBUSART((data, dataSize)`, které přijímají a odesílají data po USB za pomocí rozhraní USART. Příjimání dat není nijak omezeno, ale odesílání může proběhnout pouze pokud je USB připraveno, k čemuž slouží kontrolní funkce `USBUSARTIsTxTrfReady()`. Ta nesmí být použita jako blokující, protože by mohlo docházet k uváznutí.

Emulovaná seriová linka RS-232, běžící přes USB, je inicilizovaná s následujícmi parametry:

Komunikační rychlost:	115200 Bd/s
Počet datových bitů:	8
Parita:	žádná
Počet stop bitů:	1

Tabulka 7.6: Parametry sériové linky.

Pomocí ní lze posílat a přijímat jak binární, tak textová data. Pro testovací účely bylo využito textového formátu dat, jelikož je lze přijímat přes libovolný terminálový program. Ve firmware je naopak využito binárního formátu. Důvody jsou popsány v podkapitole 7.2.

7.1.5 Řízení, PID regulace a odometrie

Tyto tři funkcionality jsou řešeny v rámci souborů *system.c*, *pid_controller.h* a *pid_controller.c*. Jelikož využívají množství globálních/externích proměnných, které musí být k dispozici i mezi dalšími soubory, musí být deklarovány v souboru *global_variable.h*. Ten je potom vložen do těch, které proměnné využívají. Počáteční definice musí být umístěna v *main.c*.

Základem pro řízení je **kinematický model** diferenciálního podvozku robota. Uvažujme pohyb robota pouze ve dvourozměrném prostoru. Dle Hellströma [25] jsou rovnice pro výpočet obvodové rychlosti pravého (v_1) a levého (v_2) kola následující:

$$v_l = \omega(R + \frac{l}{2}) \quad v_r = \omega(R - \frac{l}{2}) \tag{7.1}$$

Konstanta l odpovídá rozchodu kol a je pro vyrobeného robota rovna $0,44$ m. Proměnná ω je v jednotkách $rad \cdot s^{-1}$ a znamená rychlost otáčení celého podvozku. Budeme-li uvažovat pouze otáčení, jehož střed leží ve středu mezi koly, bude $R = 0$. Připočteme-li navíc u obou kol lineární dopřednou rychlost celého robota (v), dostaneme výsledné vztahy pro obvodovou rychlost jednotlivých kol v $m \cdot s^{-1}$:

$$v_l = v \ + \omega\frac{l}{2} \quad v_r = v \ - \omega\frac{l}{2} \tag{7.2}$$

Kombinací lineární dopředné rychlosti a otáčení robota kolem své osy je možné docílit jeho otáčení se středem ležícím mimo něj. Pro couvání robota stačí do rovnic dosadit negativní rychlost v.

Uvedené vzorce stačí jednoduše převést do zdrojového kódu. Dosazení do nich probíhá při každém přijetí řídícího vektoru. Za hodnotu v je dosazena jeho lineární složka `x` a za ω úhlová rychlost ve směru `z`. Zbylé hodnoty vektoru nejsou využity. Výsledné hodnoty v_1 a v_2 jsou násobeny počtem impulsů enkodéru za ujetý jeden metr (452) a ty již přímo využívá PID regulátor jako cílové hodnoty prostřednictvím globální/externí proměnné. Kromě výpočtu a uložení do proměnné, probíhající při příchodu řídícího vektoru, je vynulována hodnota proměnné `_msg_received_watchdog`, která je inkrementována v rámci procedury řízení.

Při **proceduře řízení** je tedy hodnota proměnné `_msg_received_watchdog` inkrementována a zároveň je kontrolováno, jestli již nepřesáhla určené hodnoty, což značí výpadek příchodu řídících vektorů. V případě výpadku jsou motory (samozřejmě přes regulátory) zastaveny. V opačném případě je provedena úprava odometrie impulsů následovaná kontrolou změny směru otáčení kol. Ten je ukládán v proměnné `_lw_direction` pro levé kolo a `_rw_direction` pro pravé. Nabývá pouze dvou hodnot 1 a -1. Těmito konstantami jsou

pak násobeny „přírůstky“ odometrie a výstupy na motory. Výstupy na motory se vypočítávají pomocí PID regulátoru. Jeho implementace je pouze převedením vzorců uvedených v kapitole 2.3 do zdrojového kódu, přičemž integrál je implementován jako průběžná suma.

Posledním důležitým krokem procedury řízení je prevence přetečení čítače impulsů enkodéru, který nabývá `unsigned` hodnot, ale je pouze 16-bitový. Pokud je překročena polovina rozsahu, což je rozpoznáno na základě hodnoty nejvyššího bitu (MSB) rovného 1, tak je tento bit vynulován (odpovídá odečtení 32768). Tato změna musí být reflektována do proměnné odometrie impulsů konkrétního kola přičtením/odečtením hodnoty 32768.

Odometrie impulsů je přepočítávána na standardní odometrii, tak jak je běžně chápána, až při odesílání dat ze senzorů přes USB do vyšší jednotky řízení. Přepočet vychází z předchozích rovnic kinematického modelu. Nejprve se stanoví délka ujeté trasy pro jednotlivá kola.

$$D_l = \frac{\text{suma impulsů enkodéru levého kola}}{\text{počet impulsů na metr}} \quad D_r = \frac{\text{suma impulsů enkodéru pravého kola}}{\text{počet impulsů na metr}} \tag{7.3}$$

Vypočte se průměrná ujetá vzdálenost z těchto hodnot

$$D_t = \frac{D_l + D_r}{2} \tag{7.4}$$

a také průměrný úhel, o který se robot během jízdy otočil. Konstanta l odpovídá rozchodu kol.

$$theta = \frac{D_r - D_l}{l} \tag{7.5}$$

Výsledné souřadnice, o které se robot pohnul, jsou získány pomocí následných rovnic. Vypočítané hodnoty jsou v metrech.

$$X = D_t \cdot sin(theta) \quad Y = D_t \cdot cos(theta) \tag{7.6}$$

7.2 ROS driver

První testovací implementace nebyla vytvořena jako ROS uzel, ale běžný GUI program, napsaný v jazyce C++ s využitím frameworku Qt. Aplikace posloužila k otestování multiplatformní knihovny QtSerialPort (verze Qt 5.4) a ověření USB komunikace. Vzhledem k úspěšným testům byl pro výsledný ROS driver také použit jazyk C++ a QtSerialPort. Mimo knihovny pro práci se sériovou linkou je z Qt využito i QThread pro programování vláken. Výhodné je i programování s využitím signalů a slotů.

Při spuštění driveru je inicializováno prostředí ROS uzlu a spuštěna smyčka.

```
ros::Rate loopRate(10);

while (ros::ok()) {}
  ros::spinOnce();
  loopRate.sleep();
}
```

Úvodní nastavení `loopRate(10)` slouží k určení frekvence s jakou smyčka bude probíhat. Zde je rovna 10 Hz. V rámci jednoho jejího průběhu jsou vždy zpracovány příchozí nebo odchozí zprávy, volání služeb a další akce ROS. Je vidět, že smyčka probíhá „naprázno“. Ač v rámci ní není vykonáván žádný produktivní kód, tak během jejího běhu dochází k příjmu Qt signálů, které spuštějí funkce zajišťující chod robota. Signály, které jsou přijímány po celou dobu běhu programu jsou:

- `timeout()` - tento signál je pravidelně spouštěn QTimerem nastaveným na 1000 ms. Slot, který tento signál zpracovává ověřuje připojení k sériové lince. Pokud je systém ve stavu `DISCONNECTED`, pokusí se o její připojení.
- `error(QSerialPort::SerialPortError)` - při jeho přijetí je chyba zalogována a vyhodnocena její závažnost a následná reakce.

Zbylé dva signály se vyskytují až po úspěšném připojení k řídící a senzorové jednotce PIC přes sériovou linku. Navázání slotů na tyto signály je zajištěno po jejím připojení a naopak je vazba zrušena, dojde-li k chybě linky. Jedná se o signály:

- `readyRead()` - je vyvolán ve chvíli, kdy jsou na sériovém portu data ke čtení. Protože tato data obsahují údaje ze senzorů, jsou zpracovány do podoby ROS zpráv a odeslána na náležitá témata.
- `dataToWrite(float, float, float, float, float, float)` - jedná se o vlastní definovaný signál, který je emitován při obdržení ROS zprávy obsahující řídící vektor. Ten je transformován do interních struktur a odeslán přes sériový port.

7.2.1 Připojení sériového portu

Většina programů se připojuje k sériovému portu pomocí virtuálního názvu rozhraní (pro linux např. \ttyAM0, pro windows např. COM1). Tento způsob má nevýhodu, že musí být argumentem ROS driveru aktuální název portu. U této práce bylo možné využít automatickou detekci zařízení, protože lze nastavit deskriptor jednotky PIC a navíc je nepravděpodobné (a nerozumné), že bude připojeno k jedné jednotce vyššího řízení více než jedna jednotka PIC. Při připojování jsou tedy nalezeny všechny dostupné sériové porty a je vybrán ten, který má název zařízení *Robot controller* a jméno výrobce *Herman*.

