České vysoké učení technické v Praze
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

Robot HERO

Praha, 2010 **Ondřej Zeman**

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 14. 5. 2010

podpis

Poděkování

Děkuji především vedoucímu mé diplomové práce doc. Ing. Jiřímu Bayerovi, CSc. za vedení práce, cenné připomínky a za vstřícný přístup. Také bych chtěl poděkovat panu Ladislavu Čmelíkovi za úpravu trnu ruky.

Abstrakt

Cílem této diplomové práce je doplnit senzorický systém a řídicí software robotu HERO, který byl modernizován v rámci bakalářských prací realizovaných v letech 2007 a 2008.

Výbava robotu byla rozšířena o laserový skener Hokuyo a senzory vzdálenosti Sharp. Dále byly provedeny úpravy software s ohledem na navigaci v prostoru a zpracování dat poskytnutých senzory. Výchozí stav robotu také vyžadoval finální úpravy mechanických dílů, které bylo potřeba realizovat.

Tato práce navazuje na bakalářské práce Jiřího Zemánka [1] a Ondřeje Šantina [2] realizované v roce 2007 a bakalářskou práci Ondřeje Zemana [3] realizovanou v roce 2008.

Abstract

The purporse of this diploma thesis is to integrate new sensors into robot HERO's sensing system. The robot's current hardware and software was developed as outcome of Bachelors' thesis in 2007 and 2008.

The robot was given new Hokuyo laser rangefinder and Sharp optical distance sensors. Also the control software had to be upgraded to be able to process information given by new sensors. Initial state of the robot also required some mechanical work and some new parts to be manufactured.

This work follows Bachelors' thesis of J. Zemánek [1] a O. Šantin [2] from 2007 and Bachelors thesis of O. Zeman [3] from 2008.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Ondřej Zeman**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Robot HERO**

Pokyny pro vypracování:

1. Dopracujte mechanickou a elektrickou koncepci mobilního robota HERO pro jeho pohyb v prostoru s překážkami s možností identifikace specifických objektů a jejich přemísťováním pomocí ruky robota.
2. Navrhněte a realizujte úpravu a popř. doplnění senzorického systému mobilního robota pro zpracování měřených dat na vyšší úrovni vhodné zejména pro odhad polohy, identifikaci překážek a specifických objektů a pro následné mapování a řízení jeho pohybu.
3. Realizujte základní řízení pohybových os robota pro zadanou globální úlohu jeho činnosti. Ověřte funkci inovovaného senzorického systému při řízení pohybu mobilního robota po předem definované dráze v prostoru s překážkami.
4. Začleňte rozšířené funkce řízení robota do programového vybavení nadřazeného palubního systému s PC104.
5. Vypracujte podrobnou dokumentaci dodatečných změn robota.

Seznam odborné literatury:

dokumentace robota Hero, BP Zemánek, Šantin 2007, BP Zeman 2008

Vedoucí: Doc.Ing. Jiří Bayer, CSc.

Platnost zadání: do konce letního semestru 2010/2011

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 22. 12. 2009

Obsah

Seznam obrázků **viii**

Seznam tabulek **ix**

1 Úvod **1**

1.1 Účel projektu 1

1.2 Historie projektu 1

1.3 Cíl diplomové práce 2

2 Mechanické úpravy robotu **3**

2.1 Zatáčení hnacího kola 3

2.2 Ruka robotu 5

2.2.1 Uchycení v rameni 5

2.2.2 Kabeláž 6

3 Senzorický systém **8**

3.1 Skener Hokuyo URG-04LX 8

3.1.1 Připojení 9

3.1.2 SW rozhraní 10

3.1.3 Napájení skeneru 11

3.2 Senzory Sharp GP2D12 12

3.2.1 Připojení 13

3.3 Stereovidění 14

4 Kinematika **15**

4.1 Přímá kinematická úloha 15

4.1.1 Denavitova-Hartenbergova notace 15

4.1.2 Řešení přímé kinematické úlohy 18

4.2 Inverzní kinematická úloha 21
4.2.1 Řešení inverzní kinematické úlohy 22
4.3 Odometrie 25

5 Mapa prostředí **28**
5.1 Sestavování mapy a lokalizace 28
5.2 CoreSLAM 30
5.3 Hledání cesty v mapě 33

6 Úpravy SW vybavení **35**
6.1 Úpravy řídicí aplikace 36
6.1.1 Vlákna procesu 37
6.1.2 Přehled funkcí navigačního modulu 39
6.1.3 Komprese mapy 40
6.2 Ovládací aplikace 41

7 Aplikační list **44**
7.1 Komunikace a řízení 44
7.2 Nabíjení akumulátorů 45

8 Závěr **46**
8.1 Řešení práce 46
8.2 Možná vylepšení 46

Literatura **49**

A Obsah přiloženého DVD **I**

Seznam obrázků

2.1 Nákres mechanismu natáčení hnacího kola 4
2.2 Fotografie mechanismu natáčení hnacího kola 5
2.3 Upravené uchycení ramene robotu 6
2.4 Kabeláž vedoucí do převodovky zápěstí 7

3.1 Skener Hokuyo URG-04LX 9
3.2 Celkový pohled na montáž skeneru na hlavě robotu. Nalevo napájecí zdroj skeneru, napravo USB hub 10
3.3 Schéma zapojení regulátoru napětí 11
3.4 Princip funkce senzoru GP2D12 12
3.5 Namontovaný senzor GP2D12 - zadní strana 13

4.1 Rozmístění jednotlivých souřadných systémů, celkový pohled 18
4.2 Rotační kloub J_1, pohled shora 19
4.3 Rotační kloub J_2, pohled v ose rotace 19
4.4 Rotační kloub J_4, pohled v ose rotace 20
4.5 Rotační kloub J_5, pohled v ose rotace 20
4.6 Pohled shora 23
4.7 Pohled ze strany (rovina pohledu kolmá na osu kloubu J_2) 24
4.8 Odometrický model - pohled shora 25
4.9 Měření chyby odometrie 27

5.1 Princip funkce použitého algoritmu SLAM 32
5.2 Příklad mapy vytvořené robotem (výřez) 33
5.3 Cesta v mapě nalezená pomocí Theta* 34

6.1 Struktura řídicí aplikace 36
6.2 Vlákna běžící v programu a tok dat 38
6.3 Grafické uživatelské rozhraní pro vzdálené ovládání robotu Hero 42

Seznam tabulek

3.1 Technické parametry skeneru URG-04LX 9
3.2 Technické parametry senzoru GP2D12 12
3.3 Připojení senzorů Sharp k jednotkám EPOS 13
3.4 Připojení senzorů Sharp - barvy vodičů 13

4.1 D-H parametry 21
4.2 Rozměry robotu 21
4.3 Chyby odometrie 26

6.1 Přehled výzev 42
6.2 Přehled odpovědí 43

Kapitola 1

Úvod

1.1 Účel projektu

Katedra řídicí techniky zadala k vypracování projekt modernizace robotu HERO vlastněného katedrou tak, aby v budoucnu mohl sloužit jako výuková pomůcka a k prezentacím katedry.

Robot by měl být určen k výuce distribuovaných systémů, některých oborů umělé inteligence (např. orientace v prostředí) a k demonstraci moderních prostředků sofistikovaného řízení a autonomního rozhodování. Celé zařízení je myšleno jako pomůcka pro vzdálenou výuku a mělo by být plně řiditelné přes fakultní počítačovou síť.

1.2 Historie projektu

Mobilní robot Hero-1 (typové označení ET-18) začal být vyráběn americkou firmou Heathkit v roce 1982. Byl prodáván buď jako stavebnice nebo jako sestavený komplet. Určen byl především pro školy jako prostředek pro výuku a pro amatérské příznivce robotiky. V současné době lze na Internetu nalézt stále velice početnou komunitu fanoušků a vlastníků tohoto robotu. Samozřejmě většinou ve Spojených státech.

Tento původní model byl vybaven na svoji dobu relativně výkonným řídicím systémem s procesorem Motorola 6808 s 4kB RAM.

Robot se programoval pomocí vestavěné hexadecimální klávesnice s LED displejem nebo přes rozhraní RS-232. K orientaci robotu sloužil ultrazvukový dálkoměr a senzory přiblížení. Robot se pohyboval na tříkolové základně. Pohon zajišťoval stejnosměrný motor a zatáčení bylo řízeno krokovým motorem. Celý řídicí systém byl modulární, takže se

daly dokupovat další doplňkové moduly obohacující Hero-1 o nové funkce.

Jedním z těchto doplňujících rozšíření byla i ruka s pěti stupni volnosti zakončená chapadlem pro uchopování lehčích předmětů. Jako pohony jednotlivých os byly použity krokové motory, v té době populární díky jednoduchému digitálnímu řízení. Náš robot Hero-1 byl takovouto rukou vybaven.

Robot Hero-1 vlastněný Katedrou řídicí techniky Českého vysokého učení technického v Praze byl poprvé modernizován v roce 1985. Tuto modernizaci provel Ing. F. Vaněk spolu s Ing. Páchou. Ti předělali řídící systém v rámci svých diplomových prací.

V listopadu 2006 projekt znovu odstartovaly bakalářské práce Jiřího Zemánka [1] a Ondřeje Šantina [2]. Jejich výsledkem byla obnova veškerého elektrického vybavení robotu a zprovoznění základního dálkového ovládání přes počítačovou síť pomocí protokolu TCP. Řídicí systém tedy v současnosti sestává z hlavního počítače třídy PC/104+ a řídicích jednotek EPOS, které jsou součástí zmodernizovaných pohonů. Tyto komponenty jsou spolu propojeny pomocí průmyslové sběrnice CANopen. Řídicí počítač je vybaven procesorem AMD Geode 500Mhz a je na něm provozován operační systém GNU/Linux pod kterým běží uživatelské aplikace řídící robot. Tato konfigurace je o několik řádů výkonnější než předcházející systém, takže robot má dostatek výpočetního výkonu pro autonomní rozhodování, které by se mělo řešit v některém z následujících projektů týkajících se tohoto robotu.

Na tyto práce navázala bakalářská práce Ondřeje Zemana [3] realizovaná v roce 2008. Cílem této práce byla rekonstrukce mechanického manipulátoru robotu. Výsledkem bylo osazení ruky robotu pohony Maxon, stejnými jako byly použity pro pojezd a zatáčení robotu. Tyto pohony byly připojeny na sběrnici CANopen a dále tak rozšířily distribuovaný řídicí systém robotu. V rámci této práce bylo taktéž ukázáno, že díky nevhodné mechanické konstrukci ruky je inverzní kinematická úloha neřešitelná a bude potřeba dalších mechanických úprav.

1.3 Cíl diplomové práce

Cílem této práce je připojení nových senzorů a jejich integrace do stávajícího HW a SW robotu. Dále se předpokládá rozšíření funkcionality řídicího SW o schopnost sestavování mapy prostředí, vyhýbání se překážkám a identifikace specifických objektů pomocí dostupných senzorů a jejich následné uchopení pomocí mechanického manipulátoru. K dosažení tohoto cíle bude také potřeba analyticky vyřešit inverzní kinematickou úlohu a s tím související mechanickou přestavbu manipulátoru.

Kapitola 2

Mechanické úpravy robotu

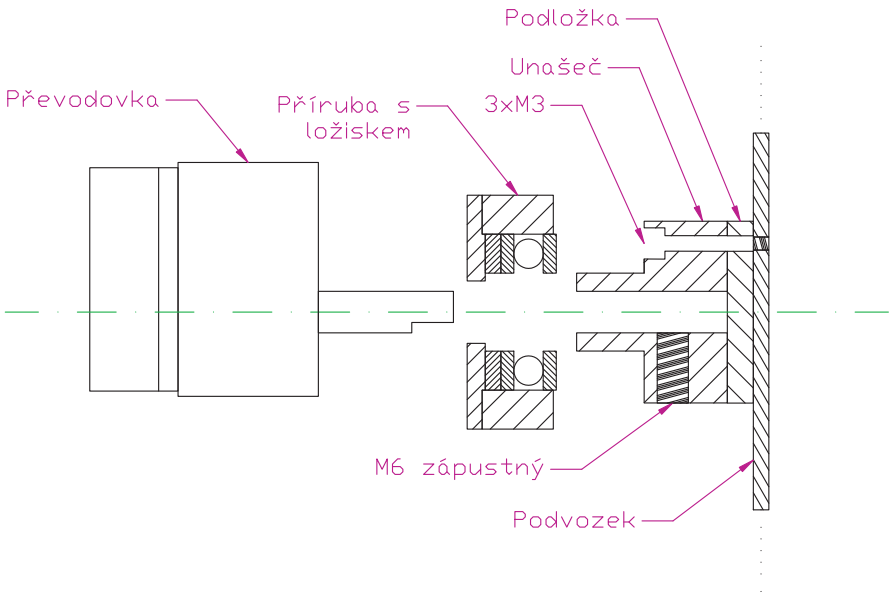
Výchozí stav robotu neodpovídal požadavkům na přesnost kladeným odměřováním odometrie a rovněž i konstrukce manipulátoru byla z hlediska jeho kinematické struktury nevhodá. Bylo tedy nutné provést několik úprav.

2.1 Zatáčení hnacího kola

Uchycení předního (poháněného) kola bylo předěláno v rámci bakalářské práce J. Zemánka v roce 2007. Toto uchycení sestávalo z příruby vytvořené pro nově instalovaný motor, z axiálního ložiska a z unašeče předního kola, který spojoval výstupní hřídel pohonu s uložením předního kola. Celé ústrojí je detailně rozkresleno v [1]. Problémem původního unašeče bylo především nedostatečné dimenzování zápustného šroubu, který zajišťoval dotažení unašeče proti hřídeli motoru. Tento šroub byl navíc vyrobený z relativně měkkého materiálu. Samotný unašeč byl také vyroben ze stejně měkkého materiálu. Kombinace malého průměru zápustného šroubu spolu s nevhodným materiálem způsobila, že nebylo možné kolo na hřídeli motoru pořádně dotáhnout, resp. po krátkém používání došlo opět k uvolnění spoje. Toto mělo za následek vůli v otáčení předního kola ve velikosti přibližně 20 - 30°. Problém byl popsán i v [1].

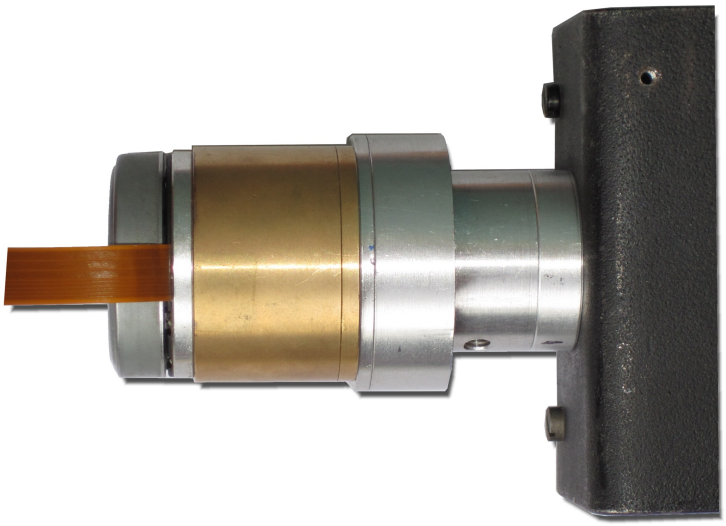
Řídicí systém tak neměl žádnou informaci o aktuálním natočení hnacího kola, protože informace o natočení je brána z řídicí jednotky motoru zodpovědného za natáčení kola. Vzhledem k tomu, že mechanická vazba mezi tímto motorem a vlastním kolem byla velmi volná, informace poskytnutá jednotkou motoru se neshodovala s realitou. Protože robot bude potřebovat k určování polohy a k navigaci odometrické odečítání ujeté vzdálenosti, bylo toto chování naprosto nepřípustné a bylo tedy nutné uchycení předního kola předělat.

