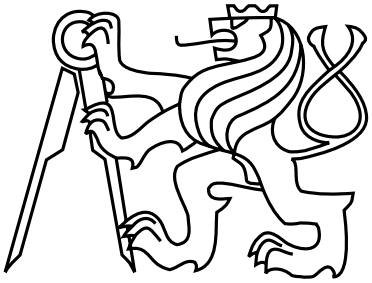
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra měření



Diplomová práce

Demonstrační robotická platforma

Bc. Michal Vokáč

Vedoucí práce: Ing. Michal Sojka Ph.D.

Studijní program: Kybernetika a robotika

Obor: Senzory a přístrojová technika

3. ledna 2012

Prohlášení

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

V Hostivici dne 2.1.2012

podpis

Poděkování

Velmi děkuji vedoucímu své diplomové práce, Ing. Michalovi Sojkovi, Ph.D., za poskytnutí možnosti podílet se na zajímavém projektu a za všechny cenné a užitečné rady, které jsem od něho získal za celou dobu spolupráce a děkuji za jeho trpělivost, ochotu a odbornou pomoc, kterou mi poskytl během zpracování této diplomové práce.

Velmi rád bych poděkoval Ing. Pavlovi Píšovi za jeho čas, který věnoval nejen mě, ale i dalším studentům a členům týmu Flamingos při řešení problémů týkajících se řízení motorů v našich robotech a bez jehož pomoci by se letos žádný z našich robotů pravděpodobně nerozjel.

Velmi děkuji mé přítelkyni a rodině za jejich podporu během celé doby studia.

Abstrakt

Tato práce popisuje návrh a vytvoření mobilního robotu pro demonstrační účely. Pro vytvoření demonstračního robotu je použita základní mechanická konstrukce a elektronické komponenty z robotu použitého týmem Flamingos při soutěži Eurobot 2010. Je zde popsán návrh a realizace hlavního stavového automatu demonstrační aplikace a jeho spolupráce s ostatními částmi software robotu. Dále práce obsahuje návod k obsluze vytvořeného demonstračního robotu.

Abstract

This thesis presents design and creation of a demonstration mobile robot. The created demonstration robot is based on mechanical structure and electronic components which were used in Flamingos robot at Eurobot 2010 robotic contest. It describes design and realization of a main finite state automaton of the demonstration application and its cooperation with other software parts. It also includes an instruction manual for the demonstration robot.

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2010-2011**

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student:	**Bc. Michal Vokáč**
Program:	**Kybernetika a robotika**
Obor:	**Senzory a přístrojová technika**
Název tématu česky:	**Demonstrační robotická platforma**
Název tématu anglicky:	**Demonstration Robotic Platform**

Pokyny pro vypracování:

1. Seznamte se s robotem vyvinutým na katedře řídicí techniky pro soutěž Eurobot 2010.
2. Navrhněte způsob, jak tohoto robota předělat pro účely demonstrace (dny otevřených dveří apod.). Výsledek by měl splňovat následující požadavky:
   a) robot bude obsluhovatelný kýmkoliv, tj. ihned po zapnutí začne provádět požadovanou činnost;
   b) pro spuštění nebudou nutné specializované prostory, bude stačit podlaha libovolné místnosti;
   c) robot bude reagovat na své okolí, tj. bude se vyhýbat překážkám a bude registrovat změny v poloze objektů, s nimiž pracuje.
3. Realizujte navržené úpravy a zaměřte se na spolehlivost a bezúdržbovost demonstrátoru.
4. Vše pečlivě otestujte a zdokumentujte, zejména vytvořte návod(y) pro obsluhu.

Seznam odborné literatury:

[1] Kubias Jiří: Hardware robota pro soutěž Eurobot. Diplomová práce ČVUT FEL, Praha 2010.
[2] Jareš Filip: Implementace stavových automatů pro soutěž Eurobot 2009. Bakalářská práce ČVUT FEL, Praha 2010.

Vedoucí diplomové práce:	Ing. Michal Sojka (K 13135)
Datum zadání diplomové práce:	24. ledna 2011
Platnost zadání do[1]:	29. června 2012

Prof. Ing. Pavel Ripka, CSc.
vedoucí katedry

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

Prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 24. 1. 2011

[1] Platnost zadání je omezena na dobu tří následujících semestrů.

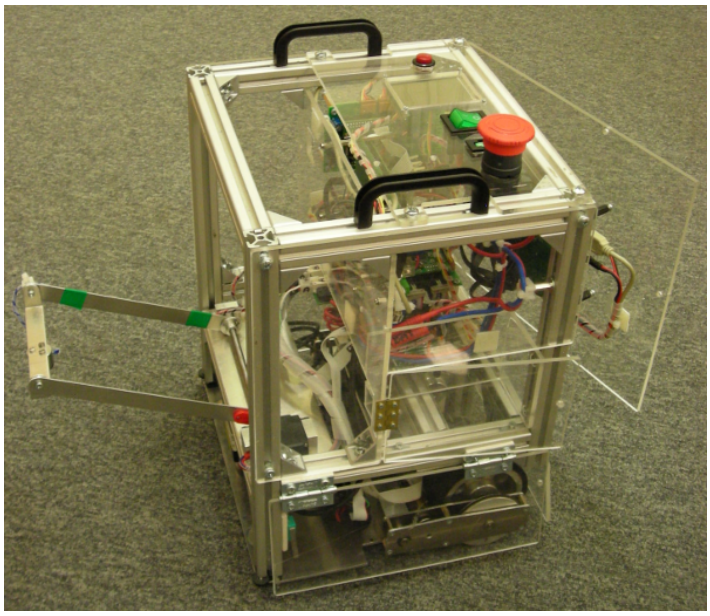
Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod



Obrázek 1.1: Fotografie demonstračního robotu

Cílem této práce bylo vytvořit demonstrační mobilní robot vhodný pro prezentaci fakulty na dnech otevřených dveří, na středních školách a při dalších příležitostech. Účelem je interaktivní formou uchazečům o studium nebo i současným studentům fakulty předvést technologie z různých oblastí měření, řízení, komunikací, robotiky a dalších, které byly vyvinuté na fakultě. Cílem je také pozitivně motivovat studenty, aby se při studiu zapojili do podobných projektů ve kterých mohou zajímavým způsobem rozšířit své vědomosti.

Na vyvíjený robot jsou kladeny následující požadavky:

- robot se bude pohybovat ve vnitřním prostředí budov, tedy nejsou kladeny poža-

davky na pohyb po nerovném a nezpevněném povrchu a na ochranu před nepříznivými vlivy (voda..),

- robot bude schopen se autonomně pohybovat v neznámém prostředí,
- robot se bude vyhýbat překážkám a osobám,
- robot bude v určeném prostoru plnit předepsanou úlohu a reagovat na změnu polohy prvků se kterými pracuje.

Podrobněji je návrh demonstrační aplikace popsán v kapitole 2.

Základním požadavkem kladeným na vytvoření takového demonstračního robotu bylo při návrhu vycházet z konstrukce použité robotickým týmem Flamingos [23] z katedry řídicí techniky při mezinárodní soutěži Eurobot [10] v roce 2010.

Většina součástí hardware i software robotu je produktem dlouhodobé spolupráce mnoha studentů a pedagogů a mnoho částí robotu bylo vyvinuto v rámci jejich bakalářských či diplomových prací. Základy celému systému položil vedoucí týmu Ing. Michal Sojka, Ph.D., který je také autorem kompletní architektury software na robotu. Na vývoji všech součástí software se různou měrou podíleli Ing. Pavel Píša, Duy Khanh Tran, Martin Žídek [8], Petr Beneš [5], Konrad Skup [7] a mnoho dalších členů týmu Flamingos i další neznámí autoři GNU GPL software, který je v robotu využíván.

Základní mechanické části robotu byly navrženy Janem Bendou v jeho diplomové práci [4] v roce 2008. Tyto součásti byly posléze upraveny a sestaveny současnými členy týmu podle aktuálních požadavků soutěže. Elektronické desky instalované v robotu byly v rámci diplomové práce navrženy a vyrobeny Jiřím Kubiasem [1].

Autorem hlavní soutěžní aplikace v roce 2010 byl Filip Jareš, který stavový automat aplikace vyřešil ve své bakalářské práci [2]. Pro zpracování obrazu z kamery a rozpoznávání konfigurace herních prvků byl použit program *rozkuk* vyvinutý Petrem Kubizňákem v jeho bakalářské práci [6].

Většina hardware i software součástí vytvořeného robotu je tedy dílem jiných studentů. Mnoho těchto částí ovšem nebylo nikdy dobře zdokumentováno a nalezené chyby nebyly většinou nikam zaznamenány pro pozdější kontrolu. Přínosem této práce je tedy revize všech součástí použitých nejen na vyvinutém demonstračním robotu a komplexní dokumentace těchto součástí a nalezených chyb.

Vyvinutý robot, viz obrázek 1.1, je mobilní robot s dvoukolovým diferenciálně řízeným podvozkem. Jeho hlavní konstrukce zůstala téměř zachována. Hlavní mechanickou změnou oproti výchozí verzi je změna v návrhu manipulátoru pro sbírání prvků a v rozmístění prvků elektroniky. Ty jsou nyní v těle robotu rozmístěny tak, aby byly

kdykoliv snadno přístupné pro opravu. Podrobný popis konstrukční části robotu se nachází v kapitole 3.

Elektronický subsystém robotu, který je popsán v kapitole 4, byl upraven s ohledem na maximální spolehlivost. To se týká především kabeláže a konektorů, které byly vždy příčinou mnoha problémů s roboty nasazenými v soutěži.

Všechny části software používané na robotu pro účast v soutěži musely být upraveny a přizpůsobeny požadavkům demonstrační aplikace, které jsou zcela odlišné od požadavků kladených na nasazení robotu v soutěži. Veškerý software je složen z obrovského množství spolu propojených modulů řešících specifické úlohy jako je detekce překážek, vytváření mapy prostředí, plánování pohybu v prostředí, řízení pohybu, řízení aktuátorů, hlavní stavový automat aplikace atd. Úpravám provedeným ve všech částech software je věnována kapitola 5.