7.2.2 Formát dat sériového portu

Protože přenášená data v obou směrech jsou číselná (`int`, `float`), je využito binárního přenosu, což značně snižuje množství přenášených dat (proti textovému přenosu). Je však nezbytné zajistit jejich správnou interpretaci pro různé architektury. K tomu slouží datové typy `int16_t` `int32_t` a `float`. Ty jsou využity pro konstrukci složených datových typů struktur. Nejprve se definují vlastní vektorové struktury:

```
typedef struct vecotr_3d_int16 {
    int16_t x;  int16_t y;  int16_t z;
} VECTOR_3D_INT16;

typedef struct vector_3d_float {
    float x;  float y;  float z;
} VECTOR_3D_FLOAT;
```

Z nich je složen transformovaný vektor řízení, jehož podoba je identická s ROS zprávou typu `geometry_msgs::Twist`, jen jsou použity bezpečné datové typy.

```
typedef struct robot_twist {
    VECTOR_3D_FLOAT linear;
    VECTOR_3D_FLOAT angular;
} ROBOT_TWIST;
```

Komplexnější je struktura pro přenos senzorických dat, které krom nich samotných ještě obsahují kontrolní sumu:

```
typedef struct sensors_measurement {
    VECTOR_3D_FLOAT compass;
    VECTOR_3D_FLOAT gyroscope;
    VECTOR_3D_FLOAT accelerometer;
    VECTOR_3D_FLOAT pose;
    float sonarRange[NUM_OF_SONARS];
    float batteryVoltage;
    float controlSum;
} SENSORS_MEASUREMENT;
```

Tyto struktury musí být stejně definovány jak v C++ kódu ROS driveru, tak ve firmware řídící a senzorové jednotky. Protože obsahují datové typy s pevnou bitovou šířkou, je jejich přenos velmi snadný, protože struktura má vždy pevně danou velikost a lze využít standardní funkce `sizeof` pro určení množství přenášených bytů.

7.2.3 ROS zprávy

Následující výčet použitých typů ROS zpráv popisuje jejich tvar a využití. Bylo snahou využít pokud možno standardních zpráv, protože to umožňuje navázat ROS driver na existující balíky v repozitářích, což je hlavním smyslem použití ROS na tomto robotovi.

U všech zpráv s vyjímkou `geometry_msgs/Twist.msg` se používá hlavička zprávy – `header`, která obsahuje časové razítko zprávy a její pořadové číslo. Pro jednu sadu měření všech senzorů je vždy pořadové číslo i časové razítko stejné.

geometry_msgs/Twist.msg

Základem pohybu robota jsou řídící vektory, což jsou právě zprávy typu `geometry_msgs/Twist.msg` (viz tabulka 7.7). Jedná se o reprezentaci pohybu složeného ze dvou složek – lineární rychlosti a úhlové rychlosti. Obě jsou definovány pro trojrozměrný prostor. Je tedy možné je použít jak pro létající roboty, tak pro pozemní.

Diferenciálně řízený podvozek, který je základem vyrobeného robota, je řízen pouze dvěma složkami z šesti, které můžou vektory obsahovat. Z vektoru lineární rychlosti je využito složky x a z vektoru úhlové rychlosti složky y. Ty postačují pro plnohodnotné ovládání jízdy robota.

Typ	Název
geometry_msgs/Vector3	linear
geometry_msgs/Vector3	angular

Tabulka 7.7: Zpráva geometry_msgs/Twist.msg

sensor_msgs/Imu.msg

Zpráva pokrývá data rovnou ze dvou senzorů naráz (viz tabulka 7.8). Hlavička (header) obsahuje název rámce, který je `imu`. Vyplňovány jsou pouze položky `angular_velocity`, obsahující měření úhlové rychlosti pro osy x, y a z, a položka `linear_acceleration` určující lineární zrychlení ve všech třech osách.

Typ	Název
std_msgs/header	header
geometry_msgs/Quaternion	orientation
float64[9]	orientation_covariance
geometry_msgs/Vector3	angular_velocity
float64[9]	angular_velocity_covariance
geometry_msgs/Vector3	linear_acceleration
float64[9]	linear_acceleration_covariance

Tabulka 7.8: sensor_msgs/Imu.msg

sensor_msgs/MagneticField.msg

Zpráva `sensor_msgs/MagneticField.msg` (viz tabulka 7.9) je obdobná jako `sensor_msgs/Imu.msg`. Rozdílem hlavičky je pouze jiný název rámce (`mag`). Vyplněna je položka `magnetic_field`, jejíž vektor je naplněn daty z kompasu.

Typ	Název
std_msgs/Header	header
geometry_msgs/Vector3	magnetic_field
float64[9]	magnetic_field_covariance

Tabulka 7.9: Zpráva sensor_msgs/MagneticField.msg

sensor_msgs/Range.msg

Tento typ zpráv (viz tabulka 7.10) je využíván sonary. Hlavička obsahuje název rámce `sonar`. Pro každý sonar musí být zpráva vytvořena zvlášť. Jako `radiation_type` se nastavuje konstanta `sensor_msgs::Range::ULTRASOUND` značící, že obsahem zprávy je měření sonaru a ne IR senzoru. Položky `min_range` a `max_range` jsou nastaveny podle definovaných parametrů HW (min - 0,03 m, max 11 m). Pro `field_of_view` je zvolena hodnota konstanta 1,1. Jediná položka, která se mění, je tedy `range`. Její hodnota je získávána na základě měření sonaru.

Typ	Název
uint8	ULTRASOUND=0
uint8	INFRARED=1
std_msgs/Header	header
uint8	radiation_type
float32	field_of_view
float32	min_range
float32	max_range
float32	range

Tabulka 7.10: Zpráva sensor_msgs/Range.msg

nav_msgs/Odometry.msg

Odometrická data, která obsahuje tento typ zprávy (viz tabulka 7.11), jsou reprezentována jako pozice, na které se robot nachází v daný čas. Tento čas je uložen v hlavičce, spolu s pořadovým číslem zprávy a typem rámce `odom`. Pozice (`pose`), kde se robot nachází je součástí senzorických dat.

Typ	Název
std_msgs/Header	header
string	child_frame_id
geometry_msgs/PoseWithCovariance	pose
geometry_msgs/TwistWithCovariance	twist

Tabulka 7.11: Zpráva nav_msgs/Odometry.msg

lwnmwr/BatteryState.msg

Jedná se o jedinou zprávu, která vyžadovala vlastní definici (viz tabulka 7.12), protože pro stav baterie žádná standardní neexistuje. Obsahem je hlavička, ke které se stav napětí baterie vztahuje, a samotné napětí baterie. Hodnotou je desetinné číslo udávající napětí ve voltech.

Typ	Název
Header	header
float32	voltage

Tabulka 7.12: Zpráva lwnmwr/BatteryState.msg

7.3 Vzdálené ovládání

Jednou z mnoha výhod ROS je síťová nezávislost uzlu. Všechny uzly mohou běžet buď na jednom počítači a nebo může každý běžet na jiném. Spojovacím prvkem je `roscore`, který běží vždy právě na jednom počítači a přes něj se spojují všechny ostatní uzly. Není tedy nutné rozlišovat vzdálené a lokální ovládání, protože pro oba přístupy jsou použity stejné balíky, jen spuštěné na jiných strojích. Dokonce lze mít na jednotce vyššího řízení, kterou nese robot, spuštěn pouze naprogramovaný ROS driver a všechny ostatní uzly provozovat

na druhém bezdrátově připojeném PC (vzdálený ovladač). To však není vhodné z důvodů, které budou níže objasněny.

7.3.1 Připojení vzdáleného PC

Pro provoz jsou využity dva počítače. Jeden je součástí robota (jednotka vyššího řízení) a běží na něm minimálně `roscore` a ROS driver. Druhý počítač je s prvním spojen přes wi-fi připojení – jsou na jedné síti. První má IP adresu 192.168.0.11 a druhý 192.168.0.22. Na prvním na kterém běží `roscore` je nutné exportovat dvě konstanty:

1. `ROS_IP=192.168.0.11` - IP adresu počítače
2. `ROS_MASTER_URI=http://192.168.0.11:11311` - označující lokální IP adresu a port, na kterém běží `roscore`.

Pro druhý jsou také exportovány dvě konstanty:

1. `ROS_IP=192.168.0.22` - IP adresa počítače
2. `ROS_MASTER_URI=http://192.168.0.11:11311` - IP adresa a port prvního počítače, na němž je spuštěn `roscore`.

Po tomto nastavení je již možné na libovolném z počítačů spouštět libovolný uzel.