Jedním z problémů uchycení předního kola je relativně krátká výstupní hřídel převodovky pohonu. Tato hřídel musí projít přírubou motoru a teprve od této vzdálenosti je možné v unašeči udělat závit se šroubem pro dotažení k hřídeli. Vzhledem k tomu, že v původním návrhu byl použit k dotažení šroub M3, byl potřebný přesah hřídele přes přírubu motoru relativně malý. Jak se ukázalo, toto mechanické řešení nebylo dostatečné, a proto byl v novém návrhu k dotažení unašeče na hřídeli použit tvrzený zápustný šroub se závitem M6. Tím ovšem vyvstal problém s malým přesahem hřídele motoru přes přírubu. Jediné schůdné řešení bylo tedy ve snížení výšky příruby.



Obrázek 2.1: Nákres mechanismu natáčení hnacího kola

Dále byla zadána výroba nového unašeče. Původní návrh, kdy byl závit pro pojistný šroub hluboký přibližně 3 mm, se ukázal jako nevyhovující. Byl tedy navrhnut unašeč s tloušťkou stěny 13,5 mm. Tato masivnější konstrukce si vyžádala bohužel i úpravy na základně robotu, kdy bylo potřeba zvětšit otvor pro uchycení předního kolo z původních 20 mm na 36 mm. S touto úpravou souviselo i přesunutí optických závor určených pro nalezení referenční polohy natočení kola. Narozdíl od původního provedení z nízkouhlíkaté oceli byl nový unašeč vyroben z duralu.



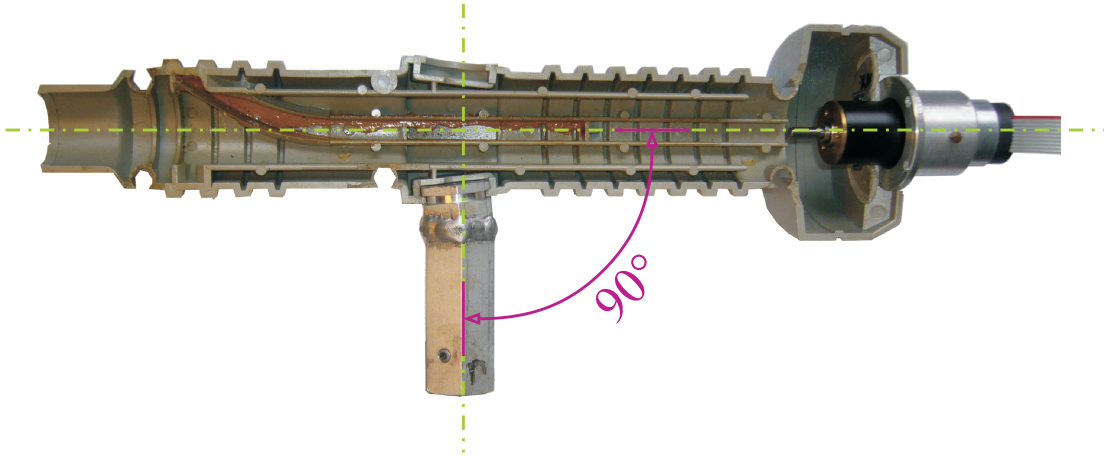
Obrázek 2.2: Fotografie mechanismu natáčení hnacího kola

2.2 Ruka robotu

2.2.1 Uchycení v rameni

Původní konstrukce robotu měla mezi osou kloubu J2 (natáčení v rameni) a osou kloubu J3 (prodloužení ruky) úhel přibližně 97°. Číslování jednotlivých kloubů je popsáno v kap. 4.1.2. Důvod, z jakého se konstruktéři robotu rozhodli pro takového mechanické uspořádání není znám. Nicméně fakt, že tento úhel není 90°, prakticky způsobil analytickou neřešitelnost inverzní kinematické úlohy (více v [3, kap. 3.2]) . Sice by bylo možné pokusit se řešit kinematiku numericky, ale tento přístup byl zavrhnut pro výpočetní náročnost a nejistou konvergenci k řešení.

Bylo tedy rozhodnuto o úpravě uchycení ruky v rameni a to tak, aby úhel mezi osami kloubů J2 a J3 byl přesně 90°. Nejsnazší cestou k dosažení cíle byla shledána úprava hliníkového trnu, který je na jedné straně vsazen do plastového uložení posuvného kloubu J3 a na druhé straně je nasazen na výstupní hřídel převodovky kloubu J2. Trn byl rozříznut, základny částí trnu byly zbroušeny tak, aby kompenzovaly rozdíl mezi původním a požadovaným úhlem uchycení a poté byl celý díl znovu svařen.



Obrázek 2.3: Upravené uchycení ramene robotu

2.2.2 Kabeláž

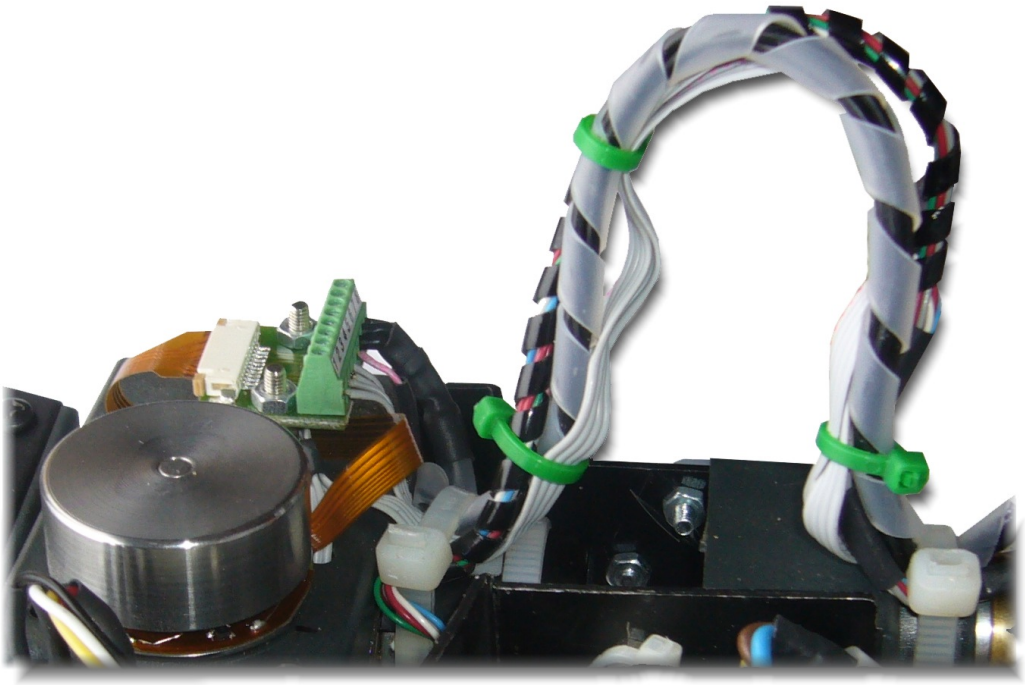
V rámci rekonstrukce mechanické ruky provedené v roce 2008 byly odmontovány všechny původní krokové motory a byly nahrazeny moderními stejnosměrnými motory Maxon. Tento zásah se týkal i motoru ovládajícího natočení chapadla robotu, který je umístěn na pohyblivé převodové skříni. Nové motory mají ovšem vyšší nároky na kabeláž, původní čtyři vodiče ovládající krokový motor byly nahrazeny celkem osmi vodiči potřebnými pro řízení motoru stejnosměrného. Nově dodaná kabeláž byla provedena relativně silnými vodiči, které omezovaly volný pohyb zápěstí při naklápění a zbytečně zvyšovaly mechanické namáhání zápěstí. Původní vedení tvořily 3 kabely:

- 3x 0,75 mm^2 s měděným opletením, sloužící buzení vinutí stejnosměrného motoru
- 5x 0,16 mm^2 plochý, sloužící k propojení Hallových čidel v motoru s řídicí jednotkou
- 4x 0,16 mm^2 plochý, pro připojení referenčního snímače polohy natáčení zápěstí
- 6x 0,16 mm^2 jednotlivé vodiče, ovinuté plastem, napájení vinutí krokového motorku pro ovládání stisku chapadla

Bylo tedy rozhodnuto o výměně úseku kabelu napájejícího vinutí stejnosměrného motoru. Tento kabel měl nepoddajnou izolaci, tloušťka vodičů byla pro daný účel předimenzovaná a celou situaci ještě zhoršovalo měděné opletení kabelu. Jako náhrada byla zvolena výměna za tři samostatné vodiče o průřezu 0,5 mm^2 se silikonovou izolací. Tento typ vodičů se používá například k měřicím přístrojům a vyniká velkou ohebností. Silikonová izolace má tu výhodu, narozdíl od běžné izolace z PVC, že časem netvrdne a zůstává

pořád pružná. Protože se jedná o tři samostatné vodiče, bylo dále nutné tyto tři vodiče společně ovinout plastovou ovinovací bužírkou.

Vzhledem k tomu, že motorem natáčení chapadla tečou nízké proudy se sinusovým průběhem, bylo původní stínění kabelu tomto úseku shledáno nepotřebným a nebylo nahrazeno.



Obrázek 2.4: Kabeláž vedoucí do převodovky zápěstí

Kapitola 3

Senzorický systém

Hlavním účelem této práce je především rozšíření senzorického systému a využití informací získaných senzory. Pro montáž na robotu jsou k dispozici následující senzory:

- Laserový skener Hokuyo URG-04LX
- 4 ks optických senzorů vzdálenosti Sharp GP2D12

Laserový skener bude umístěn na horní otočné plošině robotu a bude sloužit pro sestavování mapy prostředí a následnou orientaci v prostoru. Optické senzory vzdálenosti budou namontovány na základně robotu, co nejníže nad zemí, a jejich účelem bude předcházení kolize s překážkami v prostředí, které nezvládl detekovat laserový skener kvůli jejich malé výšce.

3.1 Skener Hokuyo URG-04LX

Laserový skener Hokuyo URG-04LX slouží pro měření vzdálenosti. Zařízení vysílá laserový paprsek a následně detekuje jeho odraz. Vzdálenost od překážky vypočte na základě uplynulého času mezi vysláním a zpětným příjmem paprsku (metoda *TOF – time-of-flight*). Je vybaven rotačním rozmítáním měřicího paprsku v rovině v rozsahu 240°. Tento typ skeneru se vyznačuje relativně dobrou přesností (max. 1%), ale na druhou stranu i celkem krátkým dosahem paprsku (okolo 4 m). Například firma SICK (*www.sick.de*) vyrábí skenery s dosahem v řádu desítek metrů, ale tato zařízení jsou přibližně 5x - 10x dražší než použitý skener Hokuyo. Další nevýhodou skenerů SICK jsou jejich větší rozměry a vyšší spotřeba. Vzhledem k tomu, že se předpokládá provoz robotu ve vnitřním prostředí, kde se typicky nebudeme potýkat se vzdálenostmi většími než 10 m, je použitý skener

vhodnou volbou. Další informace o senzoru včetně nákresů je možné najít v [6].

Obrázek 3.1: Skener Hokuyo URG-04LX

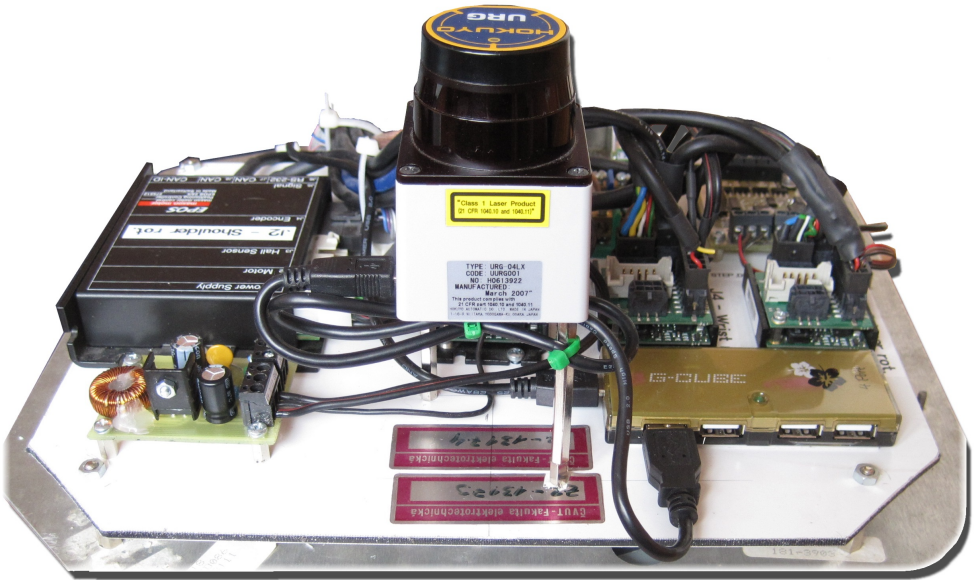
Napájení	5 V / 500 mA
Laser	785 nm, třída 1
Frekvence skenování	10 Hz
Pracovní vzdálenost	20 - 4096 mm
Pracovní úhel	240°
Rozlišení ve vzdálenosti	1 mm
Rozlišení v úhlu	0,36°
Rozhraní	RS-232, USB
Hmotnost	140 g

Tabulka 3.1: Technické parametry skeneru URG-04LX

3.1.1 Připojení

Skener nabízí dvě rozhraní pro propojení s nadřazeným systémem - RS232 a USB 2.0. Řídicí počítač robotu je taktéž vybaven oběma rozhraními. Maximální dosažitelná rychlost přenosu dat při použití RS-232 činí 115,2 kbit/s. Při připojení pomocí sběrnice USB skener pracuje v režimu *Full Speed*, tedy dle specifikace rozhraní s přenosovou rychlostí 12 Mbit/s. V tomto režimu je ovšem reálná rychlost, díky vnitřnímu HW skeneru, omezena na 9 Mbit/s. Nicméně stále je přenos pomocí rozhraní USB téměř o dva řády rychlejší než při použití starší sběrnice RS-232.

Další výhodou sběrnice USB je fakt, že ji lze pomocí rozbočovačů větvit. Vzhledem k tomu, že se počítá s instalací skeneru na horní otočnou plošinu robotu, je nutné kabelovým průstupem v kloubu otáčejícím plošinou vést kabeláž ke skeneru. Při použití sběrnice USB lze na otočné plošině instalovat rozbočovač a do něj, kromě skeneru, připojit ještě další zařízení, v našem případě například webkameru či desku stereovidění. Kvůli výhodnosti tedy bylo rozhodnuto o použití sběrnice USB.



Obrázek 3.2: Celkový pohled na montáž skeneru na hlavě robotu. Nalevo napájecí zdroj skeneru, napravo USB hub

3.1.2 SW rozhraní

Skener komunikuje s nadřazeným počítačem pomocí znakového rozhraní definovaného v [7]. Komunikační protokol je velmi jednoduchý, sestává z několika málo příkazů, v našem případě je použit pouze příkaz „G“, sloužící pro přenos naměřených hodnot ze skeneru do nadřazeného systému. Naměřené hodnoty jsou přenášeny pomocí znakového kódování v sadě ASCII. Parametry příkazu a postup dekódování odpovědi od senzoru jsou popsány v [7, str. 5].

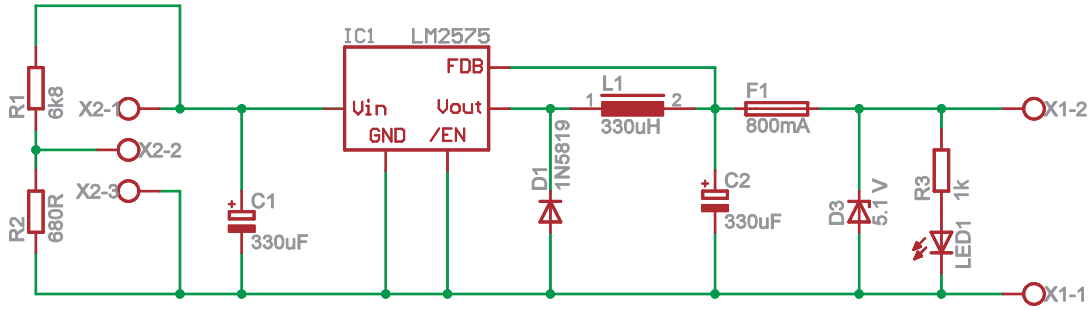
Pokud je senzor připojen pomocí sběrnice RS-232, je přenos informací pomocí kódování ASCII přirozený. V případě připojení přes sběrnici USB se skener hlásí jako zařízení třídy CDC-ACM (*Control Device Class – Abstract Control Model*). USB zařízení třídy CDC-ACM emulují standardní znakový terminál. Tato třída zařízení je používána napří-

klad pro USB modemy, převodníky USB–RS-232 a další zařízení pracující ve znakovém režimu.