Kapitola 6 této práce pak slouží jako manuál pro obsluhu vytvořeného demonstračního robotu. Je zde popsán postup jak manipulovat s robotem, příprava prostředí a celý proces od zapnutí napájení po odstartování aplikace.

Kapitola 2

Návrh demonstrační aplikace

Před vlastní přestavbou výchozího hardware, úpravami elektroniky a software bylo potřeba navrhnout proces jak bude prezentace demonstrační aplikace (demo) probíhat a jakou úlohu bude robot řešit.

Základní požadavky kladené na návrh dema byly popsány v úvodní kapitole. Předpokládá se tedy provoz dema uvnitř budovy. Například v prostoru učebny a laboratoře, na chodbě, v kanceláři a podobně.

V následujících sekcích budou upřesněny jednotlivé požadavky kladené na vyvíjený robot. Budou zde popsány požadavky na interakci robotu s okolím, definice prostoru ve kterém se bude robot pohybovat a návrh úlohy, kterou bude robot v rámci vymezeného prostoru řešit.

2.1 Reakce na překážky v prostoru

Vzhledem k možnosti provozu dema v odlišných prostorech tedy nelze předem provést žádné rozhodnutí o charakteru tohoto prostoru. Především se jedná o možnost předem vytvořit mapu prostředí podle které se poté robot v prostoru naviguje a zakreslit do mapy překážky na známých pozicích. Robot tedy musí být schopen se autonomně pohybovat v naprosto neznámém prostoru. V tomto se demo odlišuje od situace při soutěži Eurobot pro kterou byl původní software navržen. Při přípravě herní strategie pro soutěž lze totiž s využitím pravidel vydaných pro aktuální ročník soutěže upravit mapu podle technických výkresů hracího pole a umístit do mapy všechny překážky a omezení tak, jak jsou popsány v pravidlech a zakresleny v dokumentaci.

Zároveň je nutné v prostoru spuštění dema počítat s pohybem osob. Nelze tedy všechny detekované překážky zakreslit do mapy napevno. Pokud by například robot detekoval procházející osobu, vytvořil by tak v mapě dlouhou zeď, která ovšem ve

skutečném prostoru není.

Popisu realizovaného řešení vytváření mapy a zakreslování překážek se věnuje sekce 5.3.6.4.

2.2 Omezení velikosti prostoru

Další odlišností dema od soutěže je velikost prostoru ve kterém se robot pohybuje. Standardně se v software, tak jak je připraven pro soutěž, počítá s velikostí hracího pole o rozměru přibližně 3 m na šířku a 2 m na výšku. Na tento rozměr je pak omezena velikost mapy a robot tedy není schopen se pohybovat mimo tento rozsah.

Pro provoz dema v předpokládaných podmínkách by byl takovýto rozměr mapy příliš omezující. Proto bylo rozhodnuto o zvětšení mapy, tedy prostoru kde se robot může pohybovat, na čtverec o straně 5 m.

2.3 Návrh řešené úlohy

Posledním krokem v celkovém návrhu demonstrační aplikace byl návrh úkolu, který bude robot v rámci prezentace řešit. Tento návrh byl proveden tak, aby měl některé společné prvky s pravidly soutěže Eurobot pro rok 2011.

Navržená úloha pak sestává z následujících bodů.

1. Před spuštěním demonstrační aplikace robot stojí ve středu volného prostoru, na výchozí pozici.

2. V okolním prostoru robotu je umístěn jeden cílový objekt – válec o průměru 200 mm a výšce 50 mm.

3. Úkolem robotu je pohybovat se v prostoru a pomocí laserového rangefinderu detekovat tento cíl.

4. Pokud robot detekuje cíl ve svém okolí, přiblíží se k cíli na bezpečnou vzdálenost a pomocí kamery rozhodne, zda se jedná o hledaný cíl.

5. Toto rozhodnutí se provede na základě umístění čárového kódu na objektu.

6. Pokud je cíl považován za platný, provede robot přesné přiblížení k cíli na těsnou vzdálenost pomocí naměřených dat z laserového rangefinderu.

7. Po přesném přiblížení k cíli robot pomocí manipulátoru vyzvedne náklad umístěný na cílovém podstavci. Jako náklad bude použita například plechovka nebo jiný lehký válec z feromagnetického materiálu o maximálním průměru 80 mm a výšce 100 mm.

8. Po vyzvednutí nákladu se robot vrátí na výchozí pozici.

Kapitola 3

Popis demonstračního robotu

Tato kapitola se věnuje popisu konstrukční části robotu. Je zde popsána kostra, návrh a výroba zakrytování, sestava podvozku a návrh a výroba mechanizmu pro zvedání předmětů.

Některé sekce také obsahují návody dokumentující postupy návrhu mechanických dílů a ovládání strojů použitých k jejich výrobě. Tyto návody a odkazy na ně jsou určeny především pro budoucí i současné členy týmu Flamingos, a mají za úkol zjednodušit práci v příštích letech.

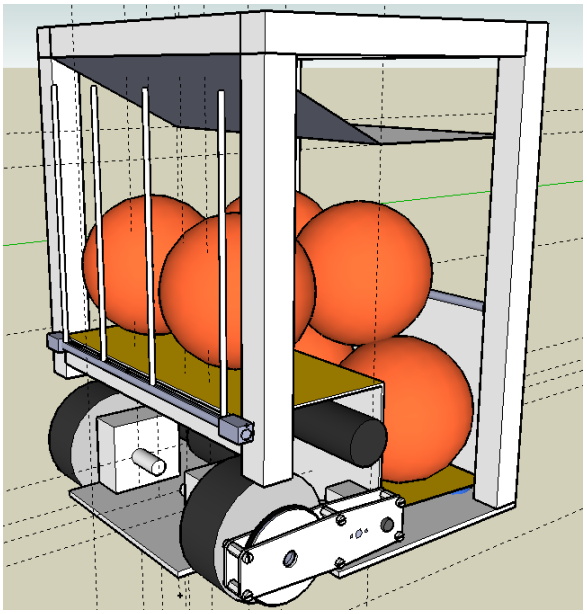
3.1 Kostra

Základní kostra robotu má tvar hranolu o rozměru 260×250×340 mm (š:h:v) a je sestavena z hliníkových profilů stavebnice ITEM 20×20 mm. Tato kostra je upevněna na základnu z duralového plechu o síle 3 mm. Na této části nebyly provedeny žádné úpravy a kostra byla použita tak, jak byla sestavena pro soutěž, viz obrázek 3.1.

3.2 Vnější kryt

Původní zakrytování kostry pomocí bílého plastu bylo odstraněno a nově byla kostra z vnější části pokryta deskami z plexiskla. Plexi materiál byl zvolen především kvůli demonstračním účelům. Veškerá instalovaná technologie je tak pro publikum viditelná bez nutnosti otevírání/snímání krytů. Zároveň tak lze dobře sledovat stav jednotlivých elektronických modulů podle stavu LED, kterými jsou moduly osazeny.

Všechny plexi součásti krytu byly vyrobeny svépomocí na CNC frézce v dílně fakulty.



Obrázek 3.1: Vizualizace konstrukce robotu pro soutěž Eurobot 2010, zdroj: Flamingos

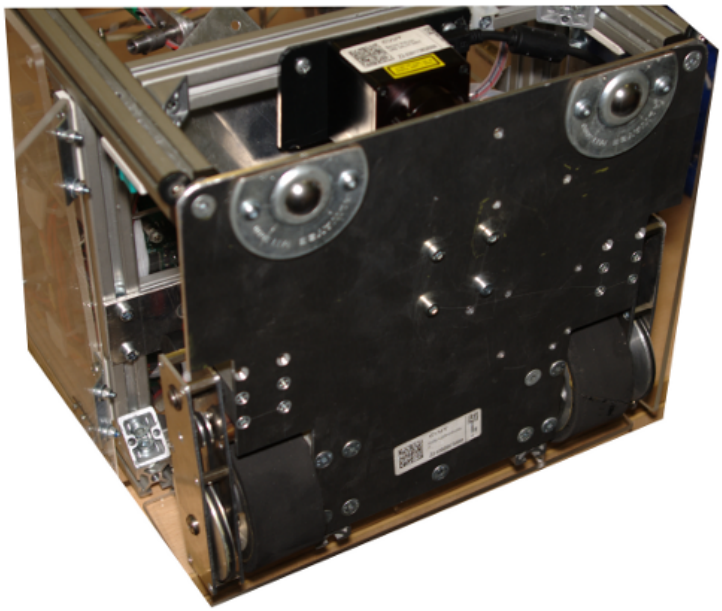
Pro návrh plexi součástí krytu byl použit program progeCAD [11], ke kterému univerzita poskytuje všem studentům a zaměstnancům školní licenci pro výukové a nekomerční účely.

Zpracování CAD výkresů a vytvoření souboru s trasami nástroje pro CNC obráběcí stroj bylo provedeno v programu ArtCAM Insignia [12]. Tento software je nainstalován na PC, které je umístěno u obráběcího stroje. K použití tohoto programu je třeba si vyžádat USB klíč od osoby zodpovědné za provoz frézky.

Pro usnadnění kompletního návrhu výrobku a následného obrábění pomocí této CNC frézky pro všechny další uživatele a především členy týmu Flamingos byl vytvořen podrobný manuál [15] popisující všechny potřebné úkony. Tento manuál je zveřejněn na wiki [14] týmu Flamingos.

3.3 Podvozek

Podvozek robotu, viz obrázek 3.2 je postaven na dvou hnaných kolech a dvou podpěrných kuličkách. Hnaná kola jsou nalisovaná na hřídele uložené v kuličkových ložiscích v masivním hliníkovém bloku, který je pevně spojen se základnou robotu. Na hřídeli je dále nalisovaná ozubená řemenice, která je řemenem spojena s motorem, který pohání dané kolo.