7.3.2 Problémy wi-fi

Síťové spojení robota a vzdáleného ovladače je realizováno přes wi-fi, což je rychlá technologie s dostatečnou datovou propustností. Dokonce je možné skrz ni přenášet i stream z kamery. Problémem je ale TCP/IP spojení, které ROS využívá. TCP/IP zajišťuje spolehlivý přenos, takže je zaručeno, že odeslaná zpráva se z ovladače dostane do robota, ale není garantován čas, za který zpráva dorazí. A právě čas je u teleoperace robota klíčový parametr a většinou se proto dává přednost UDP před TCP. U wi-fi dochází často k výpadkům paketů kvůli silnému přehlcení pásma a z toho vznikajícího rušení a doručení zprávy může trvat i jednotky sekund, což je pro řízení nevhodné. Robot proto musel být vybaven externím wi-fi adaptérem, který má lepší příjem než běžný integrovaný v notebooku. Ideální by bylo využít UDP komunikaci, ale ROS ji prozatím nepodporuje.

7.3.3 ROS uzly a témata

Pro realizaci vytyčených cílů bylo využito kromě naprogramovaného ROS driveru několik balíků z repozitáře a ještě jeden naprogramovaný pro testovací jízdy.

lwnmwr

Jedná se o balík, který obsahuje spustitelný uzel `lwnmwr_node`, což je implementovaný ROS driver. Odebírá zprávy – řídící vektory – z tématu `\cmd_vel` a publikuje zprávy na témata `\sensors\imu`, `\sensors\magnetic_field`, `\sensors\sonar1`, `\odom` a `\battery_state`.

joy

Spolu s tímto balíkem z repozitáře je nutné ještě do operačního systému nutné nainstalovat ovladač pro gamepad, který umožňuje proporcionální řízení (nebo řízení tlačítky) robota. Spouštěný uzel `joy` je schopen získávat data z gamepadu a publikovat je na téma `\joy`. Nejedná se však o řídící vektory, ale o zprávy obsahující stav tlačítek a proporcionálních pák.

U tohoto uzlu se nastavují tři parametry:

1. `dev` - název rozhraní, ke kterému je gamepad připojen (např. `/dev/input/js1`)
2. `deadzone` - „mrtvá zóna“ pák gamepadu, nastaveno 0,1
3. `autorepeat_rate` - frekvence odchozích zpráv z tohoto uzlu, nastaveno na 20 Hz

teleop_twist_joy

Pro převod zpráv z tématu `\joy` na téma `\cmd_vel` slouží uzel `teleop_node`, který je obsažen v balíku teleop_twist_joy. Tento uzel transformuje zprávy obsahující stav tlačítek a pák gamepadu na řídící vektory. Jeho nevýhodou je, že nedokáže publikovat zprávy s předem danou frekvencí, což je vyžadováno ROS driverem. Tento problém ale naštěstí řeší uzel `\joy_node`.

Pro správné fungování tohoto uzlu musel být připraven konfigurační soubor mapující tlačítka a osy pák gamepadu na řídící funkce. Mimo to soubor specifikuje rozsahy hodnot složek výstupních řídících vektorů.

uvc_camera

Uzel `uvc_camera_node` z balíku uvc_camera zpracovává data z integrované kamery notebooku, který je umístěn na robotovi, a publikuje je jako zprávy typu `sensor_msgs/Image` na několik různých témat s různými úpravami.

rqt

Jedná se o sadu uzlů, vhodných především pro vizualizaci dat. Uzly je možné spouštět jako samostatné programy nebo jako widgety sloučené do jednoho zobrazení. Navíc je možné připravené zobrazení widgetů uložit jako sestavu, čehož je využito, a připravená sestava zobrazuje data ze všech senzorů jako grafy. Mimoto je využito ještě widgetu pro zobrazování obrazu z kamery (zpráv typu `sensor_msgs/Image`) a widgetu pro logování `rosout`.

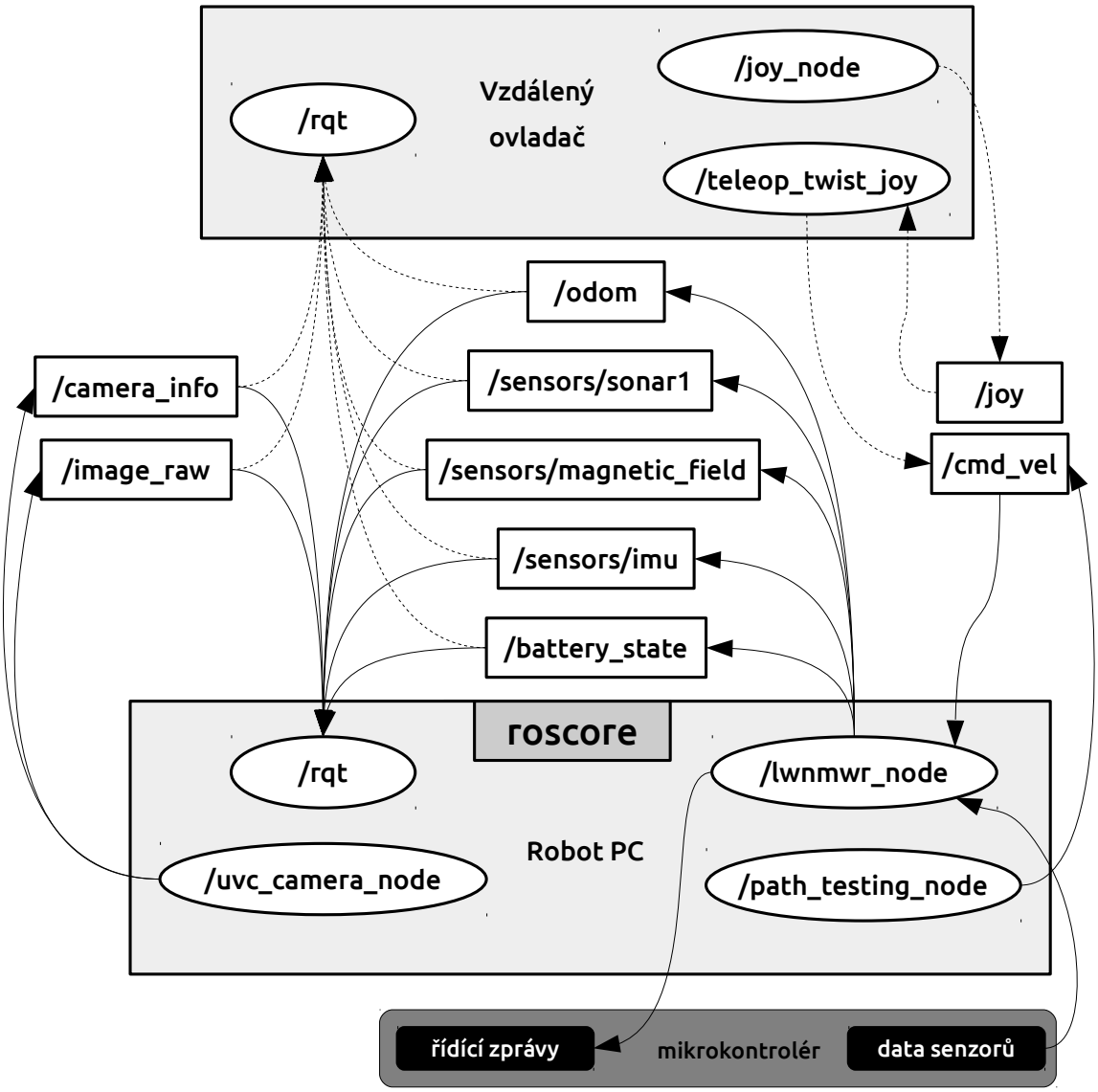
Program `rqt` je spuštěn jak na notebooku robota, tak na notebooku představující vzdálený ovladač. Na obou zařízení je tedy k dispozici grafický výstup senzorických dat.

path_testing

Balík obsahuje naprogramovaný uzel `path_testing_node`, který obsahuje několik předem připravených tras. Tyto trasy jsou definovány sekvencí řídících vektorů, které jsou odesílány na téma `\cmd_vel`. Robot tedy po spuštění tohoto uzlu samostatně (bez nutnosti ručního řízení) projede definovanou trasu.

7.3.4 Celkové schéma software

Obrázek 7.3 zachycuje kompletní schéma všech běžících komponent ROS a jejich vazby. Mimo to zobrazuje i napojení na řídící a senzorovou jednotku PIC, takže pokrývá všechny úrovně softwaru robota.