V řídicím počítači je použit operační systém Gentoo Linux. Linuxové jádro obsahuje obecný ovladač pro zařízení třídy CDC-ACM. Tento ovladač je do jádra přidán ve formě modulu pojmenovaného `cdc-acm`. Jediná úprava stávající konfigurace OS řídicího počítače tak spočívala v kompilaci daného modulu a jeho přidání do seznamu automaticky zaváděných modulů během startu systému. Po připojení je skener v souborovém systému reprezentován zařízením `/dev/ttyACM0`. K tomuto zařízení taktéž přistupuje řídicí aplikace robotu.

3.1.3 Napájení skeneru

Skener požaduje napájení napětím 5V. Průměrný odběr činí 500 mA, špičkový až 800 mA. Na horní plošině robotu je k dispozici pouze rozvod palubní sítě o jmenovitém napětí 24 V pro napájení pohonů ruky. Bylo by sice možné napájení vyvést ze sběrnice USB, ale je otázkou, zda by poskytla dostatek energie i během špiček v odběru, například při rozbíhání skeneru. Navíc by to prakticky znemožnilo připojení dalších zařízení napájených výhradně ze sběrnice USB, jako například již výše zmíněné webkamery.



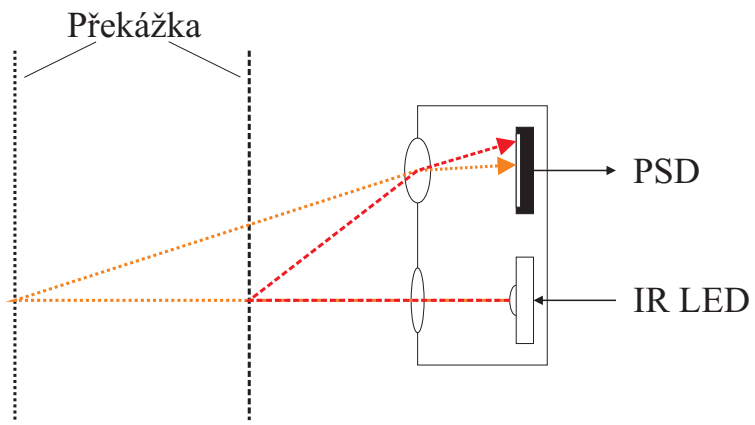
Obrázek 3.3: Schéma zapojení regulátoru napětí

Bylo tedy nutné vyrobit obvod pro snížení palubního napětí na napětí vhodné pro napájení skeneru. Pro tento účel jsem zvolil obvod LM2575HVT-05. Jedná se o integrovaný spínaný stabilizátor s pevně nastaveným výstupním napětím. Obvod je zapojen v doporučeném zapojení podle katalogu výrobce [8]. Zapojení je dále rozšířeno o zenerovu diodu D3, která spolu s pojistkou F1 chrání skener před přepětím, vzniklým například v důsledku proražení stabilizátoru. Byla použita vratná polovodičová pojistka s jmenovitou hodnotu 800 mA.

Odporový dělič tvořený přesnými odpory R1 a R2 slouží ke sledování stavu palubní sítě a je napojen na analogový vstup A_{in} 1 jednotky EPOS, která je součástí pohonu kloubu $J3$. Maximální napětí přivedené na analogový vstup jednotek EPOS činí 5 V, a je tedy nutné sledované palubní napětí snížit pod tuto mez. Řídicí systém takto může v případě poklesu napětí pod bezpečnou úroveň varovat uživatele a případně robot vypnout, aby předešel zničení akumulátorů zajišťujících napájení.

3.2 Senzory Sharp GP2D12

Senzory Sharp GP2D12 jsou optické senzory vzdálenosti pracující na triangulačním principu. Aktivním prvkem je LED dioda, která vysílá modulovaný optický signál. Při odrazu od překážky signál projde čočkou přijímací části snímače a dopadne na světlocitlivý snímač PSD. Princip funkce je naznačen na obr. 3.4. Elektronika senzoru poté vyhodnotí signál z PSD a nastaví výstupní napětí senzoru podle detekované vzdálenosti.



Obrázek 3.4: Princip funkce senzoru GP2D12

Napájení	5 V / max. 50 mA
Frekvence snímání	25 Hz
Pracovní vzdálenost	10 - 80 cm
Výstupní napětí	0,4 - 2,6 V

Tabulka 3.2: Technické parametry senzoru GP2D12

3.2.1 Připojení

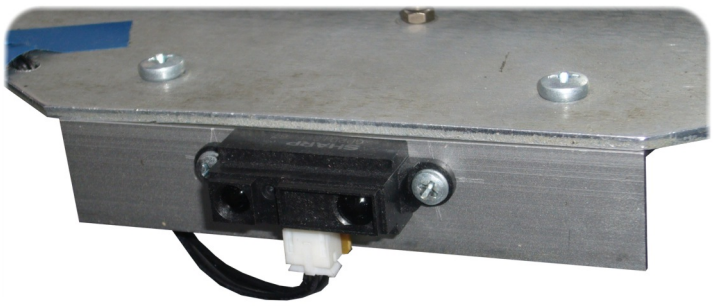
Elektrické připojení senzoru je velmi prosté - senzor je napájen napětím 5V a poskytuje jeden analogový výstup odpovídající detekované vzdálenosti. Výstupní napětí senzoru se pohybuje v rozmezí 0,4 - 2,6 V. Senzor je tedy vhodný pro přímé připojení k univerzálním vstupům jednotek EPOS. Pomocí jednotek EPOS může řídicí SW velmi jednoduše načítat hodnoty analogových výstupů. Napájení senzorů je poskytováno také jednotkami EPOS, které disponují nepoužívaným stabilizovaným výstupem 5 V / 100 mA pro napájení optických enkodérů. Připojení jednotlivých senzorů k jednotkám je provedeno podle tab.3.3.

Senzor	Jednotka	Vstup
Přední strana	EPOS 24/5 - pojezd	A_{in} 2
Levá strana	EPOS 24/5 - pojezd	A_{in} 1
Pravá strana	EPOS 24/1 - zatáčení	A_{in} 2
Zadní strana	EPOS 24/1 - zatáčení	A_{in} 1

Tabulka 3.3: Připojení senzorů Sharp k jednotkám EPOS

Senzor	V_{cc}	V_{out}	Gnd
Přední strana	modrá	fialová	šedá
Levá strana	modrá	fialová	šedá
Pravá strana	červená	hnědá	černá
Zadní strana	oranžová	žlutá	zelená

Tabulka 3.4: Připojení senzorů Sharp - barvy vodičů



Obrázek 3.5: Namontovaný senzor GP2D12 - zadní strana

3.3 Stereovidění

Jedním z možných rozšíření senzorického systému robotu Hero je instalace modulu stereovidění. Stereovidění nám poskytuje obrazovou informaci o bezprostředním okolí robotu včetně informace o hloubce obrazu (vzdálenosti). Toho by se dalo využít především ve spojení s rukou robotu, kdy by bylo možné například vyhledávat a uchopovat různé předměty. Další možné využití spočívá ve sběru dodatečných informací pro orientaci v prostředí a jejich fúze s daty získanými ze skeneru Hokuyo. Údaje ze stereokamery by posloužily například při identifikaci objektů v menší vzdálenosti od robotu a objektů, které leží mimo snímací rovinu skeneru.

Na katedře je k dispozici vývojový kit stereovidění CM-BF561-Lite od firmy Bluetechnix. Sestává ze základní desky osazené výkonným VideoDSP procesorem ADSPBF561 firmy Analog Devices a z přídavné desky se dvěma kamerami. Základní modul disponuje několika rozhraními, z nichž je pro nás nejzajímavější rozhraní USB. Modul, podobně jako skener Hokuyo, komunikuje přes rozhraní USB v režimu emulace znakového terminálu. Díky vyvedení sběrnice USB na horní otočnou plošinu robotu, kde bude pravděpodobně deska stereovidění umístěna, nebude v budoucnu tvořit integrace modulu do výbavy robotu větší problémy. Další informace o uvedené vývojové desce, procesoru a problematice stereovidění lze nalézt v [5].

Kapitola 4

Kinematika

4.1 Přímá kinematická úloha

Přímá kinematická úloha převádí souřadnice robotického manipulátoru z prostoru kloubových souřadnic (úhly natočení jednotlivých kloubů) do euklidovského prostoru (poloha a orientace). Tento přepočet bude robotu sloužit například k předejití kolizní situace v případě, že by nějaká část mechanické ruky zasahovala do známé překážky v okolním prostředí. Přímá kinematická úloha se také označuje zkratkou *DKT*.

4.1.1 Denavitova-Hartenbergova notace

Pro řešení přímé kinematické úlohy sériových manipulátorů se typicky používá takzvaná Denavitova-Hartenbergova metoda, která dává obecný návod k řešení přímé kinematiky. Výsledkem tohoto postupu je umístění lokálních souřadných systémů pevně spojených s jednotlivými kinematickými prvky robotu a návod na vzájemnou transformaci souřadnic mezi těmito jednotlivými souřadnými systémy. Prvním krokem je očíslování ramen (*links*) a kloubů (*joints*) v pořadí od základny tak, že každé rameno i má kloub blíže základně označen číslem i a končí kloubem *i+1*. Nyní si umístíme do každého kloubu lokální souřadný systém B_i s počátkem O_i a osami x_i, y_i, z_i, a to podle následujících pravidel [11, str. 199]:

- *Osa z_i je totožná s osou kloubu i+1*
  Nejdříve začínáme určením všech os z. U rotačních kloubů je tato osa jednoznačně určená (kladný směr osy je volitelný), u posuvných kloubů můžeme zvolit jakoukoliv

osu rovnoběžnou s se směrem posunu. Z praktických důvodů se ovšem většinou volí přímo osa daného posuvného kloubu.

- *Osa x_i je nejkratší spojnicí (společnou normálou) os z_{i-1} a z_i. Osa x_i směřuje od z_{i-1} do z_i.*
  Pokud jsou osy z_{i-1} a z_i rovnoběžné, dostáváme nekonečné množství možných os x_i. V tomto případě zvolíme osu x_i tak, že je totožná s osou x_{i-1}. Pokud se osy z_{i-1} a z_i protínají, neexistuje mezi nimi nejkratší spojnice (je degenerovaná do jediného bodu). Osu x_i tedy zvolíme podle vztahu $x_i = z_{i-1} \times z_i$

- *Osa y_i je určena vztahem $y_i = z_i \times x_i$*
  Osa y_i je zvolena tak, aby s osami x_i a z_i tvořila pravotočivý souřadný systém. Pro stanovení Denavitových-Hartenbergových parametrů není přímo nutná, takže se v nákresech pro větší přehlednost často vynechává.

Pokud jsme správně rozmístili lokální souřadné systémy, můžeme nyní každé rameno robotu popsat čtyřmi D-H parametry. Tyto parametry jsou:

- *Délka ramene (link length) a_i*
  Délkou ramene rozumíme vzdálenost mezi osami z_{i-1} a z_i podél osy x_i.

- *Zkrut ramene (link twist) α_i*
  Úhel α_i je úhel, o který se musí otočit osa z_{i-1} okolo osy x_i tak, aby byla rovnoběžná s osou z_i

- *Délka kloubu (joint distance, link offset) d_i*
  Délka kloubu (přesazení ramen) je definovaná jako vzdálenost mezi osami x_{i-1} a x_i podél osy z_{i-1}. V případě posuvných kloubů bývá tento parametr kloubovou souřadnicí, kterou určujeme délkou vysunutí posuvného kloubu.

- *Úhel kloubu (joint angle) θ_i*
  Úhel kloubu je úhel, o který se musí otočit osa x_{i-1} okolo osy z_{i-1} tak, aby byla rovnoběžná s osou x_i. V případě rotačních kloubů bývá tento parametr kloubovou souřadnicí, kterou určujeme natočením kloubu.

Po stanovení všech čtyř parametrů můžeme sestavit transformační matici převádějící souřadnice zadané v souřadném systému B_i do souřadného systému B_{i-1}, tedy blíže základně robotu. Postupným prováděním těchto transformací mezi jednotlivými souřadnými systémy dostaneme polohu bodu zadaného například v souřadném systému chapadla

(například bod $[0, 0, 0]$ - střed chapadla) vyjádřenou v souřadném systému rámu robotu. Odtud ji budeme moci dále převést do světového souřadného systému. Vzhledem k tomu, že se jedná o robota mobilního, je pro transformaci souřadnic ze souřadného systému rámu robotu do světového souřadného systému nutná také znalost polohy robota ve světovém souřadném systému.

Pro zjišťování možné kolize ruky s předměty v okolním prostředí si můžeme obdobně v každém lokálním souřadném systému stanovit velikosti daného ramene robotu a dalších namontovaných prvků pevně spojených s tímto souřadným systémem a vytvořit bezpečnostní obálku, pro jednoduchost například tvaru válce či kvádru. Po transformaci této obálky do světového souřadného systému můžeme zjišťovat, jestli nedochází ke kolizi se známou překážkou. Transformace ze souřadného systému B_i do B_{i-1} je realizována jako čtyři postupné transformace, dvě rotace a dvě translace.

$$
{}^{i-1}T_i = D_{z_{i-1},d_i} R_{z_{i-1},\theta_i} D_{x_{i-1},a_i} R_{x_{i-1},\alpha_i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Transformace souřadnic ze souřadného systému se provede vynásobením vektoru souřadnic transformační maticí zleva. Při těchto výpočtech používáme homogenní souřadnice, tedy souřadnice jejichž dimenze je o jeden stupeň vyšší než dimenze prostoru, ve kterém máme souřadnice zadány. Transformační vztah mezi euklidovskými a homogenními souřadnicemi je následující:

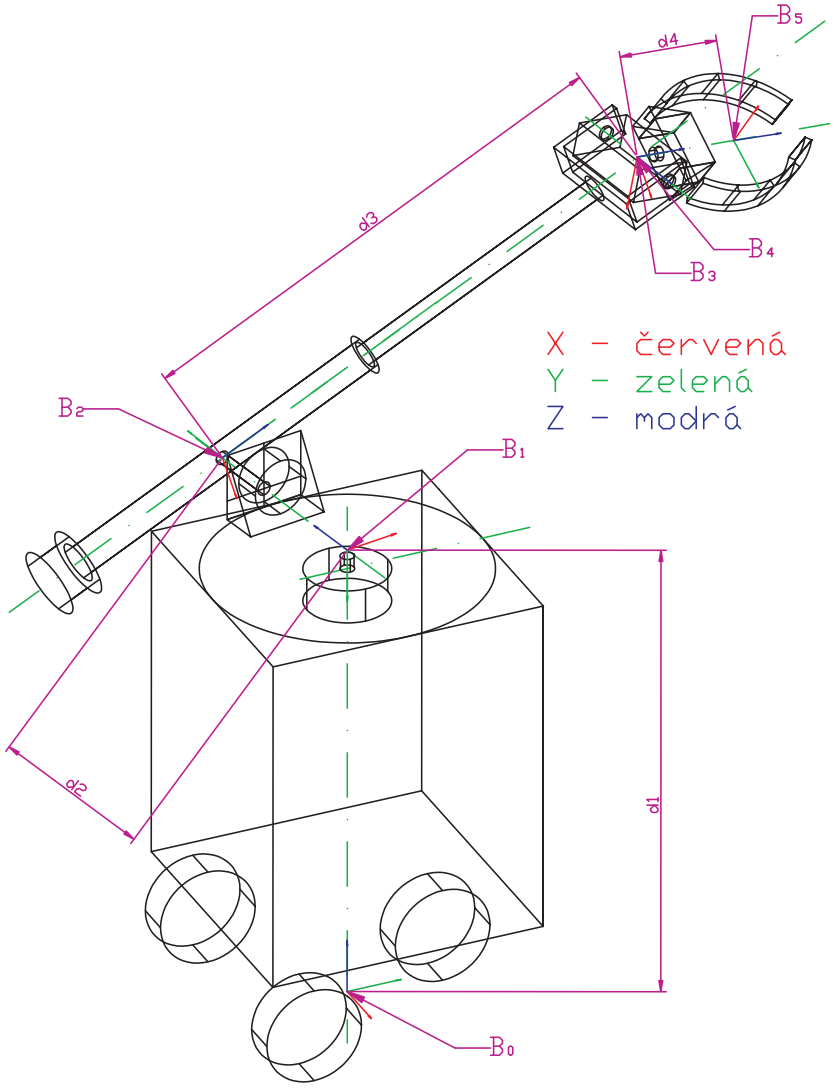
$$
\begin{bmatrix} \frac{x}{\omega} \\ \frac{y}{\omega} \\ \frac{z}{\omega} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \omega \end{bmatrix}
$$