Obrázek 3.2: Podvozek demonstračního robotu

3.3.1 Motory

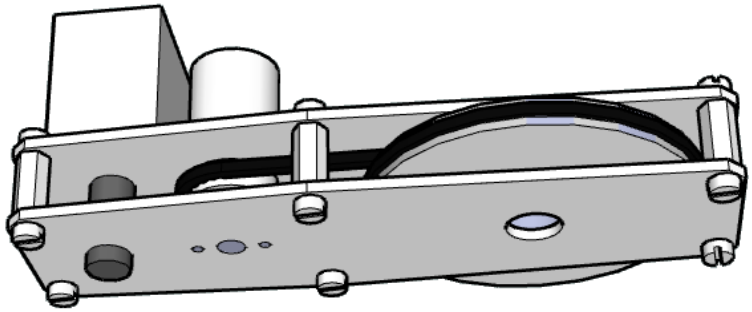
Pro pohon robotu jsou použity BLDC (brushless DC) motory Maxon EC–max 30 s převodovkou 29:1. Tyto motory byly zvoleny především pro jejich vysokou kvalitu, dlouhou životnost a vysoký rychlostní rozsah. Nevýhodou je nutnost použití speciální elektroniky pro buzení motorů třífázovým proudem. Pro více informací o návrhu pohonu, použitých motorech a jejich výběru viz sekce 2.1 v [4].

Pohonné motory v tomto robotu byly v roce 2011 použity dalšími členy týmu Flamingos při oživování a testování nové desky pro řízení motorů (viz [3], kapitola 4), navrhované pro nový typ motoru s odlišným typem přípojení fází a HALL senzorů. Během těchto testů bohužel došlo ke zničení HALL senzorů v motorech použitých v tomto robotu. Tato závada si vyžádala úpravu firmware v procesorovém modulu Hitachi, který ovládá výkonové budiče.

Popis elektroniky pro buzení motorů je v sekci 4.6 a provedená úprava firmware je popsána v sekci 5.5.

3.3.2 Odometrie

Pro účely lokalizace je robot vybaven přídavnými nehnanými koly pro měření odometrie, viz obrázek 3.3. Celý mechanický blok odometrie je přišroubován k základně tak, že osa otáčení hnaných kol je totožná s osou odometrických kol. Odometrická



Obrázek 3.3: Vizualizace návrhu odometrie, zroj: Flamingos

kola jsou řemínkovým převodem spojena s IRC snímačem s rozlišením 4096 kroků na otáčku. Mechanika pro měření odometrie byla navržena a vytvořena týmem Flamingos v roce 2010. Elektronika pro zpracování signálu z IRC snímačů odometrie je popsána v sekci 4.7.

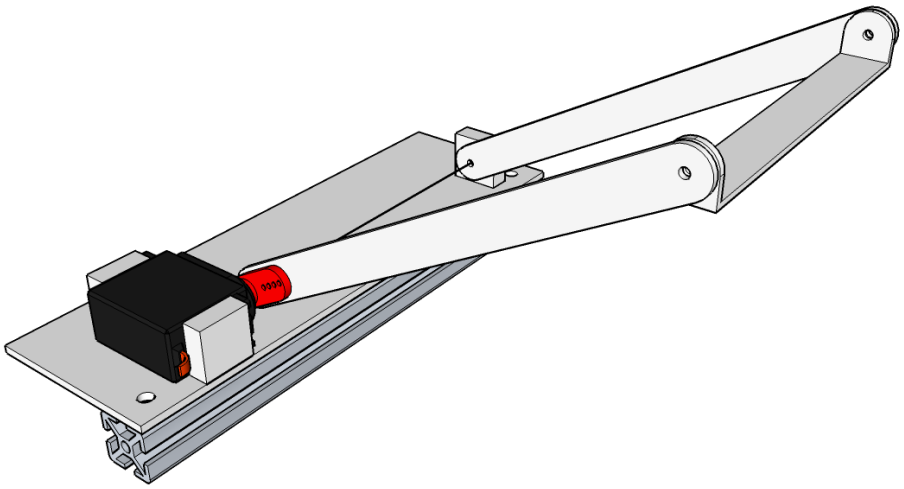
Mechanická část podvozku nebyla oproti výchozímu provedení nijak upravena.

3.4 Návrh mechanizmu pro zvedání předmětů

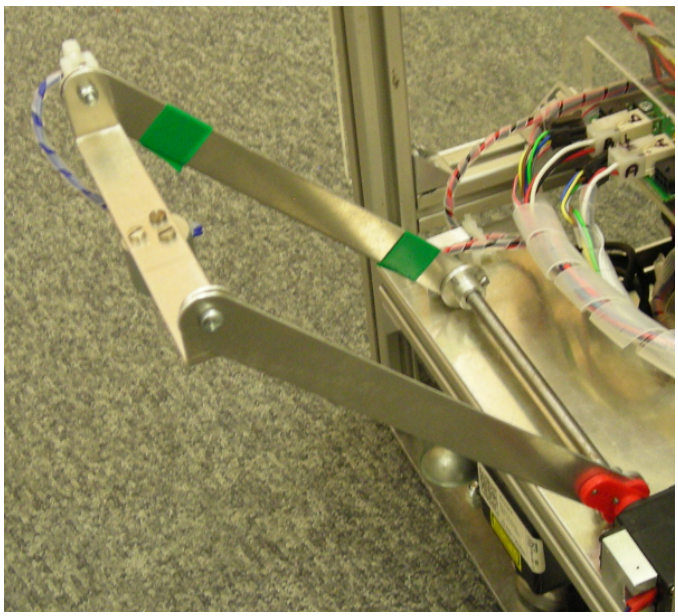
Pro vyzvednutí nákladu umístěného na cílovém podstavci byl navržen a vyroben manipulátor, viz obrázek 3.4 a 3.5 hnaný upraveným modelářským servomotorem. Návrh mechanizmu byl proveden v 3D modelovacím nástroji Google SketchUp [13]. Podle vygenerované technické dokumentace byly následně vyrobeny jednotlivé součásti. Ploché části byly vyrobeny svépomocí vystřižením z hliníkového plechu o síle 1,3 mm. Výroba ostatních částí byla zajištěna v dílnách fakulty ve spolupráci s technikem katedry p. Čmelíkem.

Rameno manipulátoru je vybaveno malým elektromagnetem, který se používá pro uchopení nákladu. Podmínkou pro vyzvednutí nákladu tedy je, aby byl náklad nebo aspoň jeho vrchní část vyroben z feromagnetického materiálu nebo byl opatřen permanentním magnetem. Otestováním manipulátoru bylo zjištěno, že by hmotnost nákladu neměla přesáhnout přibližně 200 g. Při větším zatížení dochází k přetížení servomotoru, který nemá dostatečnou sílu pro zvednutí břemene a hrozí poškození převodovky nebo spálení vinutí motoru.

Popis elektroniky použité pro řízení manipulátoru a provedená úprava servomotoru je v sekci 4.4.



Obrázek 3.4: Vizualizace návrhu mechanizmu manipulátoru



Obrázek 3.5: Vytvořený mechanizmus manipulátoru

Kapitola 4

Elektronika

Všechny části elektroniky robotu, vyjma hlavního řídicího počítače a zobrazovacího modulu s OLED displejem, byly již dříve navrženy a vytvořeny jinými studenty na katedře řídicí techniky jako bakalářské či diplomové práce. Tato elektronika je určena především pro aplikaci v robotické soutěži Eurobot a testování software vyvinutého na katedře.

Celkově se jedná o modulární systém jehož základem je sběrnice CAN. Díky tomu lze jednotlivé moduly libovolně kombinovat a sestavit výsledný systém tak, aby co nejlépe vyhovoval konkrétním požadavkům.

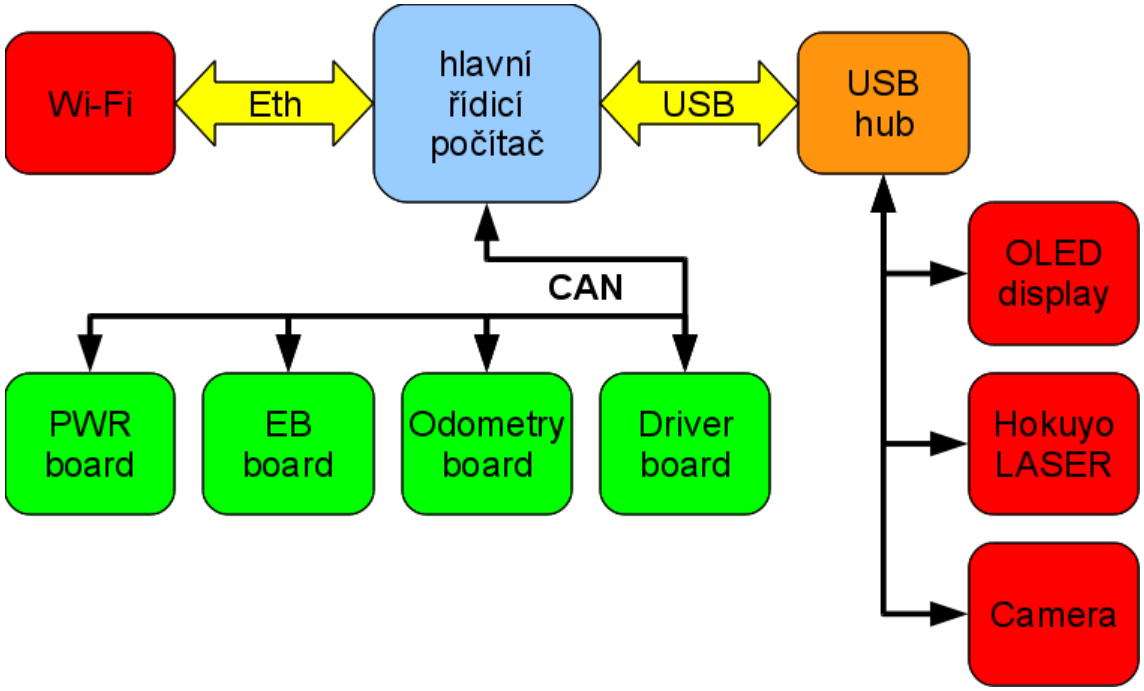
Na obrázku 4.1 je zobrazeno komunikační blokové schéma všech modulů, které jsou použity v demonstračním robotu. V následujících sekcích jsou jednotlivé elektronické součásti podrobněji popsány.