Obrázek 7.3: Schéma vazeb ROS uzlů a témat

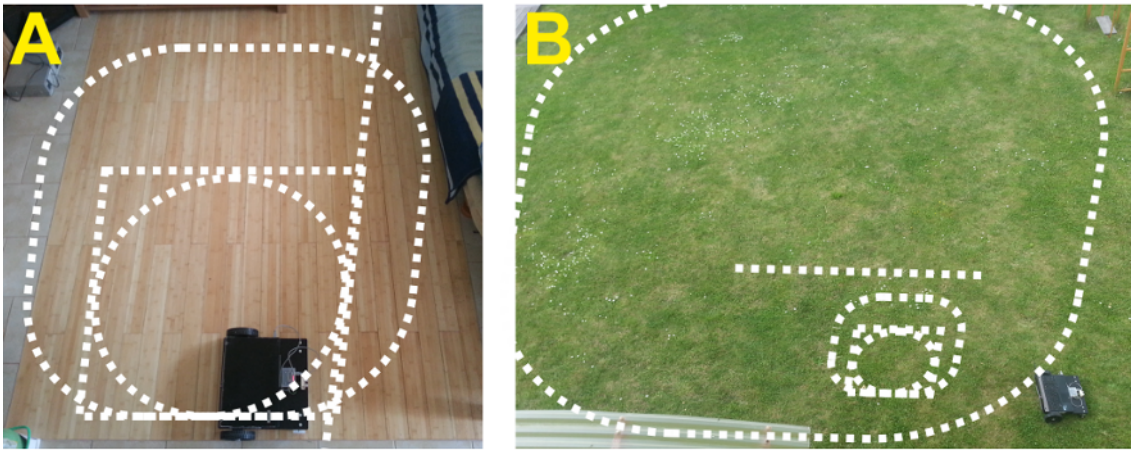
Kapitola 8

Experimenty

S vyrobeným robotem bylo provedeno mnoho experimentů, které byly převážně cíleny na ověření jízdních schopností robota. Tyto schopnosti jsou jednak určeny jeho podvozkem a také způsobem řízení jeho aktuátorů. Kromě toho byla experimentálně ověřena správnost dat poskytovaných řídící a senzorovou jednotkou, založenou na mikrokontroléru PIC. Popis a výsledky vybraných experimentů popisuje tato kapitola.

8.1 Testovací prostředí

Experimenty vždy probíhaly ve dvou zcela odlišných prostředích – vnitřním a vnějším. Jako vnitřní prostředí posloužil běžný pokoj s plovoucí podlahou. Použitelná plocha tohoto pokoje odpovídala čtvereci s délkou hrany 2,5 m. Experimenty ve vnějším prostředí probíhaly na zahradě s travnatým zvlněným povrchem. Zde byla k dispozici čtvercová plocha o rozměru 10 x 10 m. Fotografie testovacích prostředí jsou na obrázku 8.1.



Obrázek 8.1: Prostředí ve kterém byly prováděny experimenty – (A) vnitřní, (B) vnější.

V obou použitých prostředích bylo pro každý typ pohybu připraveno jednotné startovní stanoviště robota a několik záchytných bodů. Tyto body označovaly určené pozice, které měl v rámci vykonávaného pohybu robot projet. Od těchto bodů byla měřena odchylka skutečné trasy robota. Trasy pohybů, které měl robot vykonávat při níže uvedených experimentech

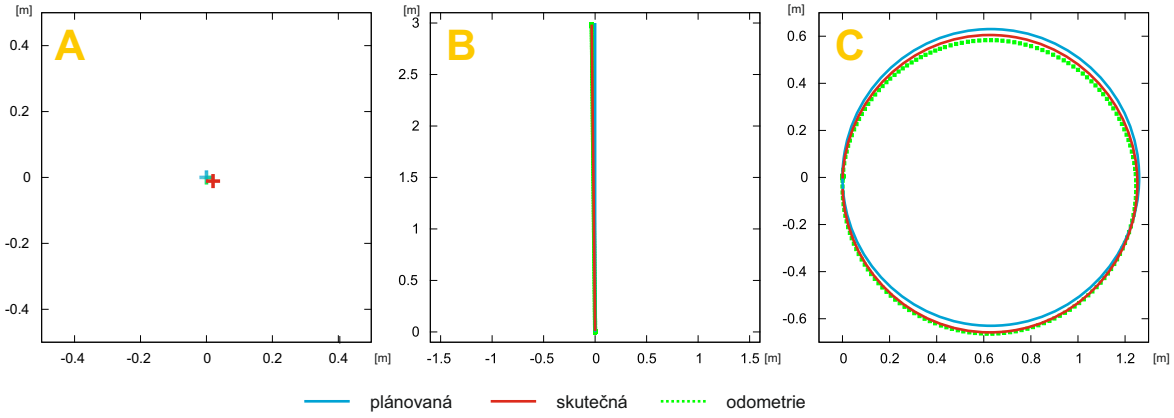
jsou představovány tečkovanými čarami na obrázku 8.1.

8.2 Základní pohyby

Mezi základní pohyby diferenciálně řízeného robota lze zařadit otáčení na místě, pohyb po přímce a pohyb po kružnici.Otáčení na místě je realizováno řídícím vektorem s jedinou definovanou složkou, kterou je úhlová rychlost otáčení v ose z. Pro pohyb po přímce je v řídícím vektoru definována pouze lineární rychlost ve směru osy x. A kruhový pohyb kombinuje obě zmíněné složky řídícího vektoru, což znamená, že je kombinací otáčení se na místě a pohybu po přímce.

Přesné vektory pohybů testovaných při tomto experimentu byly následující:

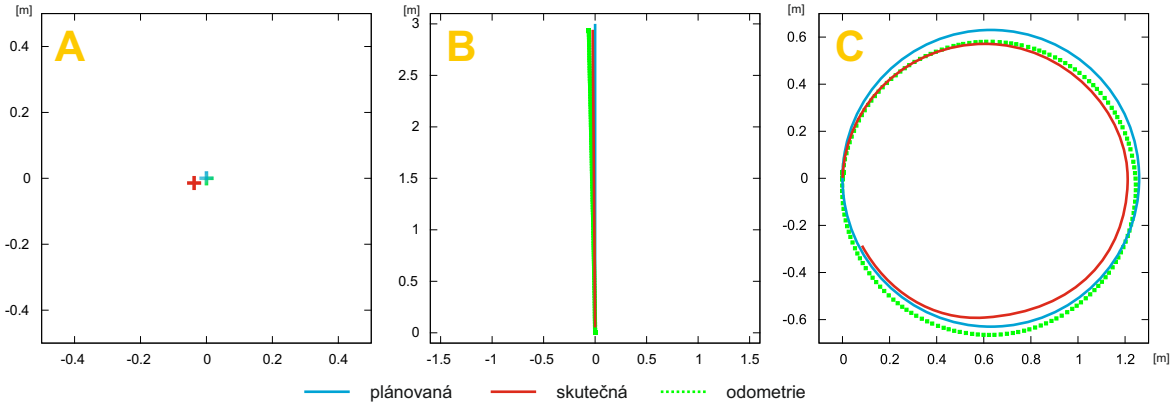
- **otáčení se na místě (360°)** - lineární rychlost - (0, 0, 0) $[m/s]$, úhlová rychlost (0, 0, 0.5236) $[rad/s]$, délka jízdy 12 s.
- **3 metry rovně** - lineární rychlost - (0.25, 0, 0) $[m/s]$, úhlová rychlost (0, 0, 0) $[rad/s]$, délka jízdy 12 s.
- **kruh o průměru 1,2732 metru** - lineární rychlost - (0.25, 0, 0) $[m/s]$, úhlová rychlost (0, 0, 0.3927) $[rad/s]$, délka jízdy 16 s



Obrázek 8.2: Výsledné grafy pro vnitřní prostředí – (A) otáčení se na místě, (B) 3 metry rovně, (C) kruh.

V každém grafu na obrázku 8.2 jsou vždy tři křivky/body, které značí: naplánovanou trasu (modrá), reálnou trajektorii robota (červená) a odometrii (zelená). Křivka reálné trajektorie je průměrem pěti měření.

V grafu otáčení se na místě je vidět, že podle odometrie se robot z počáteční pozice nepohnul, zatímco ve skutečnosti se při otáčení robot nedotočil a posunul o 2 centimetry vpravo. V grafu pohybu rovně se všechny tři křivky v podstatě neliší. Robot ve skutečnosti ujel 2,98 m, což je zapříčiněno drobným prokluzem kol při rozjezdu robota. Stejně tak tento rázový prokluz způsobil odchylku 4,5 cm u kruhového pohybu. Je vidět, že odpor válce nemá na rovném a hladkém povrchu téměř žádný negativní vliv na přesnost pohybu robota. Stejně tak prokluz poháněných kol je v podstatě nulový. Řízení lze tedy označit za velmi přesné (odchylka 1%) a stejně tak i odometrii.



Obrázek 8.3: Výsledné grafy pro vnější prostředí – (A) otáčení se na místě, (B) 3 metry rovně, (C) kruh.

Výsledky plynoucí z grafu na obrázku 8.3 jsou samozřejmě horší. Pohyb v zatravněném a nerovném povrchu má značný vliv jednak na tření válce a také na prokluz kol. To dokazuje i odometrie, která je vypočítávána z otáček kol, a ve všech případech vychází lépe než reálná trajektorie (porovnáváno s naplánovanou trasou). Odpor válce byl patrný z průběhu experimentu otáčení se na místě, kdy se po jeho dokončení nacházel ve stejné pozici, ale nedotočil se do celých 360° (otočil se pouze o 337°). Odchylka jízdy po kruhu je asi 7% a odchylka při jízdě vpřed je 5%.