Měřítko zobrazení ω se typicky volí $\omega = 1$. Transformaci souřadnic mezi sousedními souřadnými systémy tedy můžeme zapsat následovně:

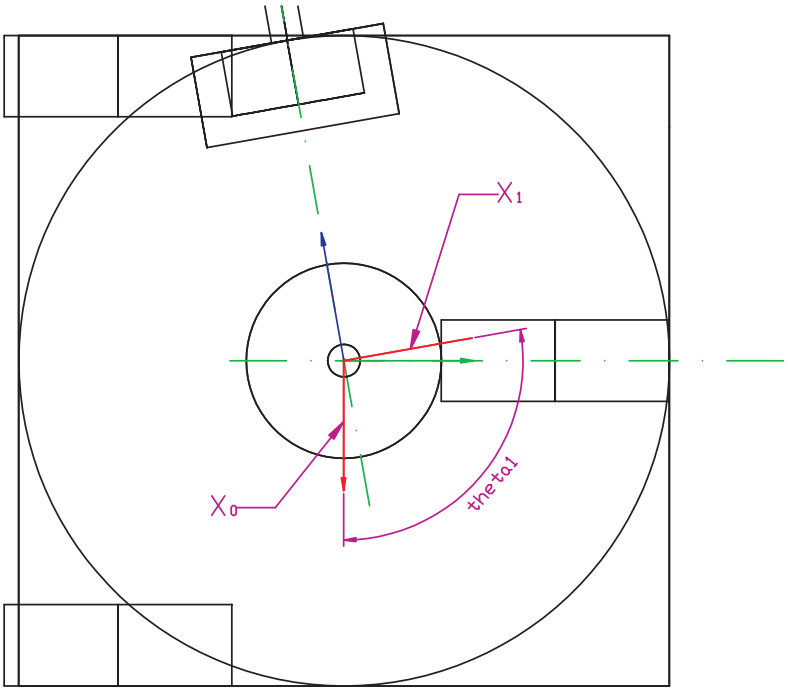
$$
\begin{bmatrix} x_{i-1} \\ y_{i-1} \\ z_{i-1} \\ 1 \end{bmatrix} = {}^{i-1}T_i \begin{bmatrix} x_i \\ {}_iy \\ z_i \\ 1 \end{bmatrix}
$$

4.1.2 Řešení přímé kinematické úlohy

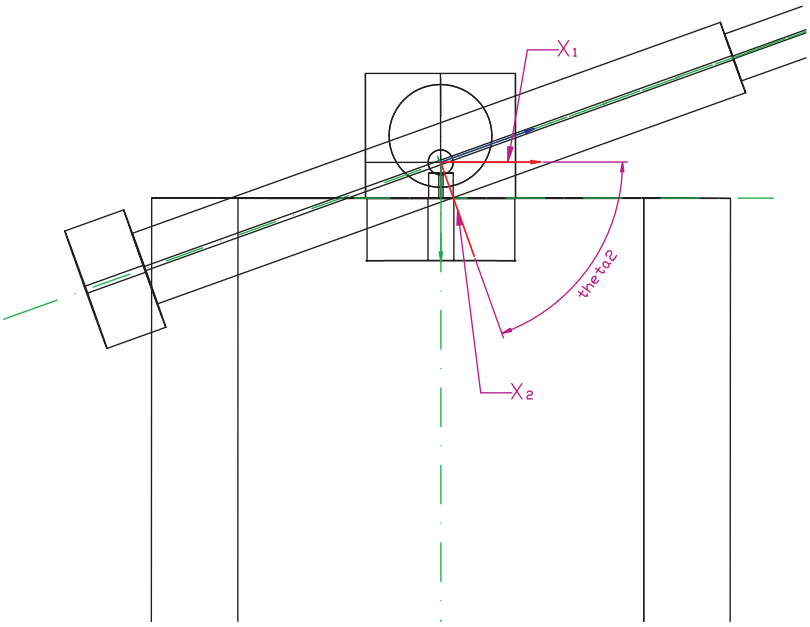
Prvním krokem je rozmístění lokálních souřadných systémů pevně spojených s kinematickými články robotu. Na obrázku 4.1 vidíme umístění souřadných systémů B_i a jejich očíslování. B_0 je souřadný systém pevně spojený se základnou robotu. Zároveň jsou na nákresu vyznačeny délky jednotlivých ramen. Na dalších nákresech jsou detaily rotačních kloubů. Pro jednotlivé klouby jsem zvolil zkrácené označení J_i (z angl. *joint*). Pro zpřehlednění nákresů jsou jednotlivé osy souřadných systémů odlišeny barvami.



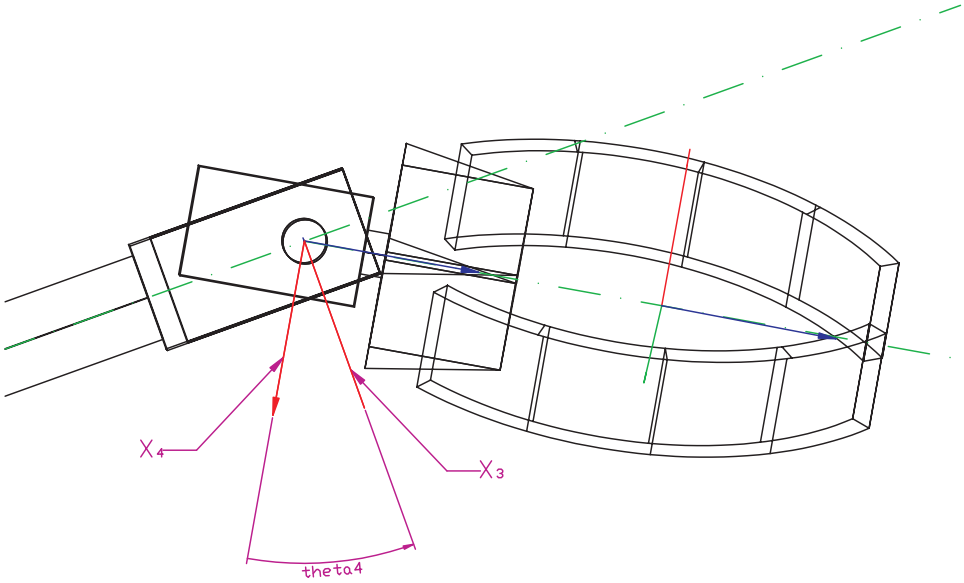
Obrázek 4.1: Rozmístění jednotlivých souřadných systémů, celkový pohled



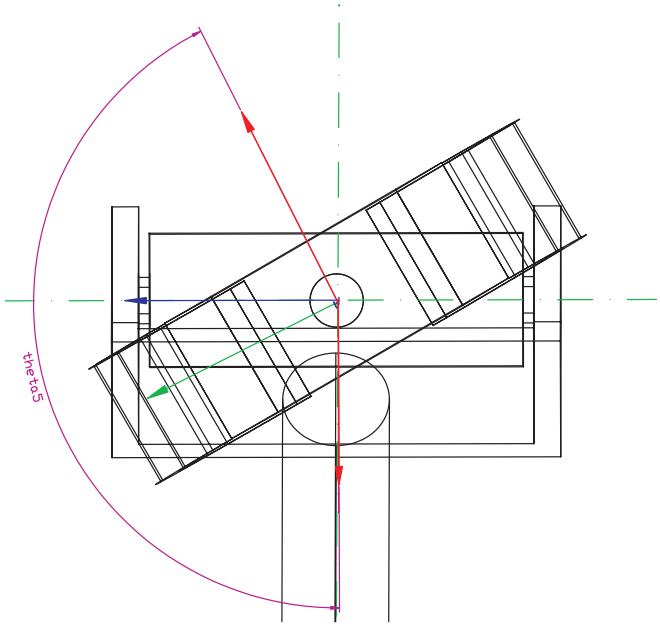
Obrázek 4.2: Rotační kloub J_1, pohled shora



Obrázek 4.3: Rotační kloub J_2, pohled v ose rotace



Obrázek 4.4: Rotační kloub J_4, pohled v ose rotace



Obrázek 4.5: Rotační kloub J_5, pohled v ose rotace

i	a_i	α_i	d_i	θ_i
1	0	$-90°$	d_1	θ_1
2	0	$90°$	d_2	θ_2
3	0	$90°$	d_3	0
4	0	$-90°$	0	θ_4
5	0	$0°$	d_4	θ_5

Tabulka 4.1: D-H parametry

Rozměr	Velikost
d_1	38 cm
d_2	26 cm
d_4	9 cm

Tabulka 4.2: Rozměry robotu

Podle výše popsaného postupu určíme všechny Denavitovy-Hartenbergovy parametry (tab. 4.1). Skutečné rozměry jsou uvedeny v tab. 4.2. θ_i a d_3 jsou kloubové souřadnice, tedy vstupní parametry výpočtu. Tímto jsme získali všechny potřebné údaje pro řešení přímé kinematické úlohy.

4.2 Inverzní kinematická úloha

Inverzní kinematická úloha je postup, který převádí požadovanou koncovou polohu a orientaci chapadla, zadanou v kartézském souřadném systému, do prostoru kloubových souřadnic. Označuje se zkratkou *IKT*. Na rozdíl od přímé kinematické úlohy neexistuje obecný návod pro její řešení. Pro většinu běžných robotů se šesti stupni volnosti a se zápěstím se třemi protínajícími se osami existuje ovšem doporučený postup, který většinou vede k výsledku. Je jím dekompozice kinematiky robotu na rameno (první tři stupně volnosti) a na zápěstí (poslední tři stupně volnosti). Vzhledem k tomu, že zápěstí se třemi protínajícími se a navzájem kolmými osami dovoluje nastavit jakoukoliv orientaci, řešíme pro první tři klouby pouze dosažení správné polohy a orientaci koncového nástroje jsme schopni plně řídit pomocí zápěstí.

Bohužel, robot Hero je vybaven rukou pouze s pěti stupni volnosti, která neumožňuje ani v omezeném pracovním prostoru dosáhnout libovolné orientace zápěstí. Vzhledem k tomu, že robot byl původně vyvinut především jako výukový nástroj bez jakýchkoliv

ambicí na autonomní řízení, nemuseli návrháři robotu příliš přemýšlet nad kinematickou strukturou ruky. Robot byl ve své době většinou řízen direktivně nahraným programem napsaným v jednoduchém programovacím jazyku. V tomto programu byly silně omezené možnosti jednoduchého rozhodování a podpora matematiky prakticky neexistovala. Pohyb ruky byl tedy řízen tak, že si jej uživatel nacvičil v manuálním režimu, zjistil do jaké polohy je třeba nastavit jednotlivé motory a robot pak pouze nastavil klouby do předdefinované polohy.

Vzhledem k tomu, že chapadlo robotu nemůže dosáhnout libovolné orientace, je standardní popis požadované polohy (např. pomocí Eulerových úhlů) nepoužitelný. Bylo tedy třeba nalézt jiný vhodný popis požadované orientace chapadla. Vzhledem ke konstrukci ruky byla zvolena parametrizace orientace chapadla pomocí dvou úhlů:

- *Úhel* γ
  Tento parametr vyjadřuje požadovaný úhel mezi osou chapadla a vodorovnou rovinou (rovina xy světového souřadného systému).

- *Úhel* ψ
  Tento parametr vyjadřuje požadovaný úhel otočení chapadla kolem osy kloubu J_5. Úhel ψ přímo odpovídá kloubové souřadnici θ_5, je zaveden pouze kvůli formálnímu oddělení prostorů kartézských a kloubových souřadnic.

4.2.1 Řešení inverzní kinematické úlohy

Na obr. 4.6 a 4.7 vidíme nákresy robotu spolu s vyznačenými veličinami použitými při výpočtu inverzní kinematiky. Písmenem E je označen koncový bod chapadla (z angl. *effector*) a písmenem W bod ve kterém se protínají osy zápěstí (z angl. *wrist*). Vstupním parametrem pro výpočet inverzní kinematické úlohy jsou souřadnice bodu E, zadané v souřadném systému rámu robotu, a úhly γ a ψ. Postup výpočtu kloubových souřadnic je následující:

$$c = \sqrt{E_x^2 + E_y^2} \tag{4.1}$$

$$r = \sqrt{c^2 - d_2^2} \tag{4.2}$$

$$r_4 = d_4 \cdot \cos\gamma \tag{4.3}$$

$$z_4 = d_4 \cdot \sin\gamma \tag{4.4}$$

$$r_3 = r - r_4 \tag{4.5}$$

$$z_3 = E_z + z_4 \tag{4.6}$$

$$d_3 = \sqrt{c_3^2 + (d_1 - z_3)^2} \tag{4.7}$$

$$\theta_{1A} = \arctan(E_y, E_x) \tag{4.8}$$

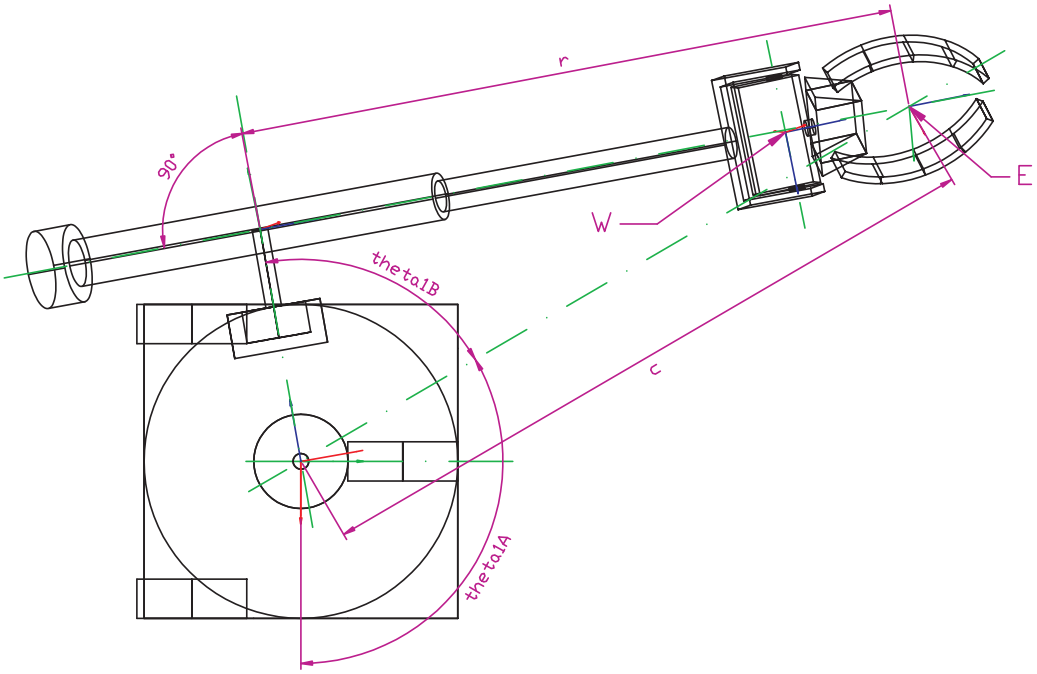
$$\theta_{1B} = \arccos\frac{d_2}{c} \tag{4.9}$$