4.1 Napájení robotu

4.1.1 Baterie

Energie pro provoz všech elektronických systémů robotu je získávána z akumulátorové baterie sestavené z osmi sério–paralelně spojených LiFePO4 článků. Jmenovité napětí baterie je 13,2 V a kapacita 4,8 Ah. Tento typ baterie byl zvolen především kvůli schopnosti snášet velmi vysoké nabíjecí proudy (až 4 C). Tato vlastnost je důležitá především při soutěži, kde je nutné dobíjet baterie v krátké době mezi jednotlivými zápasy. Při maximálním povoleném nabíjecím proudu 20 A je tak baterie nabita na 90% své kapacity za 15 minut.

Dalším důvodem pro zvolení tohoto typu baterie je velmi dobrý poměr hmotnosti, objemu a kapacity například ve srovnání s hermetickým olověným akumulátorem.



Obrázek 4.1: Komunikační blokové schéma elektronických komponent robotu

Mnohem dostupnější variantou z pohledu ceny i dostupnosti na trhu by bylo použití srovnatelných akumulátorů typu LiPol. V některých zemích, kde se konají národní a nebo mezinárodní kola soutěže (především Francie), je však použití LiPol baterií omezeno speciálními pravidly především kvůli riziku exploze při poškození nebo přetížení článků.

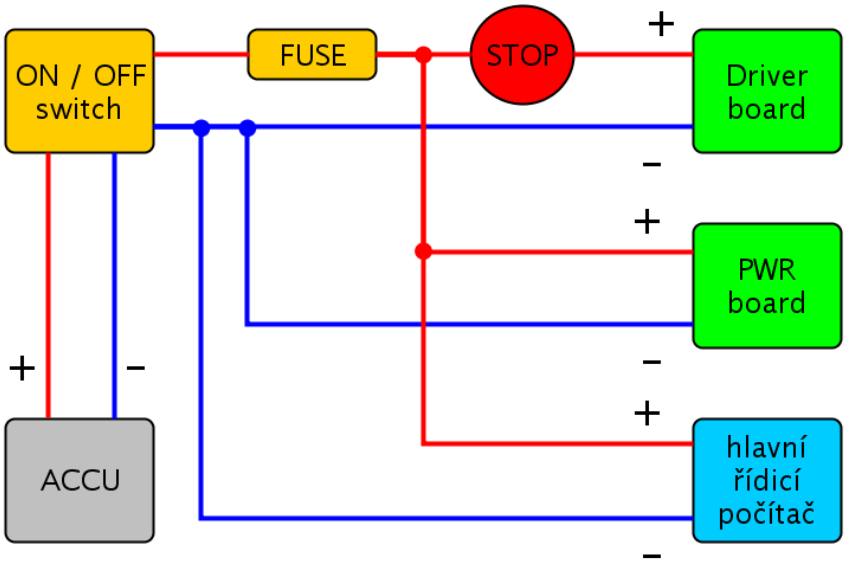
Baterie použité pro napájení demonstračního robotu jsou však již v provozu více než 3 roky a jejich kapacita za tuto dobu značně klesla a budou muset být v blízké době nahrazeny.

Pro více informací o použité baterii, postupu nabíjení a její údržbě viz sekce 5.1–5.3 v [1].

4.1.2 Ochrany

Kvůli zajištění bezpečnosti provozu robotu a ochraně elektroniky jsou v robotu instalovány různé bezpečnostní prvky. Hlavní přívod baterie je jištěn automobilovou pojistkou s tavným proudem 20 A. Úkolem pojistky je především ochrana baterie a přívodních vodičů, protože baterie může do zkratu dodávat proud až několik stovek ampér.

Dalším ochranným prvkem je červené bezpečnostní tlačítko. Podle pravidel soutěže



Obrázek 4.2: Schéma propojení napájení z baterie

Eurobot musí být každý robot vybaven červeným bezpečnostním tlačítkem. Toto tlačítko musí být umístěno na robotu tak, aby k jeho vypnutí došlo pohybem shora dolů. Stisknutím tlačítka dojde k přerušení přívodu energie do pohonných motorů robotu a zastavení robotu. Napájení všech ostatních součástí je i po stisku STOP tlačítka zachováno. Použití tohoto tlačítka bylo zachováno i na demonstračním robotu.

Obrázek 4.2 zobrazuje celkové schéma propojení napájení z baterie k jednotlivým modulům.

4.2 Procesorový modul LpcEurobot

Tento univerzální modul je určen pro sběr dat a řízení různých aktuátorů. Jedná se o malý (46×43 mm) modul s procesorem NXP LPC2119 s jádrem ARM7. Pro snadné nahrávání firmaware a komunikaci je modul vybaven miniUSB konektorem a USB–RS232 převodníkem. Fotografie modulu je na obrázku 4.3.

Tento modul je v demonstračním robotu použit na desce napájení PWR board a desce pro řízení manipulátoru – EbBoard.

Více podrobností o tomto modulu viz sekce 3.7 v [1].

Obrázek 4.3: Procesorový modul LpcEurobot, zdroj: Jiří Kubias [1]

4.3 Elektronika napájení – PWR board

Napájecí deska PWR board je určena k přípravě napájecích napětí, která jsou potřebná pro napájení jednotlivých elektronických součástí robotu. K desce napájení je přímo přivedeno napětí z baterie a pomocí tří nezávislých spínaných zdrojů deska vytváří napájecí napětí 3,3 V, 5 V a 8 V. Tato napájecí napětí jsou poté určena k distribuci ke všem dalším elektronickým součástem robotu. Každá napájecí větev je chráněna vlastní pojistkou s tavným proudem 3 A.

Deska napájení je dále osazena univerzálním procesorovým modulem LpcEurobot, viz sekce 4.2. Firmware v procesorovém modulu zajišťuje ochranu proti přepětí a podpětí na všech napájecích větvích. Pokud se napětí větve vychýlí z nastaveného rozsahu, dojde k odpojení příslušné napájecí větve. Další funkcí modulu je ochrana proti přílišnému vybití akumulátorové baterie. V případě, že dojde k poklesu napětí na baterii pod stanovenou mez, procesorový modul postupně podle stavu vybití baterie odpojuje jednotlivé napájecí větve.

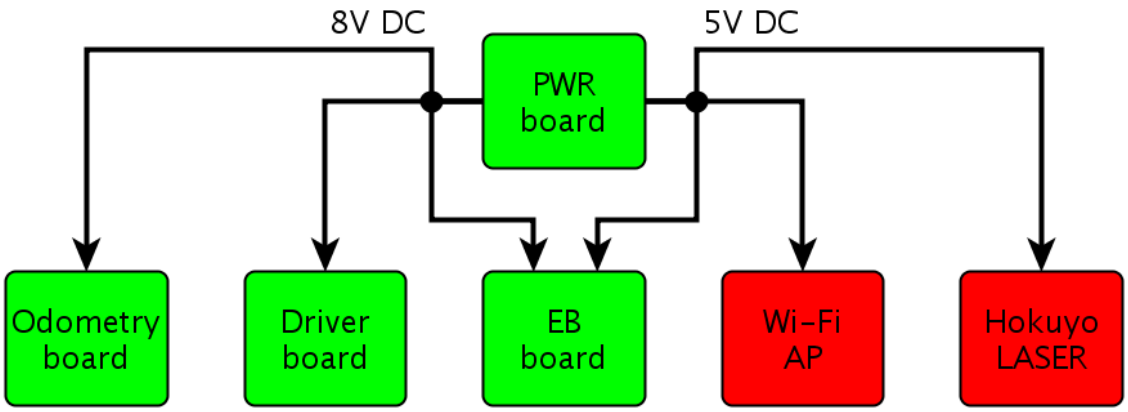
Na obrázku 4.4 je zobrazeno rozvedení distribuovaného napájení z desky PWR board k jednotlivým elektronickým součástem robotu.

Pro více podrobností o desce napájení viz sekce 3.8 v [1].

4.4 Elektronika pro řízení manipulátoru – EB board

Elektronika pro řízení manipulátoru je složena ze základní výkonové desky a procesorového modulu. Jako procesorový modul je použita deska LpcEurobot 4.2.

Základní deska je osazena integrovaným výkonovým můstkem pro buzení dvou DC motorů s maximálním trvalým zatížením 2,8 A. Dále jedním výkonovým spínačem pro



Obrázek 4.4: Schéma propojení distribuovaného napájení

spínání zátěže do 3 A trvale. Deska umožňuje zpracování čtyř napěťových analogových signálů v rozsahu 0–3,3 V, které jsou předzpracovány pomocí analogových filtrů typu dolní propust druhého řádu s mezní frekvencí $F_m = 100$ Hz pro odstranění rušení. Deska je také připravena pro připojení tří standardních modelářských servomotorů.

Pro pohon ramene manipulátoru je použit modelářský servomotor standardní velikosti 40,6×20,3×35,6 mm, typ Hitec HS–303. Pro potřeby demonstrační aplikace byl servomotor upraven tak, že byla odstraněna původní elektronika regulátoru a byly přímo vyvedeny přívody pro napájení motoru a také přívody k potenciometru, který slouží k měření úhlu natočení výstupní hřídele, viz obrázek 4.6.

Po této úpravě je možné pro řízení servomotoru použít vlastní elektroniku a jako zpětnou vazbu pro regulátor využít signál z potenciometru. Přínosem této úpravy je právě možnost zasáhnout vlastním způsobem do zpětné vazby z potenciometru a v případě potřeby úplně přerušit přívod energie do motoru.