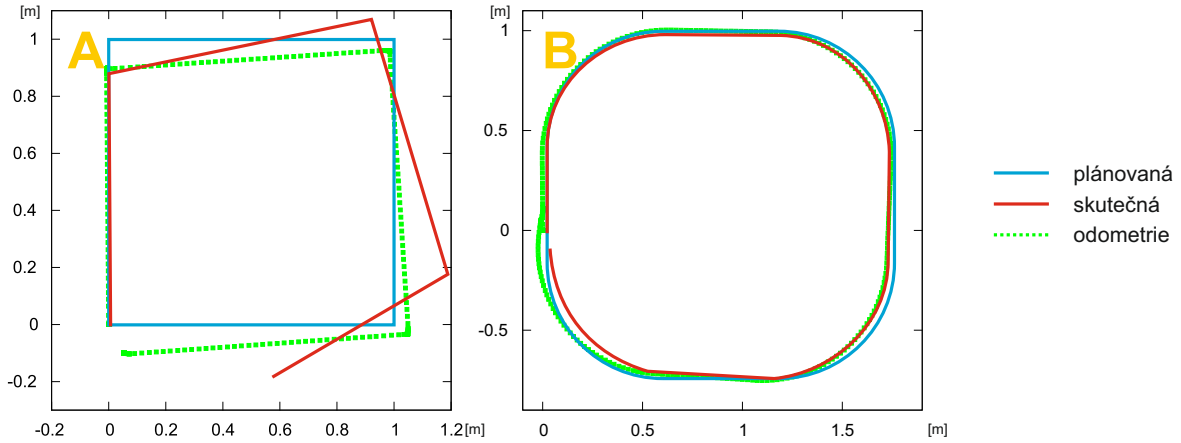
8.3 Složené pohyby

Po jednoduchých pohybech bylo experimentováno se složenými. Jejich trasa, kterou musí robot urazit, je delší a hlavně se při nich lépe projeví chyba pozice. Testy probíhaly na dvou rozdílných trajektoriích:

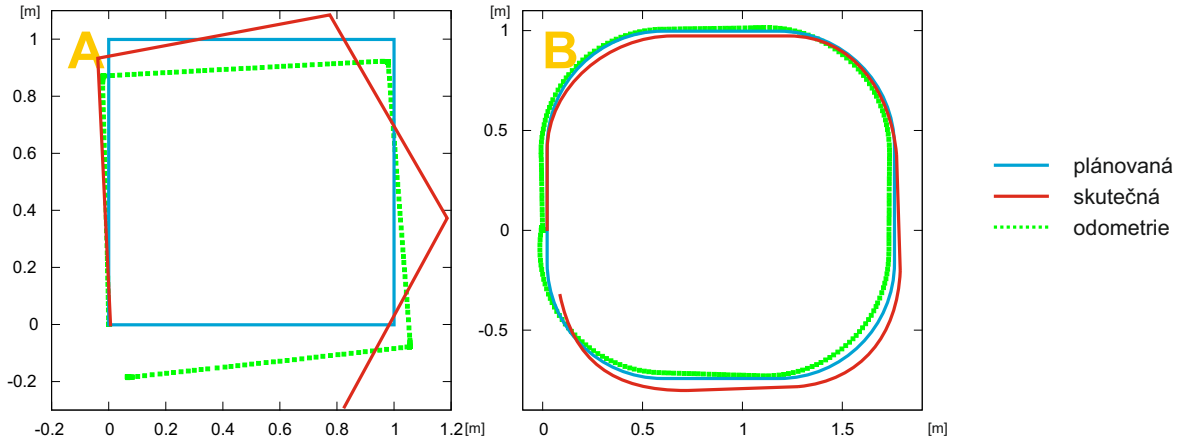
- **čtverec o hraně 1 metr** - 4 s jízdy rychlostí 0,25 m/s (pro jízdu po délce hrany), otáčení o 90° na místě za 4 s.
- **čtverec s kulatými rohy** - jízda 2 s rychlostí 0.25 m/s po rovinách, zatáčení kombinací dopředné rychlosti 0,25 m/s a úhlové rychlosti 0,3927 rad/s.

Při porovnání grafů na obrázku 8.4 je zřetelné, že otáčení na místě značně zhoršuje přesnost výsledné trasy. Zatímco u čtverce s kulatými rohy je odchylka maximálně 1%, tak u přesného čtverce, kde dochází k otáčení robota na místě, je odchylka 16,5%. To je způsobeno třením válce. Odometrie, vycházející z otáček kol, totiž vychází s podstatně menší odchylkou. To dokazuje i předchozí tvrzení, že nepřesnost odometrie vychází hlavně z úvodního protočení kol při rozjezdu. Řízení robota ve vnitřním prostředí lze tedy považovat za velmi přesné (odchylka do 1%), pokud je minimalizováno otáčení na místě.

Výsledky zachycené v grafech na obrázku 8.5 ukazují pro čtverec (s otáčením na místě) výrazný nárůst odchylek jak u reálné trajektorie, tak už i u odometrie. Řízení robota pouze na základě otáček kol je při otáčení se na místě zcela nedostatečné. Pro čtverec s kulatými rohy ale výsledky nejsou tak špatné, a proto byly provedeny další experimenty ve venkovním prostředí s delší naplánovanou trasou.



Obrázek 8.4: Výsledné grafy pro vnitřní prostředí – (A) čtverec, (B) čtverec s kulatými rohy.

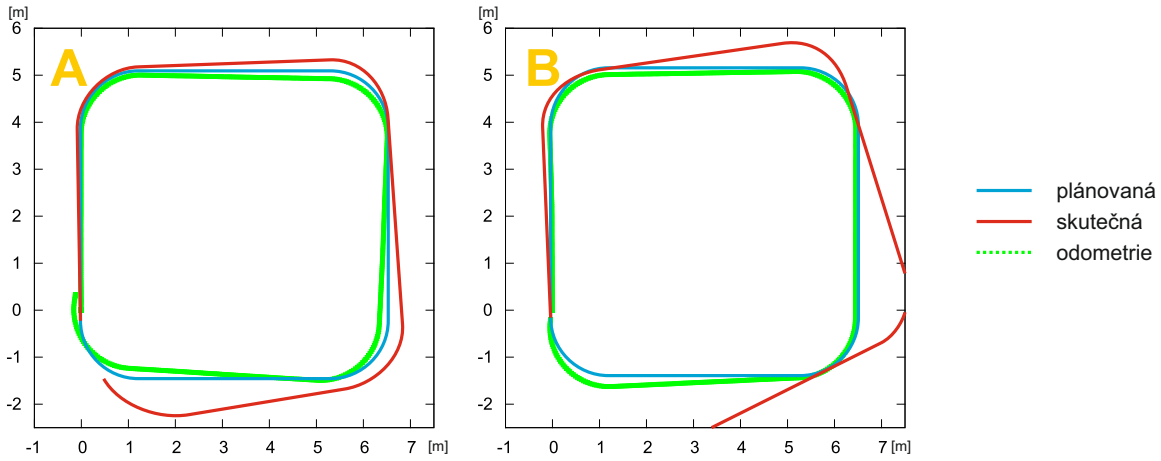


Obrázek 8.5: Výsledné grafy pro vnější prostředí – (A) čtverec, (B) čtverec s kulatými rohy.

S delší trasou se snížila přesnost. Ukázalo se však, že přesnost je silně závislá na tom, zda-li jede robot popředu (odchylka 6%) nebo pozadu (odchylka 11%), což potvrzují grafy na obrázku 8.6. Je tomu tak, protože při jízdě s válcem vepředu poháněná kola „tlačí“ přední část robota, jsou tedy více zatížená a nemají tendenci tolik prokluzovat. To vedlo k experimentu, kdy bylo na robota přidáno větší závaží, nicméně výsledky vykazovaly pouze zanedbatelné zlepšení.

8.4 Náročný terén

Jako náročný terén je v prostředí zahrady považován výjezd do kopce a sjezd z něj. Pro experimenty bylo využito souvislého kopce, jehož úhel stoupání/ klesání je 15°. Testovací trasa jízdy byla přímá a měla délku 3 metry. Nejprve byl proveden sjezd dolů, při němž robot přejel cílový bod v průměru o 12 centimetrů a neprojevila se závislost na směru jízdy. Při jízdě do kopce robot dosahoval průměrné vzdálenosti 2,84 metrů pokud jel válcem napřed.



Obrázek 8.6: Výsledné grafy pro vnější prostředí a delší naplánovanou trasu – (A) jízda vpřed (B) jízda vzad.

Pokud jel s válcem vzadu („táhl ho“), tak z pěti pokusů dvakrát uváznul a ve zbylých případech dosáhl průměrné vzdálenosti 2,63 metrů. To znamená odchylku 4% z kopce a 5,3% při jízdě do kopce válcem napřed a 12,3% při jízdě s válcem vzadu.