$$\theta_1 = \theta_{1A} + \theta_{1B} \tag{4.10}$$

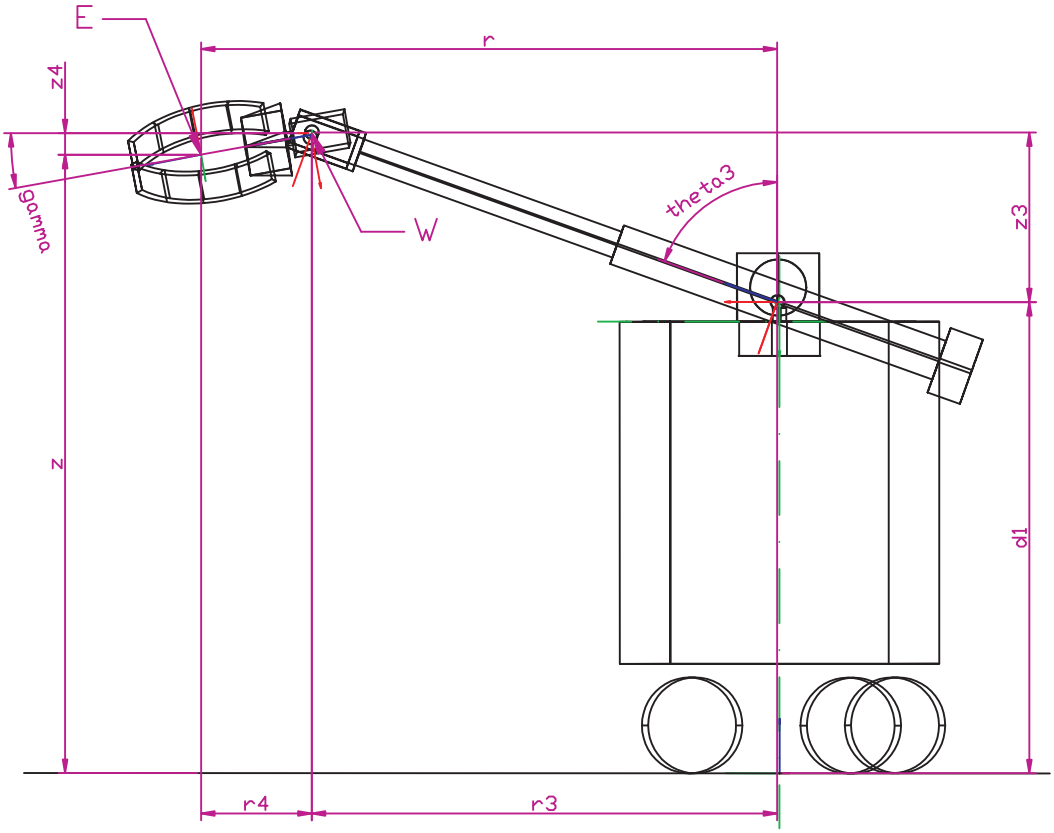
$$\theta_2 = \frac{\pi}{2} - \arctan((z_3 - d_1), r_3) \tag{4.11}$$

$$\theta_4 = \frac{\pi}{2} + \gamma - \theta_2 \tag{4.12}$$

$$\theta_5 = \psi \tag{4.13}$$



Obrázek 4.6: Pohled shora



Obrázek 4.7: Pohled ze strany (rovina pohledu kolmá na osu kloubu J_2)

Dostáváme tedy vektor kloubových souřadnic $J = [\theta_1, \theta_2, d_3, \theta_4, \theta_5]^T$. Správnost řešení byla ověřena pomocí simulace v systému Matlab, kdy jsem pro konečný počet bodů řešil rovnici

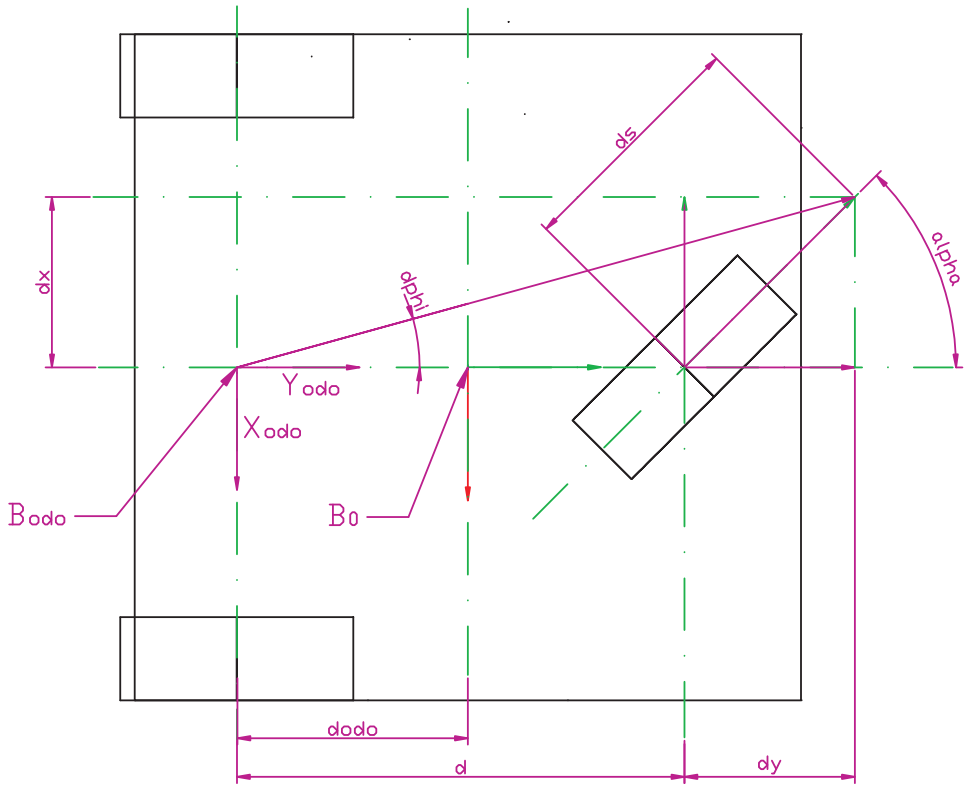
$$\bar{e} = dkt(x_i) - dkt(ikt(dkt(x_i)))$$

a sledoval velikost chyby ($\bar{e}$). Testovací body byly zadávány v prostoru kloubových souřadnic. Průměrná chyba výpočtu byla přibližně 10^{-14} m, maximální chyba přibližně 10^{-12} m. Je tedy jasné, že nepřesnost způsobená případnou nestabilitou výpočtu je daleko menší než nepřesnosti způsobené např. mechanickými nedokonalostmi robotu.

Robot může požadovaného koncového bodu dosáhnout i ve druhé možné konfiguraci manipulátoru. Tuto bychom dostali, pokud bychom rovnici 4.10 nahradili rovnicí $\theta_1 = \theta_{1A} - \theta_{1B}$ a pokračovali dále ve výpočtu. Nastavení ruky do této konfigurace by vyžadovalo překlopení ramene v kloubu J_2. Toto chování je, vzhledem k vůli v dané převodovce (popsáno v [3, str. 27]), nežádoucí, a tak robot pracuje pouze s jedinou kinematickou konfigurací.

4.3 Odometrie

Odometrické měření slouží robotu k určení polohy ve světovém souřadném systému na základě znalosti ujeté vzdálenosti a úhlu natočení poháněného kola. Robot Hero má tři kola, z nichž dvě jsou pevně spojená s rámem robotu a třetí poháněné, kolo zajišťuje zatáčení a pojezd. Odometrický model byl již vytvořen a popsán v [1]. Ačkoliv byl tento model správný, z výpočetního hlediska měl tu nevýhodu, že ve výpočtu figuroval člen $\tan\alpha + \frac{\pi}{2}$. Tento člen měl za následek nedefinované řešení v situaci, kdy robot jede rovně vpřed ($\alpha = 0$). Vytvořil jsem tedy nový model, jehož výpočet sice není tak přímočarý, ale který má definiční obor bez nespojitostí.



Obrázek 4.8: Odometrický model - pohled shora

Základním vztažným bodem při výpočtu odometrie je bod B_{odo}, ve kterém se protínají osy všech tří kol. Úhel α určuje natočení poháněného kola. Při ujetí vzdálenosti ds poháněným kolem se natočení robotu ve světovém souřadném systému změní o hodnotu

$$d\phi = \arctan \frac{ds \cdot \sin\alpha}{d} \tag{4.14}$$

a poloha počátku souřadného systému F_{odo} se změní takto:

$$dx = ds \cdot \cos\alpha \cdot \sin\phi \tag{4.15}$$

$$dy = ds \cdot \cos\alpha \cdot \cos\phi \tag{4.16}$$

a tímto dostaneme vektor $d\bar{P} = [dx, dy, d\phi]^T$. Poloha robotu je ve světovém souřadném systému vyjádřena vektorem $\bar{P} = [x, y, \phi]^T$, který aktualizujeme vektorem $d\bar{P}$:

$$\bar{P} = \bar{P} + d\bar{P} \tag{4.17}$$

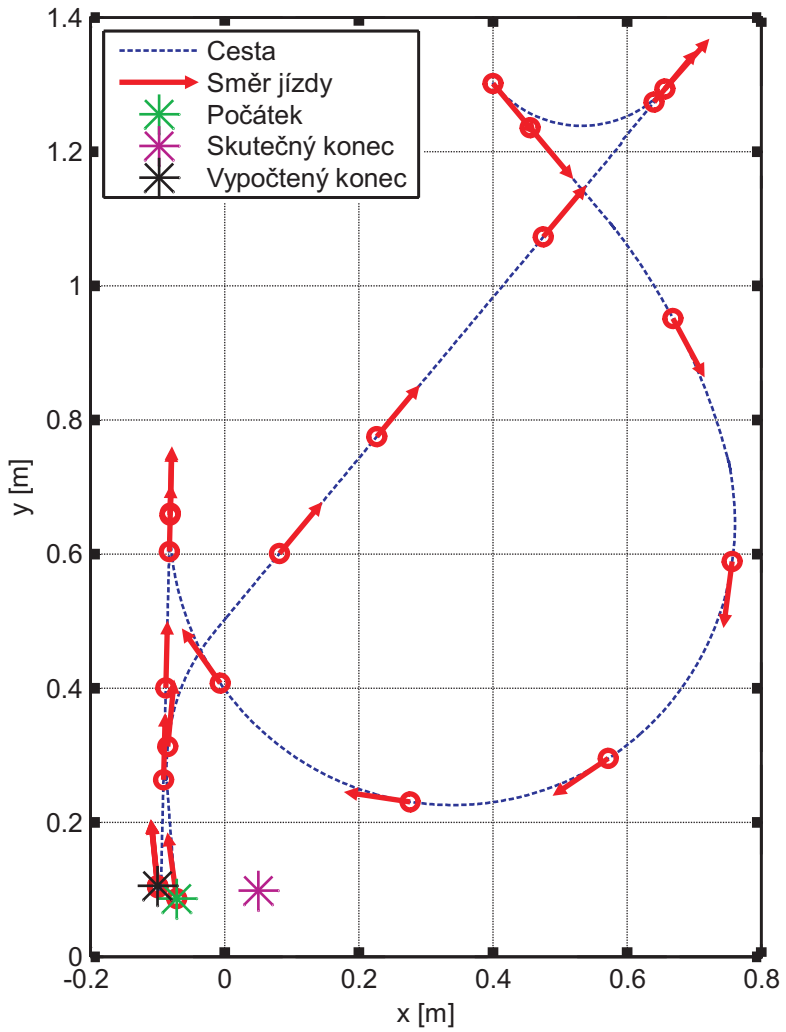
Tento postup je správný pouze pokud $ds \to 0$, poté lze provést integraci popsaných veličin a dostaneme správný výsledek. V praxi se ovšem ujetá vzdálenost vzorkuje, takže dostáváme nespojitou funkci. V případě robotu Hero jsem zvolil vzorkovací frekvenci $f_{odo} = 20$ Hz. Pokusy ukázaly, že je takto zvolená frekvence dostatečně vysoká a vzniklé chyby lze přičíst spíše nedokonalosti kol. Pro ideálně přesné odometrické měření je nutné, aby kola měla nulovou šířku, což v případě reálného robotu samozřejmě není možné. Je tedy vhodné použít kola co nejužší, ovšem na robotu jsou použita relativně široká kola a navíc nemají ideální válcovitý tvar, takže můžeme očekávat vyšší chyby způsobené mechanickými faktory.

Na obr. 4.9 je záznam reálné trasy ujeté robotem. Záznam byl pořízen v režimu ručního řízení robotu. Vidíme, že robot vyjel z výchozího bodu, zatočil mírně doprava, pak se couváním otočil o -90°a poté se pozvolnou zatáčkou dostal do osy své počáteční orientace a zacouval zpět na výchozí místo. Po změření skutečné a předpokládané polohy robotu jsme dostali následující chyby:

Veličina	Skut. poloha	Předp. poloha	Rozdíl
x [m]	0,05	-0,09	-0,15
y [m]	0,10	0,10	0,01
ϕ [rad]	0	-6,18	0,10

Tabulka 4.3: Chyby odometrie

Absolutní vzdálenost (Eukleidovská norma) předpokládané a skutečné polohy činila $0,15$ m. Rozdíl v úhlu natočení je uveden po normalizaci do základní periody.



Obrázek 4.9: Měření chyby odometrie

Kapitola 5

Mapa prostředí

Pro navigaci v prostoru je potřeba, aby si robot byl schopen sestavovat mapu okolního prostředí a poté, aby byl schopen reagovat na překážky vynesené v mapě. V robotice se běžně používají dva základní typy map:

- *Rastrové (metrické) mapy* - tyto mapy jsou reprezentovány mřížkou s přiřazenými hodnotami (např. volná/obsazená buňka) rozkládající se přes celý mapovaný prostor. Tato reprezentace je z lidského pohledu přirozená a lze v ní zcela jasně určovat vzdálenosti a polohy objektů. Nevýhodou tohoto přístupu je relativně objemná datová reprezentace mapy, což negativně ovlivňuje rychlost operací prováděných nad mapou a dále vysoká citlivost na šum při sestavování mapy.

- *Topologické mapy* - tento druh map je reprezentován pouze význačnými body určenými z výstupů senzoru (tzv. *landmarks*) a vzájemnými vztahy mezi nimi. Mapy jsou tak reprezentovány grafem. Výhodou tohoto přístupu je velmi kompaktní datová reprezentace mapy a vysoká rychlost operací s mapou. Nevýhodou je velmi komplikovaná transformace do prostoru kartézských souřadnic. Tento přístup je také méně citlivý na šum senzorů.

Pro smysluplnou funkci robotu s mechanickým manipulátorem (např. pro přemisťování předmětů) je naprosto nezbytné mít jednoznačnou informaci o poloze ve světovém souřadném systému, a proto bylo rozhodnuto o použití rasterové mapy.

5.1 Sestavování mapy a lokalizace

Sestavování mapy a současná lokalizace (angl. *simultaneous localization and mapping - SLAM*) je jedním z největších problémů mobilní robotiky. Při této činnosti vyvstávají

dva navzájem závislé problémy:

- pro přesnou aktualizaci mapy právě naměřenými daty potřebujeme velmi přesně znát aktuální polohu
- pro přesné určení aktuální polohy potřebujeme mít k dispozici kvalitní mapu, abychom k ní mohli okamžitou polohu vztáhnout.