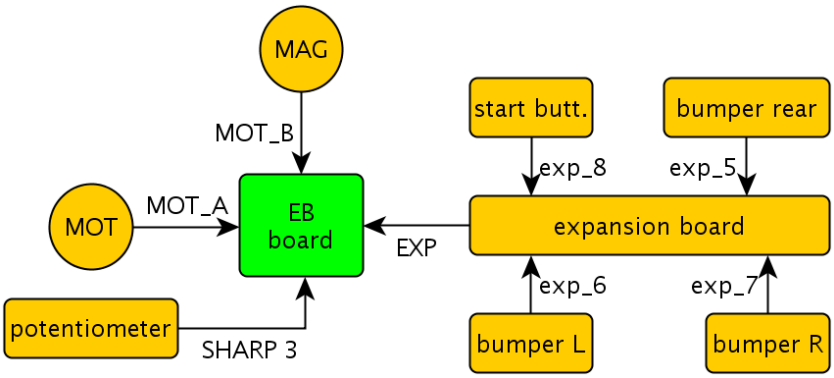
Procesorový modul na desce tedy realizuje regulátor polohy ramene manipulátoru podle zpětné vazby z potenciometru. Motor servopohonu je připojen k prvnímu výstupu integrovaného můstku.

Ke druhému výstupu můstku je připojen malý elektromagnet, který je umístěn na konci ramene. Elektromagnet se používá pro uchopení nákladu.

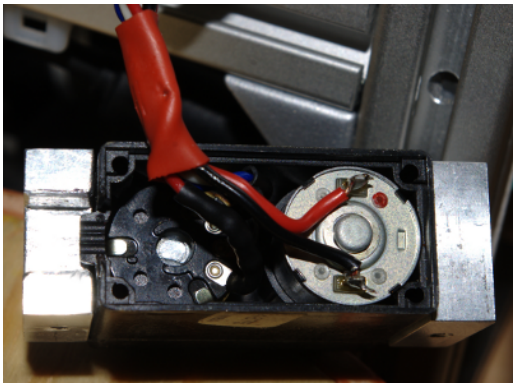
Dalším úkolem této desky je zpracování signálu ze startovacího tlačítka, které se používá pro odstartování demonstrační aplikace.

K desce Eb board je dále připojena malá rozšiřující deska, ke které jsou připojeny taktilní snímače používané pro detekci nárazu do překážky při otáčení nebo couvání.

Blokové schéma zapojení elektroniky manipulátoru a připojení startovacího tlačítka a taktilních snímačů do rozšiřující desky je na obrázku 4.5. Pro podrobnější informace



Obrázek 4.5: Blokové schéma zapojení desky Eb board



Obrázek 4.6: Úprava modelářského servomotoru

o desce Eb board viz sekce 3.8 v [1].

4.5 Hlavní řídicí počítač

Jako hlavní řídicí počítač celého robotu slouží mikroprocesorový modul MIDAM Shark [16]. Modul je osazen procesorem Freescale MPC5200 s jádrem PowerPC®. Na modulu je osazena paměť SDRAM 128MB a FLASH 64MB.

Procesorový modul je usazen v základní desce RYU_edu, viz sekce 3.5 v [1]. Základní deska je osazena veškerou elektronikou potřebnou pro fungování procesorového modulu a podpůrnými obvody, které jsou podstatné pro funkci komunikačních rozhraní použitých v robotu. Jedná se především o budiče sběrnice CAN, síťové rozhraní 10/100T Ethernet, USB host a USB–RS232 převodník pro připojení pomocí sériové konzole.

Popis všech aplikací spuštěných na této desce, které přímo souvisí s provozem demonstračního robotu jsou popsány v kapitole 5.

4.6 Elektronika pro řízení BLDC motorů Driver board

Elektronika pro řízení pohonných BLDC motorů sestává ze základní výkonové desky a procesorového modulu Hitachi. Pro detailní popis obou částí elektroniky viz sekce 3.11–3.13 v [1].

4.6.1 Výkonová část

Základní deska je připravena pro připojení fázových vodičů, HALL senzorů a IRC snímačů z obou motorů. Na základní desce jsou osazeny integrované budiče třífázového výkonového můstku a MOSFET tranzistory tvořící můstek pro spínání fází motoru.

Pro přenos signálu z IRC snímačů na motorech do desky je použita fyzická vrstva sběrnice RS422. Z toho důvodu jsou na desce osazeny dva diferenciální linkové přijímače SN75175.

Při testování a kontrole této desky před její montáží do robotu bylo odhaleno několik chyb. Nalezené chyby byly zaznamenány na robotické wiki na stránce věnující se popisu této desky [17]. Jedná se o nesoulad mezi dokumentací a skutečným stavem desky v následujících bodech. Všechny nalezené chyby byly pouze zdokumentovány, ale nebyly v podkladech opraveny, protože nebyl k dipozici žádný volný software pro úpravu těchto podkladů.

1. Chyba – konektory pro připojení fází motoru
   (a) Ve schématu je konektor J25 popsán ENGINE1, na desce je označen ENGINE2.
   (b) Ve schématu je konektor J26 popsán ENGINE2, na desce je označen ENGINE1.
2. Chyba – konektory pro připojení IRC snímačů na motorech
   (a) Ve schématu je konektor J4 popsán IRC1 a na desce je označen IRC A, ve skutečnosti je signál z tohoto konektoru zpracován jako by byl od motoru B.
   (b) Ve schématu je konektor J5 popsán IRC2 a na desce je označen IRC B, ve skutečnosti je signál z tohoto konektoru zpracován jako by byl od motoru A.

3. Chyba – konektor patice propojující PWM výstupy procesoru s budiči

   (a) Ve schématu je konektor J23 popsán PWM1, na desce je označen PWM2.

   (b) Ve schématu je konektor J24 popsán PWM2, na desce je označen PWM1.

4.6.2 Procesorový modul Hitachi

Tento modul byl navržen speciálně pro řízení BLDC motorů pro pohon robotu. Základem tohoto modulu je 16–ti bitový mikrokontrolér Hitachi (nyní Renesas) H8S2638. Procesor je vybaven šesti 16–ti bitovými čítači/časovači, které jsou použité pro zpracování signálu z IRC snímačů a HALL senzorů. Předností procesoru jsou dvě specializované PWM jednotky. Ty jsou využity pro generování třífázových průběhů pro napájení motoru. Značnou nevýhodou procesoru je jeho horší dostupnost, vysoká cena, ale hlavně nízký počet zaručených přepisů FLASH paměti (100×).

Také v případě tohoto modulu bylo při jeho testování a kontrole odhaleno několik chyb, které byly zaznamenány na wiki na stránce věnující se popisu procesorového modulu [17].

1. Chyba – konektor patice propojující PWM výstupy procesoru s budiči

   (a) Ve schématu vedou ke konektoru J23, který je popsán PWM2, výstupy procesoru PWM2(A–H). Ve skutečnosti jsou na konektor přivedeny výstupy PWM1(A–H).

   (b) Ve schématu vedou ke konektoru J24, který je popsán PWM1, výstupy procesoru PWM1(A–H). Ve skutečnosti jsou na konektor přivedeny výstupy PWM2(A–H).

2. Chyba – problém se startem procesoru

   Dále byl odhalen problém s resetem procesoru. Pokud nebyl modul s procesorem delší dobou používán a připojen k napájení, při jeho dalším připojení k napájení nedošlo k inicializaci procesoru a byl držen v resetu. K inicializaci procesoru došlo samovolně až po několika minutách od připojení napájení.

   Po důkladném prozkoumání zapojení a měření na desce bylo zjištěno, že reset procesoru způsobuje obvod převodníku USB–RS232. Kvůli možnosti programování procesoru pomocí sériového terminálu je výstupní pin CTS na převodníku spojen s resetovacím vstupem procesoru. Obvod FTDI je však podle zapojení napájen

pouze z USB sběrnice, takže v běžném provozu robotu není převodník napájen. Kvůli malému výstupnímu odporu výstupu CTS převodníku ve stavu bez napájení je procesor držen v resetu.

Problém byl odstraněn propojením napájení převodníku s napájením procesoru drátovou propojkou a přerušením vodivé cesty od napájení z USB konektoru. Díky tomu je obvod FTDI napájen vždy a ne jen při připojení USB. Po této úpravě procesor vždy spolehlivě startuje.

3. Chyba – zkrat při stisku resetovacího tlačítka

   Další chybou na procesorovém modulu byl problém se zkratem napájecího napětí 5 V ke kterému docházelo po stisku resetovacího tlačítka.

   Ve schématu modulu je navržen resetovací obvod sestavený z dvojité diody BAV99 a obvodu MCP120, jehož účelem je resetovat procesor, pokud napájecí napětí procesoru je menší než 4,85 V. Tyto součástky ovšem na desce nebyly osazeny a problém se zkratem se projevil právě až po jejich osazení.

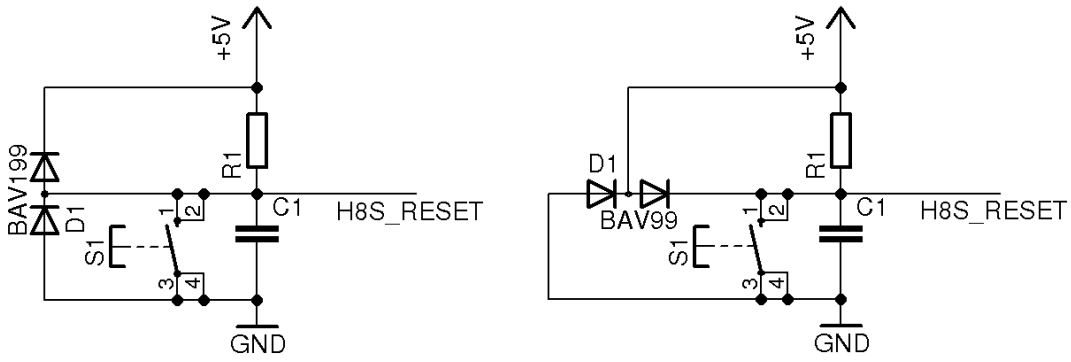
   Po proměření osazených součástek a spojů desky bylo zjištěno, že skutečné propojení plošného spoje neodpovídá schématu a pouzdro SOT–23 diody BAV99 je na DPS navrženo se špatným rozmístěním vývodů. Po stisku tlačítka tak došlo k uzavření obvodu mezi zemí a napájecím napětím přes tlačítko a jednu z diod v obvodu BAV99, viz obrázek 4.7(b).