8.5 Data ze senzorů

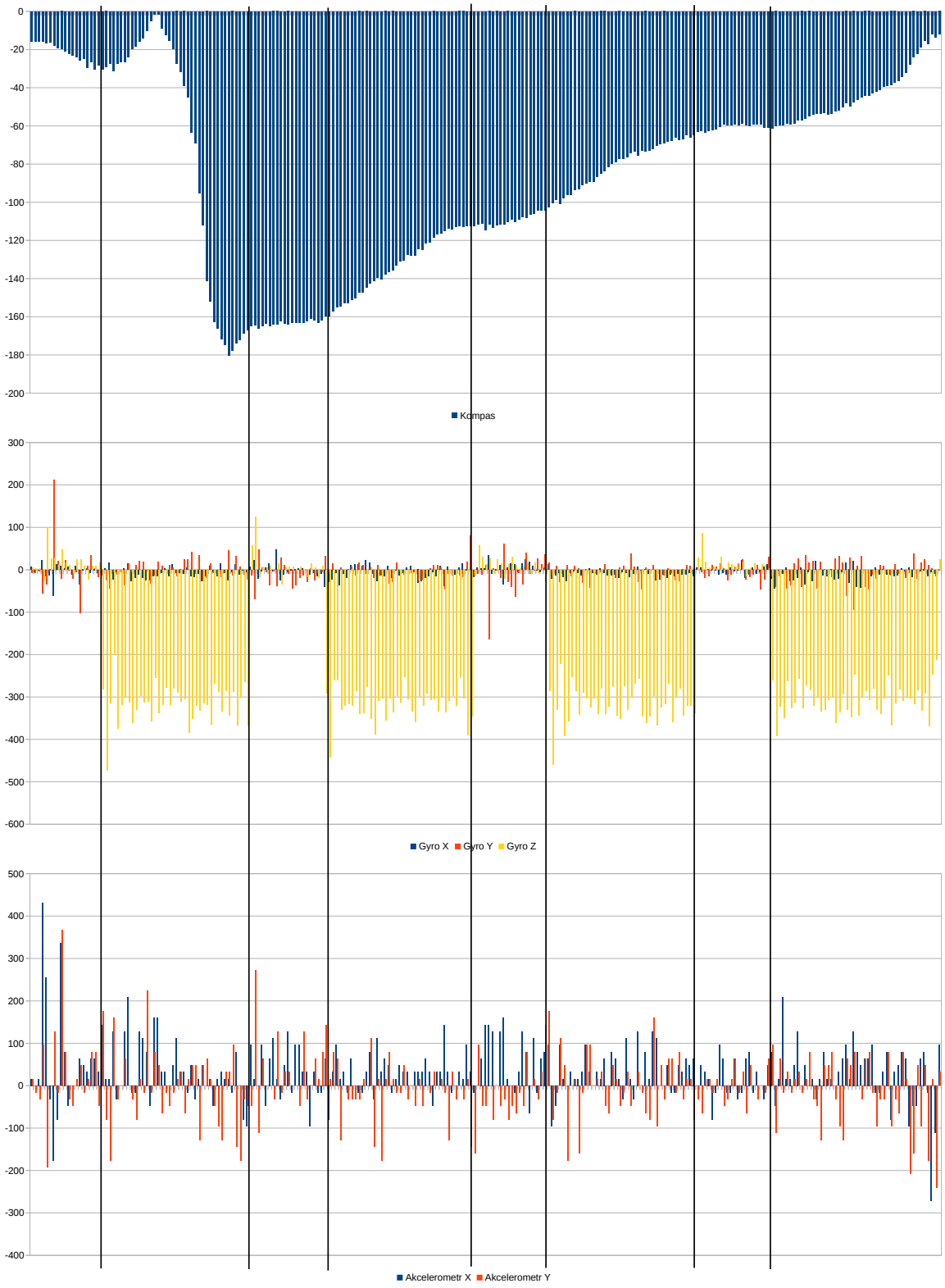
Poslední experimenty spočívaly v ověření funkčnosti senzorů a zhodnocení jejich využitelnosti pro řízení.

Nejprve byla provedena měření na kratší čtvercové trase (s kulatými rohy) ve vnitřním prostředí. Tato trasa byla vybrána zcela záměrně, jelikož obsahuje jak zatáčení, tak pohyb po rovině, který by měl být v naměřených datech sloužit jako „oddělovač“. Data, které poskytly senzory jsou uvedeny ve formě tří grafů na obrázku 8.7. Asi nejlepší výsledky ze všech senzorů poskytuje gyroskop, jelikož ve chvíli kdy se robot začne otáčet (ilustrováno oddělovacími čárami v ose y), tak naměří výrazné záporné hodnoty v ose z. Jeho reakce z klidového (nulového) stavu je okamžitá. Mimoto je hodnota naměřené úhlové rychlosti stabilně konstatní, což je opět dobře, jelikož zatáčení probíhá s konstantní rychlostí. Podobně dobře jsou na tom i data z kompasu. Na grafu je vidět čtyři konstantní části grafu, odpovídající rovinám, kdy se úhel jízdy nemění a tedy ani naměřená hodnota kompasu. A dále jsou zde čtyři lineární změny, které odpovídají zatáčkám. Při první zatáčce dojde k „prohození os“ kompasu, z kterých je vypočítávána výsledná hodnota, což se projeví náhlou změnou, kdy původně rostoucí hodnoty začnou opět klesat. Jako nejméně přesný, a proto nejméně užitečný, se projevil akcelerometr. V grafu, byla cíleně vynechána složka, která nabývá velmi vysokých hodnot, odpovídající tíhové síle. Hodnoty v osách sice nesou informaci o zatáčení a rovinách, ale ta je silně zašuměná a těžko rozlišitelná. U tohoto senzoru lze spolehlivě určit pouze to, zda-li se robot pohybuje nebo stojí.

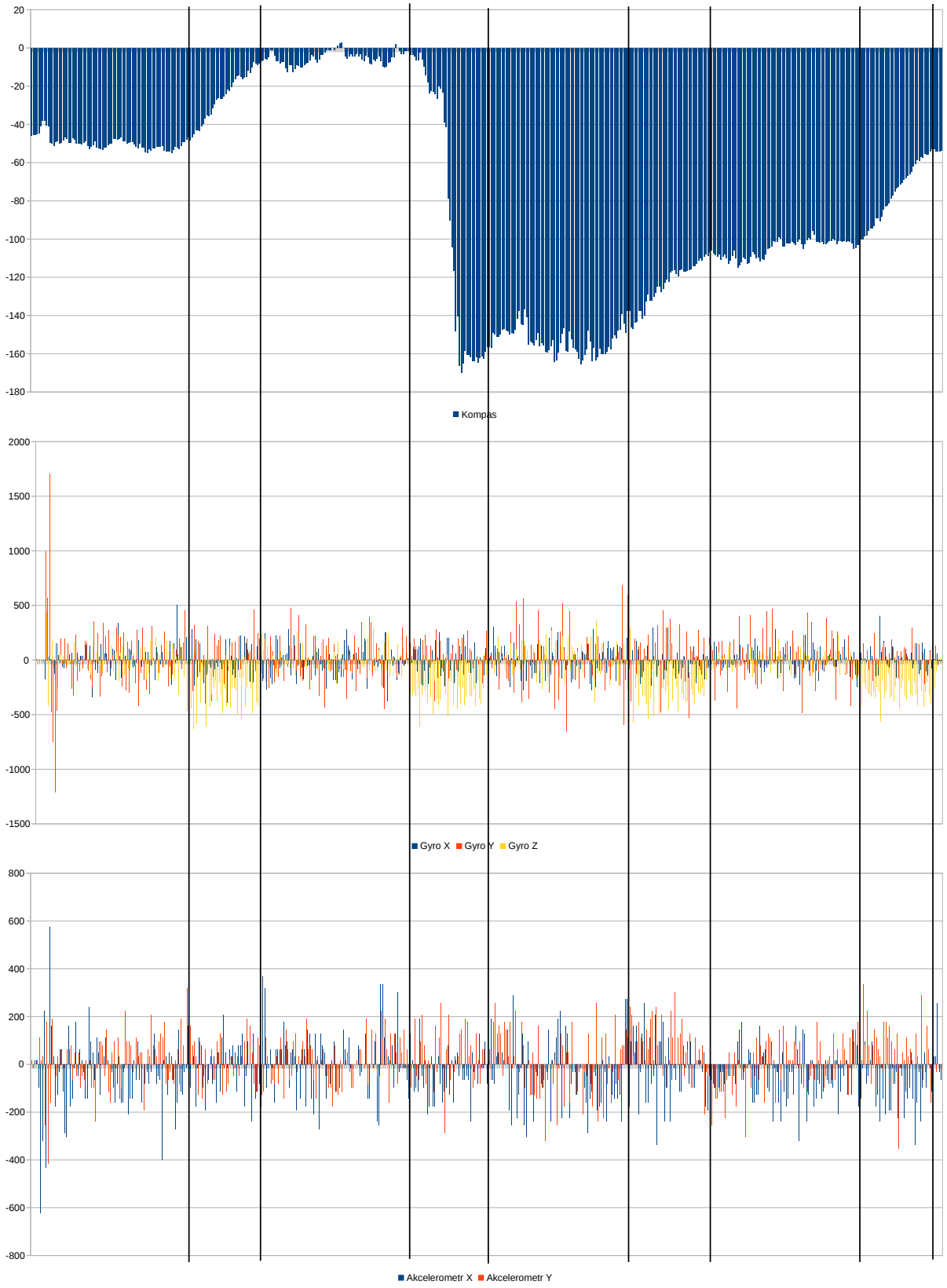
Stejný test senzorů byl ještě proveden v podmínkách vnějšího prostředí a delší trasy (stejného tvaru). Naměřená data jsou zobrazena v grafech na obrázku 8.8 a ukazují očekávetelný problém, kterým je výrazný šum způsobený jízdou po nerovném terénu. V datech z gyroskopu je stále poměrně dobře odlišitelné zatáčení, ale již nemá tak stabilní průběh. Nejhodnotnější data při tomto experimentu poskytoval kompas. Ten měl velmi podobnou