Za ideálních podmínek, kdy by použité senzory měly nulový šum a měření odometrie by probíhalo bez jakékoliv chyby, lze druhý problém přehlédnout (díky přesné odometrii máme dokonalou informaci o naší poloze) a soustředit se na sestavování mapy. V reálném případě (kdy je odometrie zatížena velkou integrační chybou) lze ovšem nahlédnout, že se jedná o složitý problém, který se neobejde bez pokročilých filtračních algoritmů. Pro řešení tohoto problému (u rasterových map) se většinou používá jeden z těchto dvou přístupů:

- *Pravděpodobnostní přístup* - v tomto případě je mřížka mapy vyplněna hodnotami reprezentujícími pravděpodobnost existence překážky v daném bodě. V případě použití algoritmu pracujícího s Bayesovskou (podmíněnou) pravděpodobností si systém dokonce udržuje mnoho map (typ. 50 - 100) a tuto sadu map aktualizuje novými daty. Při takzvaném uzavření smyčky (kdy robot dojede na již navštívené místo) dojde k sestavení výsledné mapy. Tato výsledná mapa je sestavena s vysokou přesností a tolerancí vůči šumu. Již z popisu algoritmu je zřejmé, že nároky na paměť a výpočetní výkon budou značné. Typicky je třeba paměť v řádu GB a v případě procesoru platí *„čím rychlejší tím lepší“*, přičemž pouze nejvýkonnější současné vícejádrové procesory jsou schopny zpracovávat tento algoritmus v reálném čase. Pravděpodobnostní metody pracují většinou s informacemi z odometrie a s laserovými dálkoměrnými skenery, případně s obrazy z kamer.
- *Použití rozšířeného Kalmanova filtru (EKF)* - při použití EKF (angl. *extended Kalman filter*) je normální lineární model Kalmanova filtru nahrazen modelem nelineárním. Stav modelovaného systému je tvořen polohou robotu, měřením ze senzorů a částí mapy prostředí. Stavový vektor může tedy obsahovat stovky prvků a odtud pramení výpočetní náročnost tohoto přístupu, především na operace s plovoucí desetinnou čárkou. Výhodou tohoto přístupu je snadná fúze dat v podstatě z jakýchkoliv senzorů. Použitým senzorům pouze přiřadíme jakousi *„míru důvěryhodnosti“*.

Nevýhodou je neexistence procesu obdobnému uzavření smyčky v případě pravděpodobnostních algoritmů. Dalším problémem je předpoklad šumu s normálním rozložením. Bohužel v praxi se ukazuje, že při řešení problému SLAM se reálný šum od normálního velmi liší, a tak algoritmy využívající EKF mají tendenci po čase driftovat a jejich (teoreticky vysoká) přesnost je takto snížena.

5.2 CoreSLAM

Při sestavování mapy máme v případě robotu Hero k dispozici informace ze dvou senzorů, a to výstup laserového skeneru a údaje o poloze získané odometrickým měřením. Problém lokalizace a mapování, za použití těchto senzorů, byl řešen v [12] a výsledkem byla knihovna funkcí nazvaná CoreSLAM, implementovaná v jazyku C. Tato knihovna byla veřejně publikována a je volně ke stažení např. na *www.OpenSLAM.org*. Na tomto webu lze také nalézt jiné algoritmy a materiály zabývající se problematikou SLAM.

Knihovna CoreSLAM je pro využití na robotu Hero výhodná z několika důvodů:

- senzory použité autory se shodují se senzory, které jsou na robotu k dispozici, a to včetně typu skeneru
- knihovna byla implementována s ohledem na budoucí využití v rámci embedded zařízení, je tedy optimalizována na co nejnižší výpočetní náročnost (např. kritická smyčka využívá pouze celočíselnou matematiku)
- knihovna je implementována v jazyku C, integrace do stávajícího SW je tedy bezproblémová.

Knihovna CoreSLAM využívá algoritmus s pravděpodobnostním přístupem k řešení problému SLAM. Využívá však pouze jedinou mapu, nepracuje tedy s více hypotézami a s podmíněnou pravděpodobností (Bayesovský přístup). Tím odpadá vysoká výpočetní a paměťová náročnost, ovšem za cenu horších dosažitelných výsledků.

Nejdůležitější funkce z knihovny CoreSLAM používané řídicím SW robotu jsou tyto:

- `int ts_distance_scan_to_map (ts_scan_t *scan, ts_map_t *map, ts_position_t *pos);`

  Tato funkce pouze přepočítává data získaná v souřadném systému skeneru do světového souřadného systému (souřadnice v mapě).

- `void ts_map_update(ts_scan_t * scan, ts_map_t * map, ts_position_t * position, int quality, int hole_width);`

  Funkce `ts_map_update` je stěžejní funkcí algoritmu, na základě dat poskytnutých skenerem upravuje hodnoty na příslušných souřadnicích v mapě. Čím nižší je hodnota bodu na mapě, tím je větší pradvěpodobnost, že se na daném místě nachází překážka.

  Parametrem `quality` upravujeme míru, s jakou se mění hodnoty mapy během jedné aktualizace. V podstatě se jedná o hodnotu určující důvěryhodnost dat z jediného skenu. Nižší hodnota tohoto parametru pomůže například při tvorbě mapy v dynamickém prostředí, kdy je k dobrému výsledku potřeba zprůměrovat více jednotlivých měření.

  Parametrem `hole_width` upravujeme šířku oblasti, ve která dojde k úpravě hodnot v mapě. Tímto parametrem je postihnuta přesnost skeneru. Hodnotu tohoto parametru je také třeba volit s ohledem na funkci `ts_monte_carlo_search`.

- `ts_position_t ts_monte_carlo_search(ts_randomizer_t * randomizer, ts_scan_t * scan, ts_map_t * map, ts_position_t * start_pos, double sigma_xy, double sigma_theta, int stop, int * bestdist);`

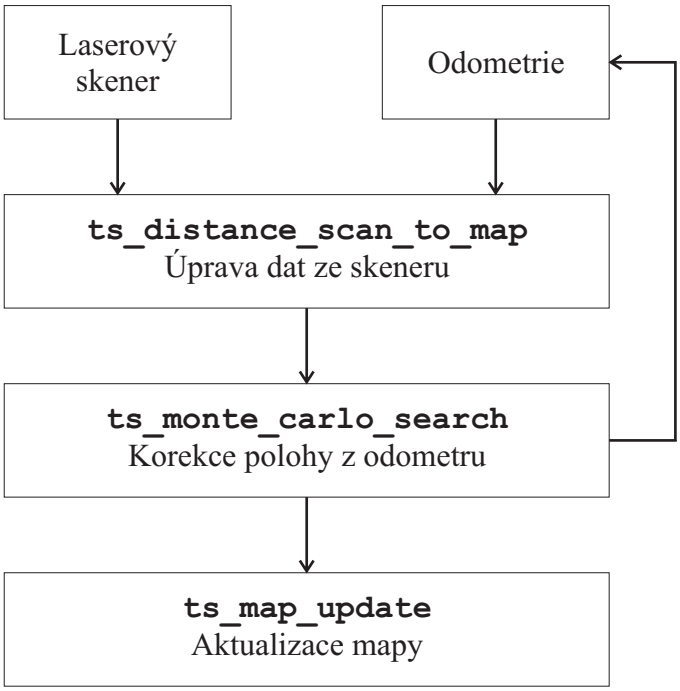
  Funkce `ts_monte_carlo_search` slouží k nalezení orientace robotu v mapě. Polohu vyhledává na základě aktuálního skenu okolí za použití optimalizační metody Monte Carlo.

  Parametrem `start_pos` je funkci předána poloha, ze které má začít vyhledávání (v našem případě odometrická data). Čím je tato poloha bližší skutečné poloze v mapě, tím je výkon funkce rychlejší.

  Dalšími důležitými parametry jsou `sigma_xy` a `sigma_theta`. Tyto určují směrodatnou odchylku v chybě polohy, resp. úhlu natočení. Jejich hodnoty mají zásadní vliv na kvalitu odhadu a bylo třeba je určit experimentálně.

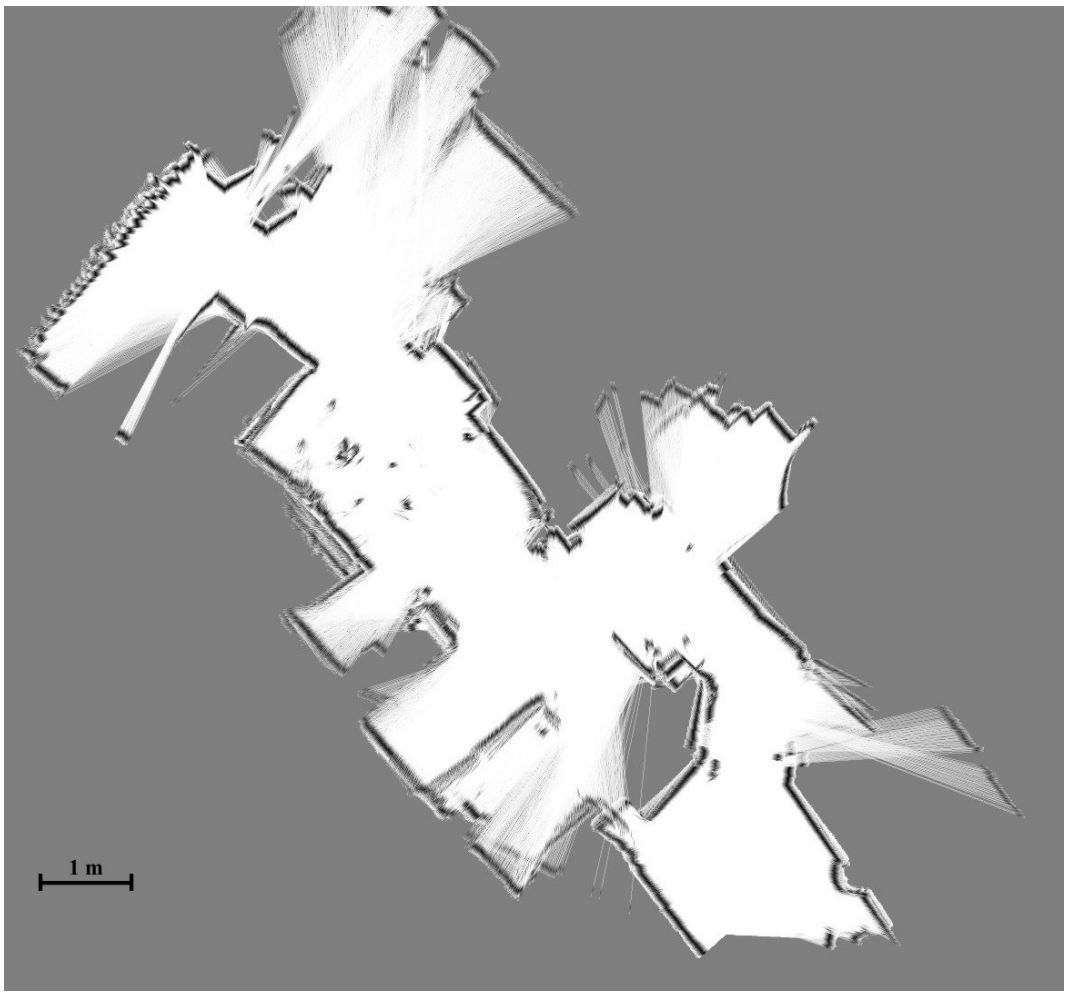
Princip funkce algoritmu a tok dat je zachycen na obr. 5.1. Výchozím bodem jsou data ze skeneru a z odometrického měření. Na jejich základě je vygenerován obraz skenu okolí ve světovém souřadném systému a ten je, spolu s odometrií a současnou mapou prostředí, poskytnut jako vstupní parametr funkci `ts_monte_carlo_search`. Výstupem této funkce je zpřesněná poloha, která je, spolu s nasnímaným skenem, použita pro následnou aktualizaci mapy. Zároveň je aktualizován i údaj o poloze robotu, čímž je upřesněna

odometrie pro další měření. Dostáváme tedy stav, kdy jsou odometrická data v každé iteraci algoritmu zpřesněna odhadem polohy z mapy a integrační chyba se tedy uplatňuje pouze mezi dvěma běhy mapovacího algoritmu, takže horní hranice této integrační chyby je omezena.



Obrázek 5.1: Princip funkce použitého algoritmu SLAM

Pro experimenty s robotem byla použita mapa o rozměrech 2048×2048 bodů s rozlišením 1 px $\sim$ 1 cm. Dostáváme tedy čtverec o délce hrany přibližně 20 m, který je robot schopen zmapovat. Velikost mapy se dá změnit ve zdrojovém kódu robotu nastavením parametru `TS_MAP_SIZE`. S ohledem na obrazovou kompresi a následný přenos mapy do nadřazeného PC je vhodné velikost mapy omezit na předpokládanou velikost prostoru, ve kterém se bude robot pohybovat. Na obr. 5.2 je ukázka mapy vytvořené robotem (bytový prostor). Do mapy je uměle dokresleno měřítko pro srovnání.



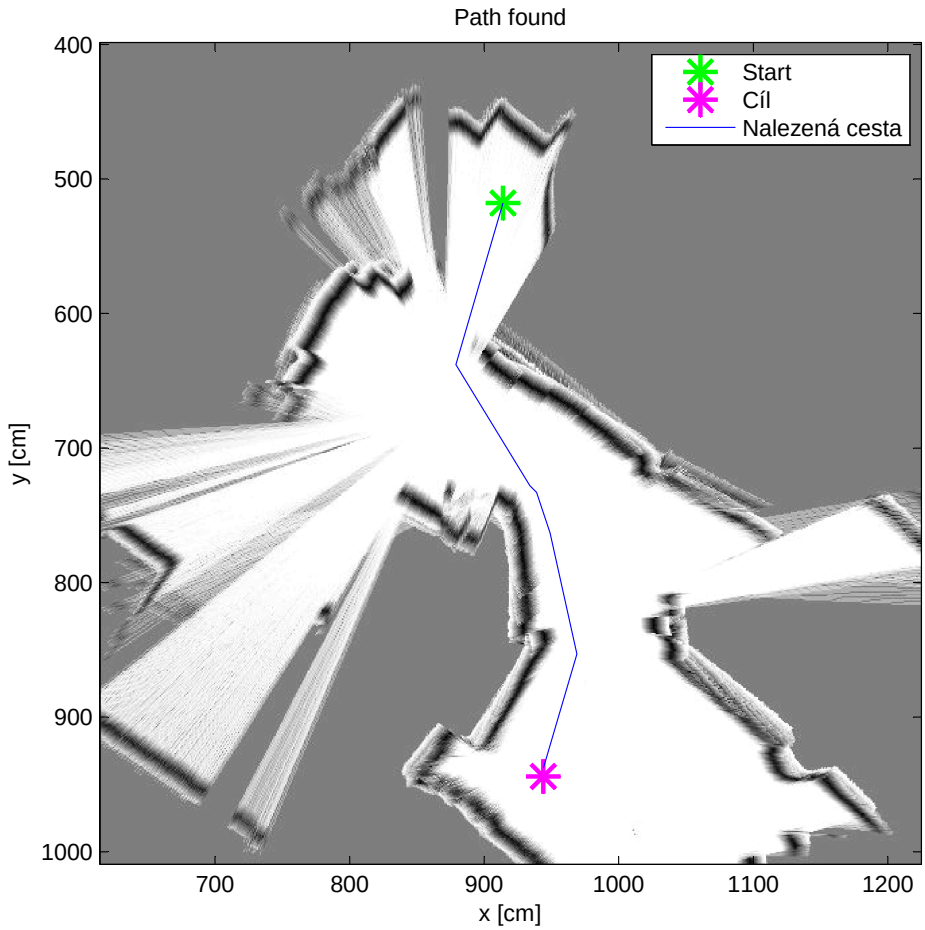
Obrázek 5.2: Příklad mapy vytvořené robotem (výřez)

5.3 Hledání cesty v mapě

Pro hledání cesty ve vytvořené mapě byl použit algoritmus Theta* popsaný v [14]. Jedná se o obdobu známého prohledávacího algoritmu A* optimalizovanou pro hledání cest ve dvourozměrných mapách. Běh algoritmu a dosažené výsledky na modelových případech jsou popsány v odkazovaném materiálu. Pro účely implementace algoritmu v řídicím SW robotu Hero byl algoritmus Theta* nejprve pokusně implementován v prostředí Matlab pro ověření součinnosti s mapami produkovanými robotem. Algoritmus byl navíc rozšířen tak, aby bral v úvahu nenulové rozměry robotu, a vyhnul se tak překážkám s dostatečným odstupem. Výsledek jednoho zkušebního běhu algoritmu je zobrazen na obr. 5.3.