   Problém byl odstraněn osazením pouzdra diody *vzhůru nohama* a otočeného tak, aby vývody na pouzdře byly spojeny s odpovídajícími ploškami na DPS. Po opravě je funkce resetovacího obvodu bezchybná.

   Tato závada byla také zdokumentována na wiki na stránce s popisem procesorového modulu, viz [17].

4.7 Elektronika pro zpracování odometrie

Elektronika pro zpracování signálu z IRC snímačů z odometrických kol vychází zapojením z procesorového modulu Hitachi pro řízení motorů a používá shodný procesor. Na desce odometrie pouze nejsou osazeny žádné výkonové součástky pro buzení motorů. Rozdíl oproti procesorovému modulu je v rozšíření desky o diferenciální linkové přijímače sběrnice RS422 pro zpracování signálu z IRC snímačů a součástky potřebné pro zajištění komunikace po sběrnici CAN.



(a) správné zapojení diody BAV99 ve schématu

(b) chybné zapojení diody BAV99 na desce

Obrázek 4.7: Zapojení resetovacího obvodu procesoru Hitachi

Tato deska byla navržena Martinem Rakovcem podle výrobních podkladů k procesorovému modulu.

Vzhledem k tomu, že deska odometrie vychází zapojením z procesorového modulu Hitachi, došlo k přenesení některých chyb i na tuto desku.

Jedná se o problém s nespolehlivým startem procesoru, viz bod 2 na straně 22, a chybu v návrhu špatného pouzdra, viz bod 3 na straně 23. V tomto případě se ovšem jednalo o součástku resetovacího obvodu MCP120. Obě chyby byly odstraněny shodným způsobem jako v předchozím případě a rovněž zdokumentovány na wiki.

4.8 Bezdrátový přístupový modul Wi–Fi

Pro možnost bezdrátového připojení k robotu je na robotu instalován bezdrátový Wi–Fi router. Router je propojen s řídicím počítačem pomocí ethernetového kabelu. Router vytváří zabezpečenou bezdrátovou počítačovou síť CTU_demo. Využití bezdrátové sítě usnadňuje správu celého systému a umožňuje monitorování stavu systému pomocí simulátoru *robomon*, viz sekce 6.6, bez závislosti na kabelovém propojení.

Nastavení přístupového bodu a bezdrátové sítě je popsáno v bodu 2b na straně 55.

4.9 USB zařízení

4.9.1 USB hub

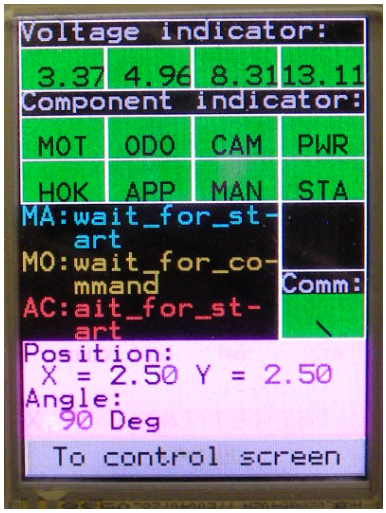
V robotu je instalován čtyřportový USB hub připojený k USB portu hlavní řídicí desky. Do USB hubu jsou připojena všechna ostatní instalovaná USB zařízení.

4.9.2 OLED displej

Pro účely diagnostiky je na horním panelu robotu instalován dotykový grafický OLED displej s rozlišením 240×320 pixelů, viz obrázek 4.8. Jedná se o inteligentní grafický modul programovatelný přes sériové rozhraní ve speciálním jazyce 4DGL.

Displej slouží pro zobrazení stavu všech elektronických systémů robotu. Zobrazuje stav celé napájecí soustavy, tedy napětí akumulátoru a napětí na všech napájecích větvích PWR boardu. Dále zobrazuje stav všech systémů připojených ke sběrnici CAN a USB.

V textové oblasti displeje jsou vypisovány aktuální stavy, kterými právě prochází hlavní stavový automat aplikace.



Obrázek 4.8: OLED displej pro diagnostiku systému robotu

4.9.3 Laserový rangefinder Hokuyo

Hlavním senzorem pro snímaní okolního prostoru robotu je laserový rangefinder Hokuyo URG–04LX–UG01. Senzor umožňuje snímání ve 2D poli v polárních souřadnicích. Parametry senzoru jsou uvedeny v tabulce 4.1.

Rangefinder je instalován na přední části robotu ve směru jízdy a je umístěn přibližně 3 cm nad rovinou podlahy.

Hlavním použitím senzoru je detekce všech překážek v okolí. Podle naměřených dat jsou poté detekované překážky zakresleny do mapy, kterou robot používá pro plánování pohybu. Více o zpracování naměřených dat a tvorbě mapy je v sekci 5.3.6.4.

Parametr	**Hodnota**
úhlový rozsah	240 °
úhlové rozlišení	0, 36 °
rozsah měření vzdálenosti	20–5600 mm
frekvence měření	10Hz
přesnost měření	60–1000 mm: ± 30 mm
přesnost měření	1000–4095 mm: ± 3% z měření

Tabulka 4.1: Parametry laserového rangefinderu Hokuyo URG–04LX–UG01, uvedená přesnost platí pro bílý list papíru, zdroj: Hokuyo [19]

Dále je rangefinder použit pro detekci cílového objektu v okolním prostoru. Problematika detekce cílového objektu je popsána v sekci 5.3.6.5.

Poslední využití rangefinderu je pro přesné najetí robotu k cílovému objektu po jeho detekci a rozpoznání pomocí kamery.

4.9.4 Kamera

Pro rozpoznávání cílového objektu je robot vybaven kamerou. Jedná se o obyčejnou webovou kameru Microsoft s rozlišením 0,3 Mpx.

Algoritmus pro rozpoznávání cíle je popsán v sekci 5.3.5.

4.10 Kabeláž

Důležitou součástí elektronického systému robotu je kabeláž propojující jednotlivé komponenty. Častou příčinou poruch robotu v minulých letech při soutěžích byla právě nekvalitně vytvořená kabeláž a upevnění konektorů. Při vytváření a ukládání nové kabeláže a zakončování kabelů pomocí konektorů byl kladen velký důraz na kvalitu provedení práce a spolehlivost výsledného elektrického propojení.

Kapitola 5

Software

Tato kapitola se věnuje popisu částí software, použitých na demonstračním robotu. V úvodu je stručně popsán operační systém spuštěný na hlavním řídicím počítači a dále je představena komunikační vrstva ORTE, použitá pro předávání dat mezi všemi ostatními aplikacemi spuštěnými v operačním systému. Tyto aplikace jsou zde také podrobněji popsány. Blokové schéma popisující strukturu software a komunikaci mezi moduly je na obrázku 5.2.

V závěrečné části této kapitoly je popsán simulátor *robomon* a úprava firmware pro desku Driver board.

5.1 Operační systém

Na hlavním řídicím počítači robotu – procesorovém modulu MIDAM, je nainstalován operační systém Linux.

Jedná se o velmi jednoduchou distribuci vytvořenou speciálně pro účely testování řízení procesů v reálném čase a testování komunikačních technologií vyvinutých na katedře řídicí techniky.

Základem distribuce je linuxové jádro `Vanilla v2.6.28.8`. Jako základní sada utilit je použit `Busybox v1.12.1`, minimalistická verze základního linuxového uživatelského prostředí určená především pro embedded aplikace. Komunikaci s řídicím počítačem prostřednictvím SSH protokolu zajišťuje nainstalovaný SSH server `Dropbear v0.51`.

Detailní dokumentace popisující kompilaci kernelu, aplikaci patchů a celkovou správu procesorového modulu je k dispozici na stránce HW wiki katedry řídicí techniky [20].

5.2 ORTE - Open Real Time Ethernet

Software ORTE (*Open Real Time Ethernet*) [21] je součástí projektu OCERA (*Open Components for Embedded Real-time Applications*) [22]. Obrázek 5.1 znázorňuje začlenění vrstvy ORTE a obecný komunikační model ORTE.

Citace a překlad z anglické dokumentace, [21]:

> ORTE je open-source implementace RTPS (*Real-Time Publish-Subscribe*) komunikačního protokolu. RTPS je protokol aplikační vrstvy a má dva hlavní komunikační módy. Jedním je publish-subscribe protokol pro přenos procesních dat od zdrojů k příjemcům. Druhý je *Composite State Transfer* (CST) protokol, který přenáší stav. RTPS protokol byl navržen pro použití se síťovým protokolem IP/UDP.
>
> Architektura publish-subscribe byla navržena pro zjednodušení distribuce dat od jednoho zdroje k více příjemcům. Zdroj dat nemusí mít žádnou znalost o počtu příjemců nebo jejich umístění. Zároveň příjemci přijímají data anonymně a nepotřebují žádnou informaci o zdroji dat. Každá aplikace může být současně zdrojem a příjemcem dat.
>
> Architektura publish-subscribe je velmi vhodná pro distribuované aplikace. Je snadno rozšiřitelná a tok dat lze snadno kontrolovat nezávisle na počtu bodů (zdrojů/příjemců) zapojených do systému.