přesnost jako u přechozího testu. Dokonce lze naměřené hodnoty považovat za natolik přesně, že by bylo možné na jejich základě upravovat přesnost řízení při zatáčení, které je při použití enkodéru, jako jediné zpětné vazby, velmi nepřesné. Akcelerometr vrací o něco málo lepší výsledky než v předchozím testu, nicméně se stále jedná o velmi zašuměná data, na které se nelze příliš spoléhat.

U tohoto experimentu nebyla sledována absolutní přesnost dat senzorů, jelikož by to vyžadovalo jejich důkladnou kalibraci, což není obsahem této práce. Veškeré porovnání tedy probíhalo na základě relativních změn měření, které ale dávají zcela dostatečnou informaci pro porovnání vhodnosti či nevhodnosti senzoru pro řízení. Výsledkem je tedy to, že spojením dat z kompasu a gyroskopu by bylo možné výrazně zpřesnit, jinak problematické, zatáčení. A to bez závislosti na prostředí, ve kterém bude robot provozován.



Obrázek 8.7: Hodnoty senzorů měřené během jízdy ve vnitřním prostředí. Trajektorie jízdy byla čtverec s kulatými rohy.



Obrázek 8.8: Hodnoty senzorů měřené během jízdy ve vnějším prostředí. Trajektorie jízdy byla čtverec s kulatými rohy.

Kapitola 9

Závěr

Cílem této diplomové práce bylo vytvořit robota a jeho řídící a senzorovou jednotku založenou na mikrokontroléru PIC, implementovat firmware mikrokontroléru, napsat program zajišťující komunikaci mezi jednotkou a řídícím počítačem a připravit software pro vzdálené ovládání robota a zobrazování dat z jeho senzorů.

Prvním krokem byla výroba podvozku robota a jednotky PIC. Nejprve byl vyroben testovací prototyp, který odhalil nedostatečný výkon pohonné soustavy podvozku a chybějící senzory na jednotce PIC. V druhém prototypu tedy byla nahrazena pohonná soustava, a také byla přepracována jednotka PIC. Finální podvozek robota a řídící a senzorová jednotka vychází z druhého prototypu a opravuje již pouze drobné nedostatky. Celková cena výroby se pohybuje kolem 5000,- Kč.

Návrh a implementace firmware mikrokontroléru PIC24FJ64GB002, který je použit jako základ řídící a senzorové jednotky, vychází z technických dokumentací jednotlivých použitých součástek a mikrokontroléru samotného. Program zajišťuje řízení motorů přes modelářské regulátory na základě zpětné vazby od enkodérů. Pro tento účel je využito algoritmu PID regulace. Dále je implementována komunikace se senzory přes I^2C a USB komunikace s vyšší jednotkou řízení.

USB komunikace emuluje sériový port a je spojovacím prvkem jednotky PIC a ROS driveru. Robotický operační systém (ROS) zprostředkovává funkce vyššího řízení a vyžaduje program, který abstrahuje platformu robota. Tímto programem je právě ROS driver, který byl implementován. Přijímá řídící data vystavená na ROS tématu, transformuje je a odesílá jednotce PIC. Od ní naopak získává data ze senzorů a ty publikuje na ROS témata, která mohou využívat jiné uzly ROS.

Poslední programovou částí je vzdálené ovládání, které využívá existujících ROS balíků, jejichž nastavení a provázání s ROS driverem je popsáno a otestováno. Pomocí připraveného řešení je možné robota řídit pomocí gamepad ovladače připojeného na druhý bezdrátově připojený počítač. Na tomto počítači je mimo řízení ještě možné zobrazovat data ze senzorů robota.

Robot, řídící a senzorová jednotka a veškerý software byly podrobeny důkladným testům, aby byla ověřena funkčnost všech částí. Pro zhodnocení jízdních schopností a řízení robota bylo provedeno mnoho experimentů, které byly rozebrány a vyhodnoceny. Plyne z nich, že robot je schopen velmi přesného pohybu ve vnitřních prostorách. Ve venkovních je řízení, závislé pouze na enkodérech, nedostatečně přesné. To by bylo možné zpřesnit na základě dat ze senzorů. Ohodnocení využitelnosti jednotlivých senzorů, které jsou součástí robota, je v experimentech také diskutováno.

9.1 Možnosti dalšího vývoje

Vzhledem k využití ROS jsou možnosti dalšího vývoje velmi široké, protože lze snadno využít existujících balíků, nemělo by být větším problémem zprovoznění lokalizace, mapování a případného autonomního chování robota. Vzhledem k ceně může sloužit pro studenty a jejich akademické projekty.

Tento konkrétní robot bude dále vyvíjen s cílem vyrobit automatickou sekačku trávníků. Dalším krokem rozšiřujícím tuto práci bude zpřesnění odometrie právě na základě senzorů, které byly vyhodnoceny jako přínosné.

Literatura

[1] NOVÁK, P.: *Mobilní roboty: pohony, senzory, Řízení*. BEN, 2005, ISBN 9788073001414.

[2] ČERNOHORSKÝ, J.: *ZÁKLADY ROBOTIKY: Úvod do mobilní robotiky* [online]. [cit. 2015-01-08]. Dostupné z: http://www.fm.tul.cz/esf0247/index.php?download=388

[3] SIEGWART, I., R.; NOURBAKHSH: *Introduction to Autonomous Mobile Robots*. Bradford Company, 2004, ISBN 026219502X.

[4] ARVIN, F.; SAMSUDIN, K.; NASSERI, M.: Design of a differential-drive wheeled robot controller with pulse-width modulation. In *Innovative Technologies in Intelligent Systems and Industrial Applications, 2009. CITISIA 2009*, July 2009, s. 143–147. Dostupné z: http://www6.in.tum.de/Main/Publications/5224223.pdf

[5] ŘEZÁČ, K.: *Krokové motory: princip funkce, metody řízení* [online]. 2002-10-28 [cit. 2015-01-10]. Dostupné z: http://robotika.cz/articles/steppers/cs

[6] MARZI, H.: Using AC Motors in Robotics. In *International Journal of Advanced Robotic Systems, Vol. 4, No. 3 (2007)*, 2007, s. 365–370. Dostupné z: http://cdn.intechopen.com/pdfs-wm/4243.pdf

[7] FOJT, B.: *Modul pro řízení DC motorů* [online]. 2010. 57 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=29635

[8] WINKLER, Z.: *Řízení pohybu: a co je to vlastně ta zpětná vazba* [online]. 2007-12-05 [cit. 2015-01-10]. Dostupné z: http://robotika.cz/guide/control/cs

[9] G.C.Y., L. Y. A. K. C.: PID control system analysis and design. *IEEE Control Systems Magazine*, ročník 26, č. 1, Únor 2006: s. 32–41. Dostupné z: http://eprints.gla.ac.uk/3815/

[10] ŘÍHA, Z.: *Nastavení PID* [online]. 2010. [cit. 2015-05-15]. AMiT, spol. s r.o. Dostupné z: http://www.amit.cz/support/cz/aplikacni_poznamky/ap0042_cz_01.pdf

[11] EVERETT, H. R.: *Sensors for Mobile Robots: Theory and Application*. A. K. Peters, Ltd., 1995, ISBN 1-56881-048-2.

[12] encoder., R.: *Wikipedia: the free encyclopedia* [online]. 2014-12-13 [cit. 2015-01-02]. Wikimedia Foundation. Dostupné z: http://en.wikipedia.org/wiki/Rotary_encoder

[13] ORSÁG, F.: *Studijní opora do předmětu ROB*. 2006. 131 s.

[14] KING, A.: *Inertial Navigation – Forty Years of Evolution* [online]. 1998 [cit. 2014-12-20]. Dostupné z: http://imar-navigation.com/downloads/papers/inertial_navigation_introduction.pdf

[15] ROZMAN, J.: *Efektory a senzory* Prezentace z předmětu ROB. 2014.