V [14] je dokázáno, že algoritmus Theta* poskytuje pouze suboptimální řešení. Experimenty autorů ovšem ukazují, že rozdíl řešení poskytnutého algoritmem a Theta* a optimálního řešení problému je řádově v jednotkách procent. Pro účely navigace robotu je

takovýto rozdíl zanedbatelný. Autoři dále popisují rozšíření algoritmu nazvané AP-Theta* (*angle-propagation Theta**), které zásadně snižuje výpočetní náročnost algoritmu. Toho je dosaženo omezením počtu kontrol obsazených a volných oblastí v mapě. Toto rozšíření by bylo vhodné implementovat v dalších fázích projektu.



Obrázek 5.3: Cesta v mapě nalezená pomocí Theta*

Kapitola 6

Úpravy SW vybavení

Původní programová výbava robotu byla tvořena dvěma částmi:

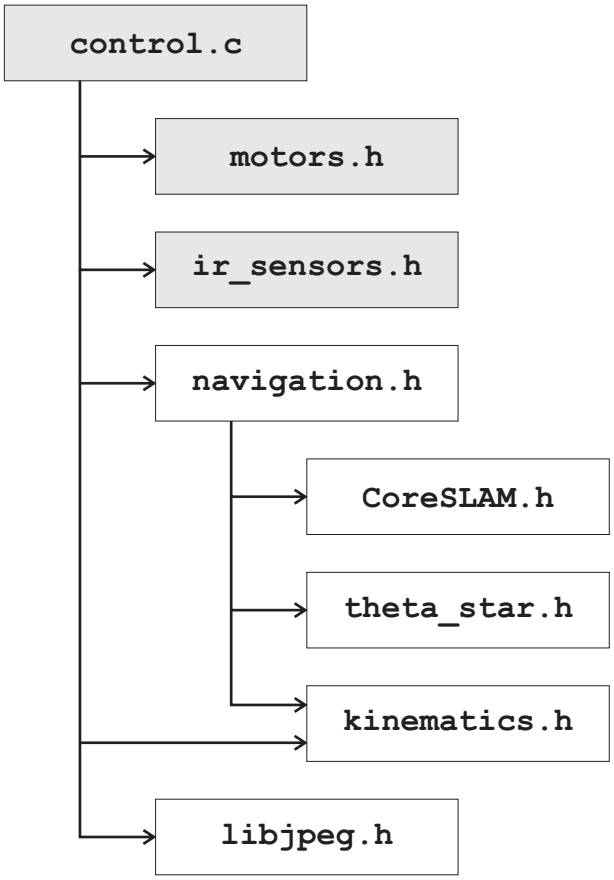
- *Řídicí aplikace (Control)* - běží v palubním počítači typu PC/104. Zprostředkovává komunikaci s řídicími jednotkami pohonů a poskytuje síťové rozhraní pro zadávání povelů z nadřazené ovládací aplikace. V palubním počítači je nainstalován operační systém Gentoo Linux a počítač je s jednotkami pohonů propojen sběrnicí CAN. Řídicí aplikace je implmentována v jazyce C.

- *Ovládací aplikace (HeroGUI)* - běží na nadřazeném (uživatelském) PC a komunikuje s aplikací *Control* běžící v robotu. Uživatel může zadávat požadovanou rychlost a směr natočení hnacího kola, nastavení kloubů manipulátoru a také zjišťovat hodnoty stavových veličin robotu. Aplikace *HeroGUI* je implementována v jazyce JAVA, je tedy platformně nezávislá a je možné ji spustit na všech operačních systémech běžně provozovaných na dnešních počítačích (Windows, Linux, MacOS X a další).

Řídicí SW je výsledkem bakalářské práce O. Šantina [2]. V této práci lze také nalézt další informace o struktuře a funkci obou programů a také důležité informace týkající se konfigurace palubního OS.

Pro zapracování nově požadované funkcionality do řídicího systému bylo potřeba upravit obě popsané aplikace. Vzhledem k tomu, že robot má být ve výsledku autonomním zařízením, je potřeba, aby veškeré procesy pokročilého řízení běžely přímo v řídicím počítači robotu. Většina SW úprav se tedy týkala řídicí aplikace *Control*. Úpravy operátorského rozhraní *HeroGUI* byly poté prováděny v návaznosti na rozšířené schopnosti řídicího SW tak, aby mohl uživatel tyto nové funkce využívat.

6.1 Úpravy řídicí aplikace



Obrázek 6.1: Struktura řídicí aplikace

Struktura řídicí aplikace je naznačena na obr. 6.1. Šedě vyznačené soubory jsou součástí původní SW výbavy. Ostatní byly vytvořeny či přidány v rámci této práce. Z původního projektu jsou kvůli přehlednosti uvedeny pouze tři moduly, které jsou nejdůležitější pro běh nově přidávaných algoritmů pokročilého řízení robotu. Podrobný popis původního SW lze nalézt v [2] a [4]. Základní funkce jednotlivý programových modulů vynesených v diagramu lze charakterizovat takto:

- `control.c` - základní modul starající se o spuštění programu, inicializaci robotu, inicializaci a obsluhu front zpráv (LAN a CAN), komunikaci s operátorským PC a konečně správu běhu jednotlivých vláken (threadů) programu.

- `motors.h` - soubor funkcí pro komunikaci s řídicími jednotkami Maxon EPOS s využitím aplikačního standardu CanOPEN. Patří sem například funkce pro parametrizaci jednotek, volbu pracovního režimu jednotky, nastavování požadované

rychlosti či polohy daného servomechanismu a také funkce pro obsluhu univerzálních vstupů a výstupů jednotek.

- `ir_sensors.h` - modul pro zpracování naměřených výstupů z optických senzorů vzdálenosti Sharp GP2D12. Nejdůležitější funkcí modulu je linearizace naměřených hodnot a jejich převod z napětí na vzdálenost.

- `navigation.h` - zastřešující soubor funkcí zajišťujících navigaci robotu v prostředí. Tento modul poskytuje ucelené rozhraní využívané hlavním modulem programu `control.c`. Mezi hlavní funkce patří výpočet odometrie, tvorba mapy prostředí (pomocí knihovny `CoreSLAM.h`), výpočet cesty v mapě za použití algoritmu Theta* realizovaného souborem `theta_star.h` a následný výkon plánu cesty (průjezd robotu po nalezené trase). V tomto souboru jsou také definovány základní datové struktury používané při navigaci (definice polohy robotu, mapy a další).

- `kinematics.h` - modul pro výpočet přímé a inverzní kinematiky manipulátoru.

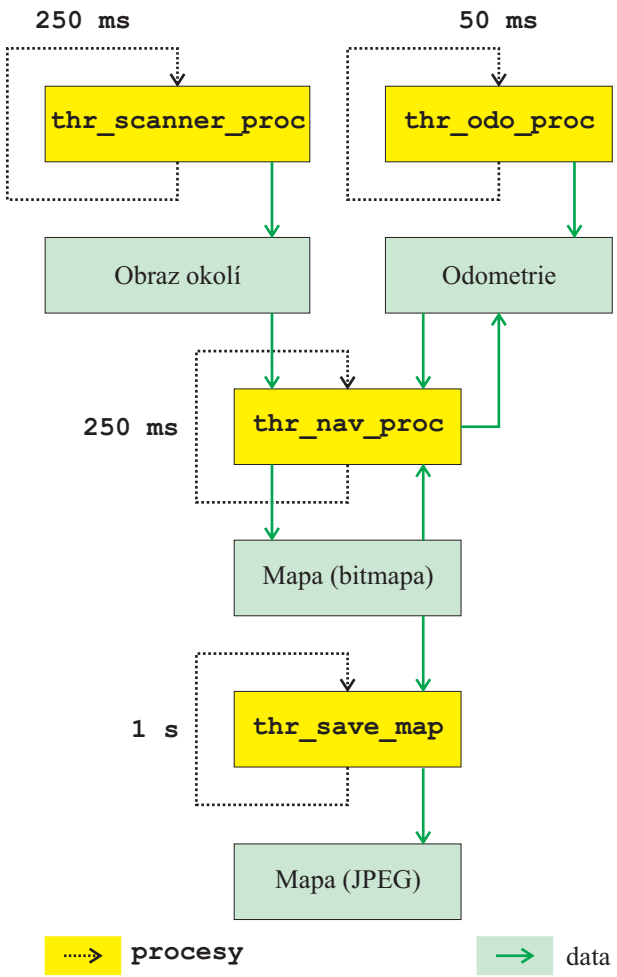
- `libjpeg.h` - standardní knihovna pro (de)kompresi obrázků ve formátu JPEG. V programu je využívána pro kompresi mapy prostředí před jejím následným přenosem do uživatelského PC.

6.1.1 Vlákna procesu

Řídicí aplikace je od počátku koncipována jako aplikace s jedním běžícím procesem s více vlákny. Jednotlivá vlákna procesu dávají možnost zpracovávat více problémů paralelně. V případě použitého hardware je toto zpracování kvůli jednojádrovému procesoru pouze virtuální, protože ve skutečnosti dochází ke sdílení procesoru a vlákna jsou přepínána na základě přednastavených priorit. Na víceprocesorových systémech může ovšem docházet ke skutečně paralelnímu výkonu instrukcí dvou vláken.

V původní aplikaci běžela za normálního provozního stavu dvě vlákna realizovaná cyklickým výkonem funkcí `thr_com_server` a `thr_can_command`. Tato vlákna obstarávala komunikaci s operátorským PC pomocí protokolu TCP, resp. obsluhu sběrnice CAN. Nově byla do programu přidána následující vlákna:

- `thr_scanner_proc` - toto vlákno realizuje komunikaci se skenerem Hokuyo a zajišťuje dekódování dat. Data ukládá do pole celočíselných hodnot `scannerdata`. Spouští se s periodou 250 ms.



Obrázek 6.2: Vlákna běžící v programu a tok dat

- `thr_odo_proc` - tento proces zajišťuje odečítání odometrických dat z jednotek EPOS a následnou aktualizaci polohy robotu. Je spouštěn s periodou 50 ms.
- `thr_nav_proc` - toto vlákno zajišťuje aktualizaci mapy daty získanými ze skeneru a odometrie. Jedná se o hlavní vlákno zajišťující sestavování mapy prostředí. Je spouštěno s periodou 250 ms. Ovšem díky výpočetní náročnosti trvá zpracování většinou déle, a tak docházi k aktualizaci mapy v řidších intervalech. Z hlediska algoritmu není periodicita aktualizace mapy podstatná.
- `thr_ir_catch` - slouží k periodickému odečtu hodnot ze senzorů vzdálenosti Sharp a jejich následnému zpracování a uložení do příslušných proměnných.
- `thr_save_map_proc` - tento proces zajišťuje kompresi mapy a její uložení do paměti, aby byla ihned k dispozici, pokud si jí nadřazená aplikace vyžádá.

- `thr_plan_exec` - tento proces se spouští pouze v případě, kdy je třeba vykonat plán sestavený procedurou pro vyhledávání cesty v mapě, tedy pokud je třeba robotem pohybovat a sledovat naplánovanou trasu.

6.1.2 Přehled funkcí navigačního modulu

V následujícím výčtu jsou uvedeny hlavní funkce poskytované modulem `navigation.h`. Tyto funkce jsou využívány vyššími vrstvami řídicího softwaru.

- `void nav_init (navdata * nd);`

  Tato funkce slouží pro inicializaci datové struktury typu `navdata`, která uchovává informace potřebné pro navigaci (mapa, poloha robotu v prostoru, ...).

- `void nav_update_position (pose2d * position, double distance, double direction, double delta_t);`

  Tato funkce slouží pro pravidelnou aktualizaci polohy na základě vzorkování odometrických dat (ujeté dráhy a úhlu natočení poháněného kola).

- `void nav_map2buffer (ts_map_t * map, char * outbuf, int size_sqrt);`

  Funkce `nav_map2buffer` slouží pro vytvoření kopie mapy v paměti počítače. Tuto kopii je nutné vytvořit kvůli tomu, že po dobu komprese mapy do formátu JPEG (která trvá nezanedbatelnou dobu a probíhá s nízkou prioritou - nejedná se o kritický proces) jsou zdrojová data blokována a nelze s nimi pracovat, a tím pádem by po dobu komprese mapy nebylo možné mapu prostředí aktualizovat. Proto je mapa nejprve pouze zkopírována (což je velice rychlé) a komprimována je teprve tato kopie.

- `int nav_save_map_jpeg (char * in_buffer, char * out_buffer, int size_sqrt);`

  Funkce pro vlastní kompresi dat do formátu JPEG. Využívá funkcí poskytnutých knihovnou libJPEG.

- `void nav_update_map (navdata * nd, unsigned int * scannerdata);`

  Tato funkce je hlavní funkcí pro sestavování a aktualizaci mapy prostředí. Prakticky vykonává postup popsaný na obr. 5.1, zpracovává tedy údaje ze skeneru a na jejich základě aktualizuje mapu a zpřesňuje informaci o poloze robotu.

- `int nav_create_plan (navdata * nd, nav_plan * plan, pose2d * goal_pos);`

  Tato funkce vyhledá cestu ze současné polohy robotu tak, aby se robot dostal na požadované cílové souřadnice a vyhnul se známým překážkám zaneseným v mapě. Funkce využívá modulu `theta_star.h`.

- `int nav_execute_plan (navdata * nd, nav_plan * plan,`
  `void (* speed_callback)(double speed),`
  `void (* direction_callback)(double direction));`

  Tato funkce vykonává plán vytvořený funkcí `nav_create_plan`, řídí tedy pohyb robotu a ten sleduje naplánovanou trasu. Funkci je potřeba předat jako dva parametry funkce poskytované nadřazeným modulem (`control.c`) používané pro nastavení požadované rychlosti a natočení kola robotu.

6.1.3 Komprese mapy

Nekomprimovaná mapa, o velikosti jaká byla použita při experimentech s robotem, má velikost 4 MB. Po převodu do kódování Base64, bezpečného pro přenos přes znakové rozhraní do nadřazeného PC, vzroste velikost na 8 MB. Tato velikost by znemožnila dostatečně rychlý přenos mapy v případě pomalejšího propojení s operátorskou stanicí, například při použití přístupu přes Internet nebo při provozu na zarušené WiFi síti.

Bylo tedy rozhodnuto o nutnosti komprimace dat před vlastním přenosem mapy. Díky velké rozšířenosti a dobré dostupnosti aplikačních příkladů byla pro kompresi vybrána knihovna libJPEG [15]. Po kompresi mapy do formátu JPEG se výsledná velikost pohybuje v rozmezí 100 - 200 kB v závislosti na nastavené úrovni komprese a na komplexnosti mapy. Tato kompaktní velikost mapy je vykoupena výpočetně náročným převodem do formátu JPEG. Proto by bylo vhodné, v rámci dalších úprav SW, vyzkoušet například přenos mapy ve formátu PNG, který by měl být při kompresi obrazu výpočetně méně náročný a zhodnotit případný přínos, zejména s ohledem na poměr výpočetní náročnost/výsledná velikost mapy.

PNG je bezeztrátová komprese založená na knihovně zlib (základ známé komprese ZIP). Vzhledem k tomu, že vytvořená mapa má relativně malou entropii, v porovnání například s plnobarevnou fotografií, nemusela by se výsledná velikost mapy komprimované bezeztrátovým algoritmem PNG příliš lišit od mapy komprimované pomocí ztrátového JPEG.