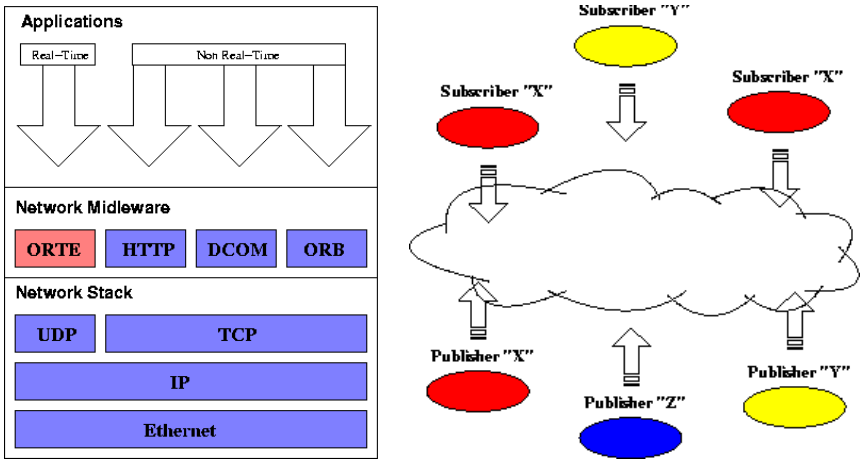
Komunikační vrstva ORTE je použita pro předávaní dat mezi všemi aplikacemi obsluhujícími různé části systému robotu, viz obrázek 5.2. Dále je ORTE použito pro předávání dat prostřednictvím bezdrátové sítě mezi aplikacemi spuštěnými na robotu a aplikacemi spuštěnými na host PC.

5.3 Aplikace spuštěné na řídicím počítači

Na řídicím počítači je spuštěno několik aplikací, jejichž vzájemnou spoluprací prostřednictvím vrstvy ORTE je zajištěna celková funkce demonstrační aplikace, viz obrázek 5.2.

5.3.1 ortemanager

Aplikace *ortemanager*, která je součástí software ORTE, slouží aplikaci k prvotnímu získání informací o ostatních aplikacích komunikujících prostřednictvím ORTE



(a) začlenění vrstvy ORTE (b) publisher-subscriber model

Obrázek 5.1: Začlenění komunikační vrstvy ORTE a obecný publish-subscribe model, zdroj: OCERA [22]

a musí být spuštěna na každém bodu komunikační sítě, viz obrázek 5.3. Po registraci u *ortemanageru* už aplikace komunikují přímo mezi sebou a služeb *ortemanageru* už nevyužívají. Tato aplikace nebyla nijak upravena.

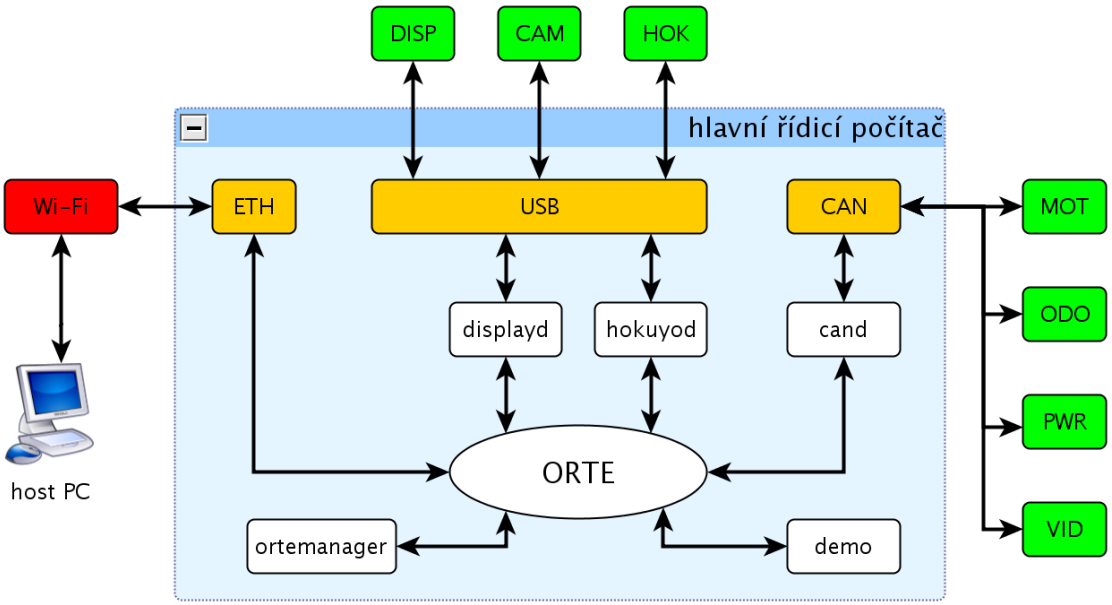
5.3.2 cand

Aplikace *cand* slouží jako bridge mezi sběrnicí CAN a vrstvou ORTE. Aplikace odchytává všechny zprávy, které přicházejí na CAN interface řídicího počítače a publikuje tyto zprávy do ORTE. Naopak z ORTE přijímá pouze ty zprávy, které jsou určeny pro vyslání na CAN sběrnici. Tyto zprávy zformátuje podle obsahu, připojí ke zprávě ID podle datového obsahu a odešle zprávy na CAN interface. Blokově je tato část systému zobrazena na obrázku 5.4.

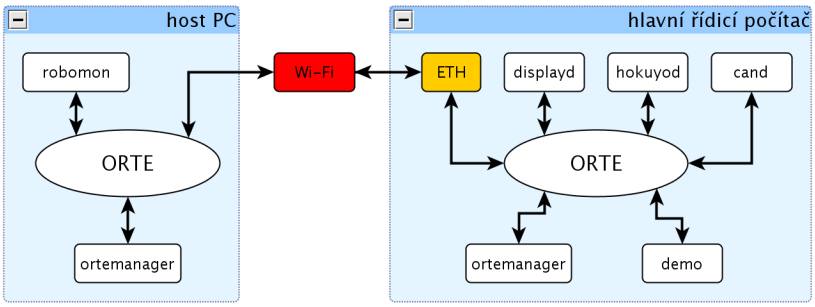
Tato aplikace byla již dříve vytvořena ostatními členy týmu. Pro potřeby demonstračního robotu byly upraveny některé části zdrojového kódu, týkající se zpracování nového typu zpráv předávaných po CAN sběrnici a publikovaných do vrstvy ORTE, které jsou specifické pro demonstrační aplikaci.

5.3.3 hokuyod

Aplikace *hokuyod* je určena ke zpracování dat z rangefinderu, který je připojen do USB hubu. Aplikace komunikuje s rangefinderem po vyhrazeném virtuálním sériovém portu ttyACM0 rychlostí 115200 baudů. Každý celý odměr, složený z 681 vzorků, je poté vložen do komunikačního objektu a publikován do vrstvy ORTE. Blokové schéma



Obrázek 5.2: Blokové komunikační schéma vrstvy ORTE v rámci všech systémů robotu



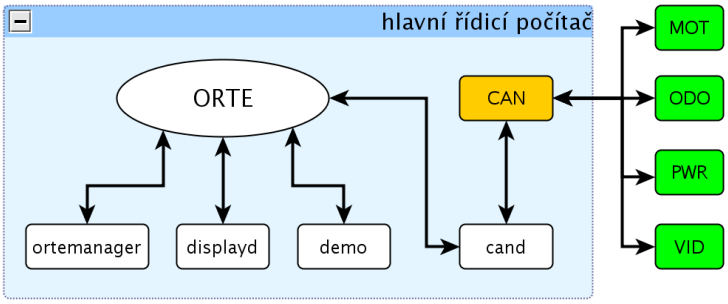
Obrázek 5.3: Blokové komunikační schéma vrstvy ORTE v rámci dvou síťových bodů

znázorňující tuto část systému je na obrázku 5.5.

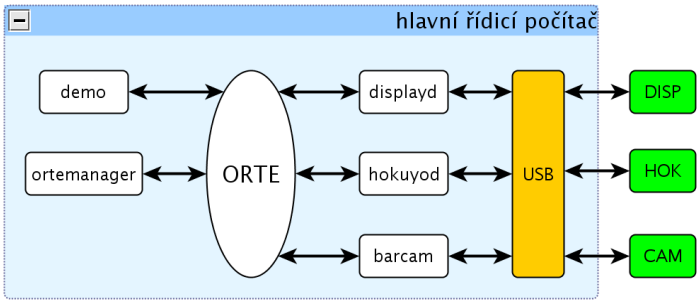
Aplikace *hokuyod* byla již dříve vytvořena jinými členy týmu a nebyla nijak upravena.

5.3.4 displayd

Aplikace *displayd* slouží k příjmu a zpracování zpráv publikovaných do vrstvy ORTE všemi ostatními systémy robotu. Tato aplikace komunikuje pomocí virtuálního sériového portu s grafickým displejem, který je připojen do USB hubu. Podle obsahu zpráv přijímaných z vrstvy ORTE poté aplikace posílá příkazy do grafického displeje pro zobrazení stavu jednotlivých systémů. Blokové schéma této části systému je na obrázku 5.5.



Obrázek 5.4: Aplikace *cand* slouží jako bridge mezi CAN interface a vrstvou ORTE



Obrázek 5.5: Aplikace propojující USB zařízení s vrstvou ORTE

Tato aplikace byla již dříve vytvořena ostatními členy týmu. Pro potřeby demonstračního robotu byly upraveny části zdrojového kódu, zpracovávající nové typy zpráv specifických pro demonstrační aplikaci. Také byl upraven grafický program uložený v modulu displeje tak, aby na displeji byly zobrazeny pouze stavy těch systémů, které jsou skutečně instalovány v demonstračním robotu.