[16] *Low-Level Robot Control* [online]. 2007 [cit. 2015-01-05]. Dostupné z: http://www.engr.mun.ca/~dpeters/6806/lectures/Robot_Lecture1.ppt

[17] stránky, W.: ROS [online]. [cit. 2015-05-15]. Dostupné z: http://www.ros.org

[18] *3-Axis Digital Compass IC HMC5883L* [online]. 2013. [cit. 2014-12-15]. Honeywell International Inc. Dostupné z: http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf

[19] *LSM330DL: Linear sensor module 3D accelerometer sensor and 3D gyroscope sensor* [online]. 2011. [cit. 2014-12-15]. STMicroelectronics. Dostupné z: http://media.digikey.com/pdf/Data%20Sheets/ST%20Microelectronics%20PDFS/LSM330DL.pdf

[20] *Dual H-bridge motor driver* [online]. 2013. [cit. 2014-12-20]. Texas Instruments Inc. Dostupné z: https://www.pololu.com/file/0J534/drv8833.pdf

[21] *PIC24FJ64GB004 Family Data Sheet* [online]. 2010. [cit. 2014-11-28]. Microchip Technology Inc. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/39940d.pdf

[22] *Univerzální DC regulátory obousměrné / jednosměrné* [online]. 2014. [cit. 2015-05-05]. www.dsys.cz. Dostupné z: http://dsys.cz/downloads/MDE_cz_51.pdf

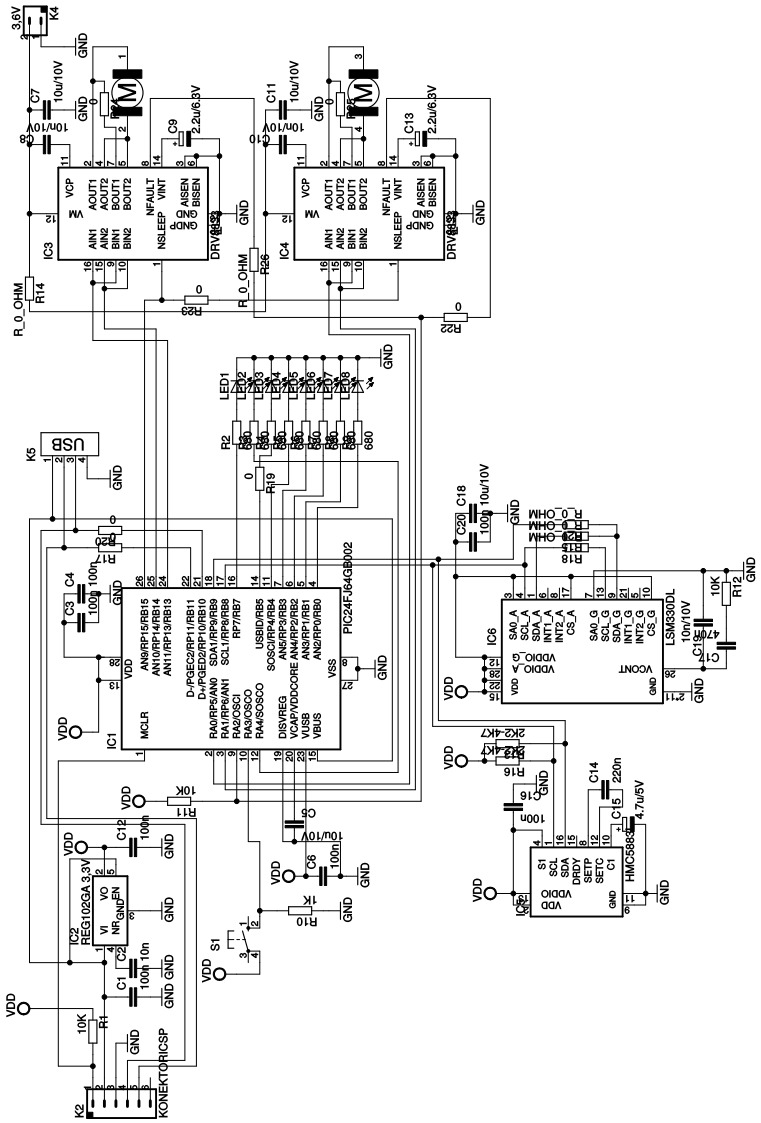
[23] *TLE4906K / TLE4906L: High Precision Hall Effect Switch* [online]. 2009. [cit. 2015-01-11]. Infineon Technologies AG. Dostupné z: http://www.infineon.com/dgdl/TLE4906+Data+Sheet+V2.0.pdf?folderId=db3a30431f848401011facc1c83b4674&fileId=db3a30431f848401011fbc8bb25c6373

[24] *SRF08 Ultra sonic range finder* [online]. [cit. 2015-05-05]. Devantech Ltd. Dostupné z: http://www.robotshop.com/media/files/pdf/devantech-srf08-ultrasonic-range-finder-specifications.pdf

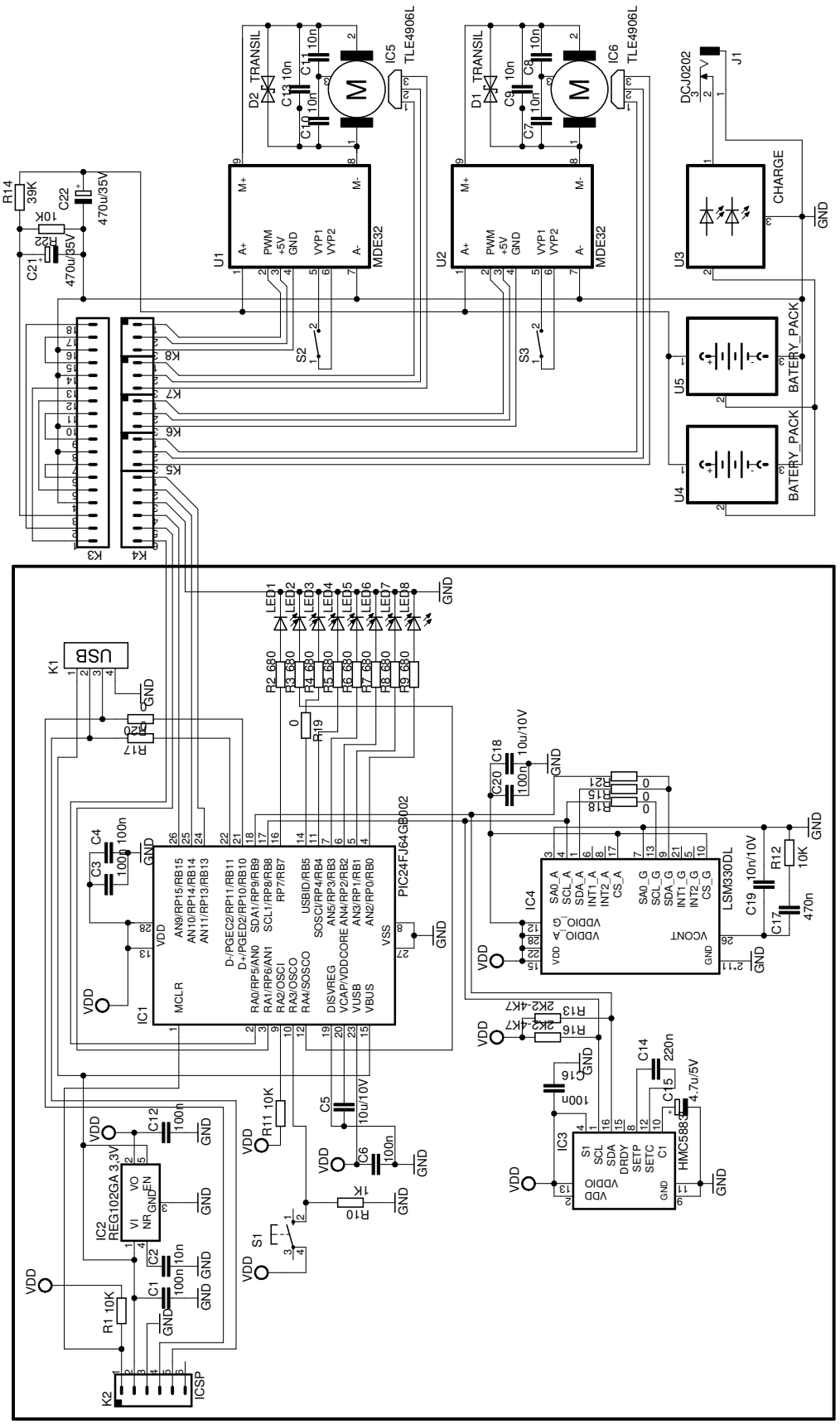
[25] T, H.: Kinematics Equations for Differential Drive and Articulated Steering. 2011, iSSN: 0348-0542.

Příloha A

Elektrická schémata



Obrázek A.1: Schéma zapojení prvního prototypu řídící a senzorové jednotky.



Obrázek A.2: Schéma zapojení druhého prototypu řídící a senzorové jednotky.