6.2 Ovládací aplikace

Ovládací uživatelská aplikace *HeroGUI* slouží k ručnímu řízení robotu a k přehlednému zobrazování provozních údajů robotu. Ve výchozím stavu aplikace obsahovala pouze dva hlavní panely - jeden pro ovládání pojezdu a ruční řízení a druhý panel, informativní, zobrazoval komunikaci mezi robotem a ovládací aplikací. Dále bylo možné nastavovat parametry robotu. S rozšířením funkcí palubního řídicího SW vyvstala potřeba rozšíření ovládací aplikace tak, aby byla schopna nové funkce robotu využít. Programová struktura ovládací aplikace je popsána v [2].

V návaznosti na rozšíření funkcí robotu je potřeba zobrazovat v ovládací aplikaci následující údaje:

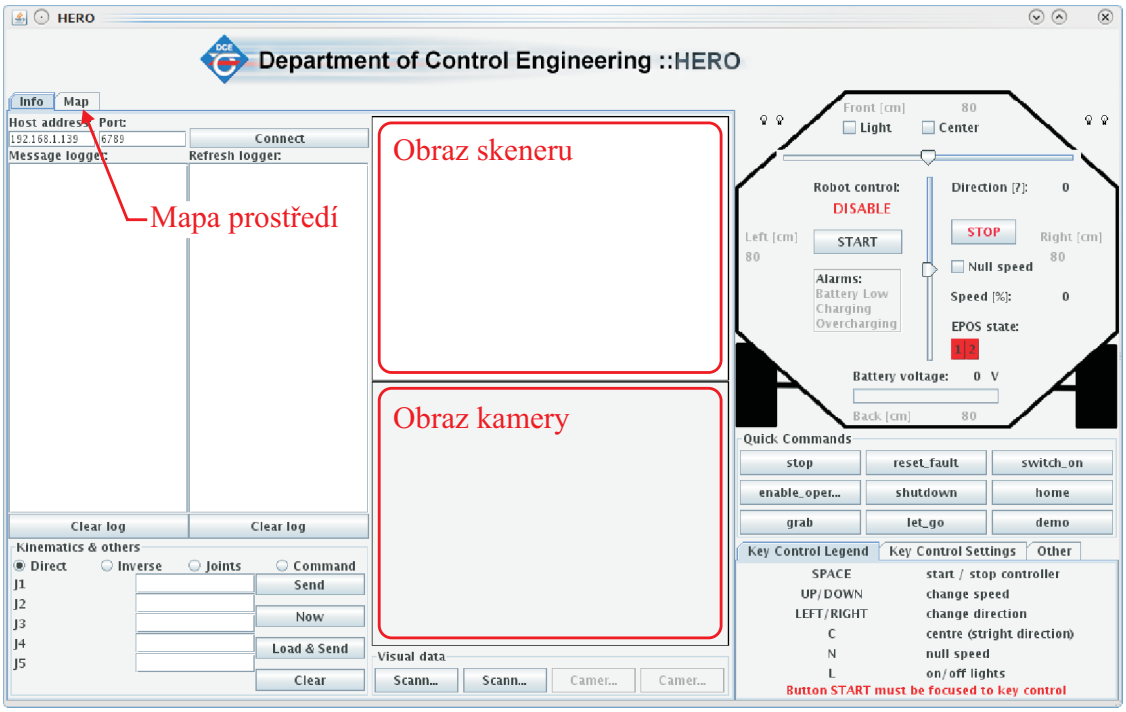
- výstup skeneru Hokuyo (aktuální obraz okolí)
- snímaný obraz z kamery robotu (do budoucna je s ním nutné počítat, ať již to bude prostá webkamera nebo stereometrické vidění popisované v kap. 3.3)
- mapu prostředí vytvářenou robotem
- ovládací prvky například pro zadávání požadované cílové polohy robotu

Formulář aplikace *HeroGUI* je již ovšem zaplněn ovládacími prvky a přidání dalších prvků by ho učinilo nepřehledným. Bylo tedy potřeba identifikovat části uživatelského rozhraní, u kterých se nepředpokládá současné využití, a následně tyto části separovat tak, aby si mohl uživatel vybrat, které ovládací prvky bude mít v daný čas k dispozici na obrazovce a které budou skryté, a současně aby toto řešení neztěžovalo ovládání robotu.

Nakonec bylo rozhodnuto o rozdělení hlavního okna aplikace na třetiny. Dvě třetiny zleva jsou vyhrazeny pro zobrazení zvolených údajů. Tento výběr je realizován pomocí ovládacího prvku `JTabbedPane` (klasické záložky známé z grafických rozhraní počítačových programů). Celkem jsou použity dvě záložky mezi kterými je možno přepínat:

- *Info* - na tuto záložku byly přemístěny stávající prvky zajišťující logování komunikace s robotem a dále pak přidány dva panely. Jeden pro zobrazování aktuálního obrazu skeneru, druhý pak pro budoucí zobrazování výstupu stereometrické či webové kamery.
- *Map* - tato záložka je celá vyhrazena mapě budované robotem a také je zde možno zadávat požadavky pro přesun robotu na vybrané souřadnice.

Pravá třetina hlavního okna aplikace *HeroGUI* zůstala nezměněna. Slouží pro ruční řízení robotu a je tedy vhodné, aby tyto prvky byly pořád na vrchu obrazovky v aktivní oblasti, například kvůli potřebě náhlého zastavení robotu.



Obrázek 6.3: Grafické uživatelské rozhraní pro vzdálené ovládání robotu Hero

Komunikační protokol pro výměnu informací mezi palubní stanicí robotu a operátorskou aplikací je definován v [2] a [4]. Tento protokol bylo třeba rozšířit o definice nových zpráv. Nově definované výzvy robotu a odpovědi jsou popsány v tab. 6.1 a 6.2.

Výzva	Význam
`state_scan`	Požadavek na zaslání obrazu z laserového skeneru
`state_map`	Požadavek na zaslání mapy prostředí
`navigate_to=x;y`	Požadavek na zahájení automatického přesunu na souřadnice zadané hodnotami x a y
`navigation_cancel`	Přerušení automatického přesunu robotu (zastavení)

Tabulka 6.1: Přehled výzev

Odpověď	Význam
`scan=`n_1`;`n_2`;...;`n_{682}	Nasnímaná data ze skeneru, kde n_i odpovídá hodnotě daného vzorku. Hodnoty jsou přenášeny v milimetrech a formátovány na pevnou délku čtyř číslic na každou hodnotu.
`map=...=`	Mapa okolí sestavená robotem ve formátu JPEG a přenesená za použití kódování Base64. Zpráva má proměnnou délku a je ukončena rovnítkem.
`navigation_start`	Podařilo se najít cestu v mapě na požadované souřadnice a robot zahájil přesun
`navigation_complete`	Robot úspěšně dokončil přesun do požadovaného bodu
`navigation_failed`	Požadovaná cílová poloha robotu je nedosažitelná

Tabulka 6.2: Přehled odpovědí

Kapitola 7

Aplikační list

V této kapitole jsou popsány nejdůležitější kroky potřebné pro zprovoznění robotu.

7.1 Komunikace a řízení

Zprovoznění komunikace mezi robotem a ovládací aplikací sestává ze tří kroků.

1. **Síťová komunikace**

   Na robotu je nainstalován přístupový bod sítě WiFi. Tento může pracovat buď jako klient (připojení k jiné WiFi síti) nebo jako Access Point (poskytování vlastní WiFi síťe). Je nakonfigurován jako AP poskytující síť „HeroWifi“, přístupové heslo je „hero“. Na tomto zařízení také běží DHCP server, který přiřazuje IP adresy připojeným zařízením. Webové administrační rozhraní přístupového bodu je dostupné na IP adrese 192.168.1.10, řídicí počítač robotu na IP 192.168.1.11. Přístupový bod lze libovolně překonfigurovat pomocí webového rozhraní, přístupové heslo není vyžadováno.

2. **Spuštění řídicí aplikace v robotu**

   V robotu je třeba spustit řídicí aplikaci *control* uloženou v adresáři */home/control*. Tohoto lze nejsnáze dosáhnout připojením k robotu pomocí protokolu SSH za použití programů SSH či Putty. Uživatelské jméno pro přístup k operačnímu systému robotu je „root“, heslo je „hero“. V robotu lze, pro větší uživatelský komfort, spustit správce souborů příkazem `mc`. Řídicí aplikace může být nakonfigurována tak, aby se spouštěla automaticky po startu řídicího PC. Postup konfigurace je uveden v [2].

Pokud by nebylo možné vzdálené připojení k příkazové řádce OS robotu (třeba kvůli nefungujícímu síťovému spojení), je nutné k robotu připojit obrazovku a klávesnici (konektory na levé stěně robotu, klávesnice může být PS/2 i USB) a provést konfiguraci lokálně. V této fázi je také nutné zprovoznit síťové připojení. Návod na konfiguraci OS Gentoo Linux lze nalézt na [16], k dipozici je na odkazovaných stránkách i v češtině.

3. **Spuštění ovládací aplikace v uživatelském PC**

   V ovládacím PC (například notebook připojený k WiFi síti robotu) je nutné spustit ovládací aplikaci HeroGUI. K tomu je nutné mít nainstalováno prostředí JRE (*Java Runtime Environment*). Případně lze také provést spuštění ovládací aplikace přímo ze zdrojových kódů ve vývojovém prostředí NetBeans. Po spuštění je třeba v ovládací aplikaci zadat adresu robotu (nastaveno na 192.168.1.11) a pomocí tlačítka „Connect“ se připojit. Po připojení začne komunikace mezi robotem a řídicím programem, budou aktualizovány údaje z robotu, začne se tvořit mapa prostředí a bude možné robot řídit. Podrobně jsou funkce ovládací aplikace popsány v [2].

Při vypínání robotu je nanejvýš vhodné zadat do příkazové řádky robotu příkaz `poweroff`, počkat jednu minutu a poté robot vypnout hlavním vypínačem. Tímto příkazem dojde k ukončení činnosti OS robotu a k odpojení připojené paměťové karty tak, aby se předešlo ztrátě dat.

7.2 Nabíjení akumulátorů

Pokud je robot vypnut hlavním vypínačem, není z akumulátorů odebírána žádná energie. Nicméně i tak dochází k pozvolnému samovybíjení akumulátorů, které by je mohlo časem zničit. Pokud bude tedy robot delší dobu skladován, je třeba zajistit pravidelné dobíjení akumulátorů v intervalu cca 1 - 2 měsíce.

Pro nabíjení je potřeba připojit robot přiloženým kabelem ke zdroji napětí 15 - 24 V stejnosměrných a stisknout zelené tlačítko na robotu. Poté začne nabíjení akumulátorů řízené zabudovanými nabíječkami. Konec nabíjecího cyklu poznáme podle toho, že na nabíječkách svítí pouze zelené kontrolky. Bližší informaci o elektrické výbavě robotu lze nalézt v [1] a [3].

Kapitola 8

Závěr

8.1 Řešení práce

Tato práce navazuje na bakalářské práce řešené v letech 2007 až 2008 a zabývá se dopracováním mechanické koncepce a senzorické výbavy robotu Hero. Pro úspěšné řešení práce bylo potřeba navrhnou a realizovat popsané mechanické úpravy a rozšířit elektronickou výbavu robotu o výše popsané senzory. Bylo také navrhnuto budoucí možné rozšíření v podobě integrace stereometrické kamery do senzorického systému robotu.

Druhou velkou částí práce byla úprava stávajícího softwarového vybavení robotu tak, aby řídicí aplikace byla schopna zpracovat a smysluplně využít informace poskytované novými senzory. Byly implementovány základy algoritmů pro mapování prostředí a plánování trasy a zároveň i algoritmy postihující výpočet kinematiky mechanického manipulátoru.

Výsledkem této práce je rozšíření řídicího programu, díky němuž je robot schopen samostatného pohybu v prostředí na zadané souřadnice. Toho je docíleno sestavením mapy prostředí a plánováním cesty v prostředí. Všechny povely jsou zadávány robotu pomocí řídicí aplikace *HeroGUI*, která byla také příslušně rozšířena.

8.2 Možná vylepšení

Mezi možná vylepšení patří například instalace zmíněné stereometrické jednotky, případně rozšíření senzorického systému tak, aby byl robot schopen rozeznávat předměty v prostředí a uchopovat je pomocí ruky. Dále by bylo vhodné se zaměřit na ovládací uživatelskou aplikaci, zejména s ohledem na odstranění nepoužívaných funkcí a zkvalitnění

komunikačního modelu mezi robotem a uživatelským rozhraním. Také by bylo dobré jak v robotu, tak v ovládací aplikaci, vhodně ošetřit nestandardní události jako je například rozpadnutí síťového spojení, a celkově tak zvýšit robustnost řídicího systému. Z tohoto důvodu lze také doporučit montáž STOP tlačítka na hlavu robotu. Na stisk tohoto tlačítka by reagovala řídicí aplikace a okamžitě zastavila běh všech instalovaných pohonů.

Co se týče mechaniky robotu, tak by bylo nanejvýš vhodné upravit zdeformovaná kola pojezdu tak, aby opět měla ideální válcový tvar, například opracováním na soustruhu. Současný stav kol ztěžuje robotu plynulý pohyb.

Literatura

[1] J. Zemánek: *Bakalářská práce: Robot HERO.*
Katedra řídící techniky, České vysoké učení technické v Praze , 2007

[2] O. Šantin: *Bakalářská práce: Robot HERO.*
Katedra řídicí techniky, České vysoké učení technické v Praze , 2007

[3] O. Zeman: *Bakalářská práce: Robot HERO.*
Katedra řídicí techniky, České vysoké učení technické v Praze , 2008

[4] O. Šantin: *Semestrální projekt: Robot HERO - Řízení ruky.*
Katedra řídicí techniky, České vysoké učení technické v Praze , 2008

[5] O. Ton: *Diplomová práce: Orientace v prostoru.*
Katedra řídicí techniky, České vysoké učení technické v Praze , 2008

[6] Hokuyo Automatic co., ltd: *Scanning Laser Range Finder URG-04LX Specifications*
Hokuyo Automatic co., ltd, Osaka, 2008

[7] Hokuyo Automatic co., ltd: *URG Series Communication Protocol Specification (SCIP Ver1.1)*
Hokuyo Automatic co., ltd, Osaka, 2009

[8] National Semiconductor corp.: *LM2575 1A Step-Down Voltage Regulator*
National Semiconductor Corporation, Santa Clara, 2007

[9] Sharp corp.: *GP2D12 Datasheet*
Sharp Corporation, Tokyo, 2002

[10] Mark W. Spong, Seth Hutchinson, M. Vidyasagar: *Robot modeling and control*
John Wiley & sons, inc., 2006

[11] Reza N. Jazar: *Theory of applied robotics*
Springer, 2007

[12] Bruno Steux, Oussama El Hamzaoui: *CoreSLAM : a SLAM Algorithm in less than 200 lines of C code*
Mines ParisTech - Center of Robotics, Paris, 2010

[13] Sebastian Thurn, Wolfram Burgard, Dieter Fox: *Probabilistic robotics*
The MIT Press, Cambridge, Massachusetts, 2005

[14] Alex Nash, Kenny Daniel, Sven Koenig, Ariel Felner: *Theta*: Any-Angle Path Planning on Grids*
University of Southern California, 2007

[15] Independent JPEG Group: *libJPEG - A reference implementation of JPEG format*
http://www.ijg.org/, 2010

[16] Gentoo Foundation: *Gentoo Linux*
www.gentoo.org

[17] Heathkit Zenith Educational Systems: *ET-18 ROBOT ARM MANUAL*
HEATH COMPANY, Benton Harbor, 1982

[18] Maxon: *EPOS Firmware Specification*
maxon motor a.g. 5/2007

Příloha A

Obsah přiloženého DVD

K práci je přiloženo DVD s následující strukturou:

source	Zdrojové kódy řídicí aplikace a uživatelského rozhraní
image	Obraz disku (záloha) řídicího počítače robotu
matlab	Zdrojové kódy systému Matlab použité při řešení práce. Především se jedná o výpočet kinematiky a plánování trasy.
datasheet	Odkazované katalogové listy součástek a senzorů použitých na robotu
ZemanDP.pdf	Tato práce ve formátu PDF