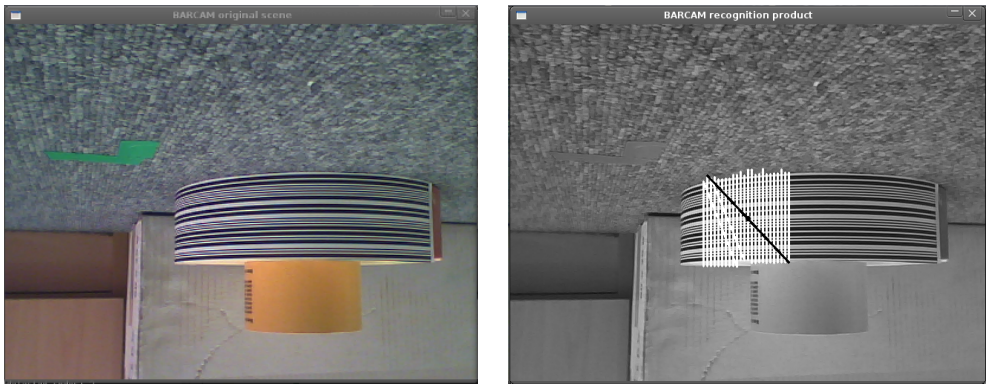
5.3.5 barcam

Aplikace *barcam* je program určený pro zpracování obrazu z kamery a rozpoznání čárového kódu umístěného na cílovém objektu v obraze. Aplikace je použita pro rozhodnutí zda se jedná o platný cíl v okamžiku, kdy robot stojí ve vzdálenosti přibližně 1 m od potenciálního cíle. Výsledek rozpoznávání je poté publikován do vrstvy ORTE.

Kostra této aplikace je totožná s aplikací *rozkuk*, která byla použita při soutěži v roce 2010 pro rozpoznávání konfigurace herních prvků. Autorem aplikace *rozkuk* je Petr Kubizňák, který tuto aplikaci vytvořil v rámci své bakalářské práce [6].

Obecně může aplikace *barcam* fungovat následujícím dvojím způsobem.

1. Pokud je zkompilovaná pro host PC, obsahuje jednoduché grafické rozhraní s okny,



(a) Originální scéna z kamery (b) Produkt rozpoznávání

Obrázek 5.6: Okna aplikace *barcam* s originálním obrazem a produktem rozpoznávání

která zobrazují originální obraz z kamery a produkt rozpoznávání, viz obrázek 5.6. Toto použití je určeno pro vývoj a testování aplikace. V tomto případě má aplikace následující módy.

mode video Aplikace po spuštění na PC ihned přejde do módu video a zobrazí se okno s obrazem z kamery, viz obrázek 5.6(a). Pomocí klávesy **S** lze uložit aktuální obraz. Klávesou **O** se přejde do režimu nastavení ROI – *region of interest* a pomocí číselných kláves lze nastavit oblast zájmu ve které se bude provádět rozpoznávání, viz obrázek 5.7. Režim nastavení ROI je opuštěn po stisku klávesy ESC a v okně s originální scénou je zobrazen pouze vybraný výřez. Režim nastavení ROI nebyl součástí výchozího programu *rozkuk* a do programu *barcam* byl přidán a naprogramován pro testovací účely.

mode wait Do wait módu lze přejít ze všech ostatních módů klávesou **P**. Ve wait módu aplikace nezískává obraz z kamery a čeká na stisk klávesy pro přechod do jiného módu. Wait mód lze opustit opět klávesou **P**.

mode recognize Stiskem klávesy **R** aplikace přejde do módu rozpoznávání. Zobrazí se druhé okno s monochromatickým obrazem a výsledky rozpoznávání, viz obrázek 5.6(b). Samotný proces zpracování obrazu a rozpoznávání je popsán v sekci 5.3.5.1.

2. Pokud je aplikace zkompilovaná pro řídicí počítač robotu neobsahuje žádné grafické rozhraní.

mode wait Po spuštění na řídicím počítači robotu aplikace přejde do módu wait a čeká na příkazy a hlavní řídicí aplikace *demo*.

mode recognize Pokud aplikace prostřednictvím vrstvy ORTE přijme příkaz ke spuštění procesu rozpoznávání, přejde do módu recognize, provede rozpoznávání a odešle výsledek rozpoznávání zpět do vrstvy ORTE.

Obrázek 5.7: Nastavení oblasti zájmu – ROI v obraze kamery

5.3.5.1 Rozpoznávání čárového kódu v obraze

Pro potřeby demonstrační aplikace musel být mód rozpoznávání oproti původní aplikaci *rozkuk* zásadně upraven.

Vlastní algoritmus pro nalezení čárového kódu v obraze byl převzat z programu *barcol* jehož autorem je Ondřej Holešovský z robotického týmu Gymšpit [25].

Popis algoritmu pro nalezení čárového kódu v obraze, upravená citace autora programu *barcol*.

1. Získáme obraz z kamery a provedeme konverzi z barevného obrazu na černobílý.
2. Najdeme v obraze sloupce pixelů, které splňují požadované parametry – větší než N počet ostřejších barevných přechodů, alespoň jeden velmi tmavý pixel a jeden velmi světlý.
3. Z nalezených sloupců pixelů vybereme ty, které tvoří obdélníky o délce L.
4. Sloučíme obdélníky, které jsou blízko sebe.
5. Najdeme střed každého obdélníku a určíme největší obdélník.
6. Polohu středu největšího obdélníku v obraze přepočítáme na úhel mezi přímou osou kamery a přímkou mířící ke středu nalezeného čárového kódu.

Naprogramovaný proces rozpoznávání v programu *barcam* je tedy složený ze základní kostry aplikace *rozkuk* a algoritmu pro hledání čárových kódů v obraze převzatého z programu *barcol*.

Rozpoznávání v aplikaci *barcam* probíhá v následujících krocích.

1. Aplikace čeká na příkaz z klávesnice (host PC) nebo z vrstvy ORTE (robot) a po jeho přijetí přejde do bodu 2.

2. Aplikace rozpoznává. Do výše popsaného algoritmu je předán monochromatický obraz a je proveden jeden průchod rozpoznávacího algoritmu. Do návratové hodnoty algoritmu se připočítává každý detekovaný obdélník, ne pouze ten největší.

3. Rozpoznávání podle bodu 2 se provede na pěti různých snímcích z kamery a návratové hodnoty ze všech pěti průchodů se sečtou. Výsledná návratová hodnota je poté publikována do ORTE jako hodnota true nebo false. Pokud je tedy aspoň v jednom z pěti obrazů nalezen aspoň jeden obdélník reprezentující sloupec v čárovém kódu, je cíl považován za platný.

4. Po skončení rozpoznávání aplikace přejde do bodu 1.

5.3.6 demo

Aplikace *demo* je hlavní řídicí aplikace demonstračního robotu a skládá se z několika stavových automatů, (dále jen FSM–*Finite State Machine*) a modulů řešících specifické úkoly, přičemž některé úkoly jsou prováděny v samostatných vláknech. V této sekci následuje popis automatů a modulů, které byly pro realizaci demonstrační aplikace nově vytvořeny či upraveny.

Základní struktura aplikace a rozdělení do vláken je následující.

1. vlákno

   `MAIN FSM` Hlavní automat demonstrační aplikace, dále jen *main fsm* .
   Popis návrhu tohoto automatu je v sekci 5.3.6.1.

   `MOTION FSM` Automat pro řízení pohybu robotu, dále jen *motion fsm* .
   Popis změn provedených v tomto automatu je v sekci 5.3.6.2.

   `robot_orte` Modul pro komunikaci prostřednictvím vrstvy ORTE,
   dále jen *robot_orte* .
   Popis změn provedených v tomto modulu je v sekci 5.3.6.3.

map_handling Modul pro zpracování dat z rangefinderu, publikovaných do ORTE aplikací *hokuyod*, a zakreslování překážek do sdílené mapy, dále jen *map_handling*. Popis změn provedených v tomto modulu je v sekci 5.3.6.4.

2. vlákno

forget_obstacles() Vlákno pro procházení sdílené mapy a postupné zapomínání detekovaných překážek.

3. vlákno

trajectory_folower() Vlákno regulátoru pro sledování naplánované trajektorie.

5.3.6.1 Hlavní stavový automat

Pro řešení demonstračního úkolu byl naprogramován zcela nový *main fsm*. Naprogramovaný automat byl realizován pomocí knihovny FSM, podle informací uvedených v bakalářské práci Filipa Jareše [2]. V jeho práci je také uveden obecný popis knihovny FSM, význam různých specifických termínů používaných pro popis signálů a událostí používaných pro přechod mezi stavovými funkcemi. Vývojový diagram vytvořeného automatu *main fsm* je v příloze A.1. Funkce automatu *main fsm* je realizována následujícími stavy.

1. `FSM_STATE(wait_for_start)`

   (a) Po spuštění aplikace *demo* se provede inicializace všech systémů robotu a automat čeká na příchod události `EV_START` od modulu *robot_orte*.

   (b) Tato událost je poslána v okamžiku, kdy modul *robot_orte* obdrží zprávu z desky Eb board, informující o stisku startovacího tlačítka.

   (c) Na základě této události *main fsm* přejde do stavu 2.

2. `FSM_STATE(survey)`

   (a) Provede se detekce cílů v datech naměřených rangefinderem a souřadnice detekovaných cílů jsou uloženy do vektoru. Algoritmus detekce cíle je stručně popsán v bodu 1 v sekci 5.3.6.5.

   (b) Pokud je z aktuální pozice detekován alespoň jeden cíl, *main fsm* přejde do stavu 3.

(c) Pokud není detekován žádný cíl, robot se otočí na místě o 120 ° a přechází na začátek tohoto stavu. Takto se otočí maximálně 2× a pokud z této pozice nedetekuje žádný cíl, přejde do stavu 4.

3. `FSM_STATE(approach_target)`

Vektor se souřadnicemi detekovaných cílů se postupně prochází a provádí se následující body.

(a) Pokud vektor neobsahuje žádný neprozkoumaný cíl, automat přejde do stavu 4.

(b) Pokud je ve vektoru se souřadnicemi detekovaných cílů k dispozici další cíl, vypočítá se pozice pro přiblížení k dalšímu cíli a robot přejede na vypočtenou pozici. Algoritmus určení pozice pro přiblížení k cíli je stručně popsán v bodu 2 v sekci 5.3.6.5.

Po dokončení pohybu *main fsm* přejde do subautomatu a provádí následující body.

i. `FSM_STATE(recognize)`
Subautomat odešle prostřednictvím modulu *robot_orte* signál do aplikace *barcam* pro spuštění rozpoznávání.
Poté čeká na událost `EV_CAMERA_DONE`, že bylo rozpoznávání dokončeno. Pokud je podle rozpoznávání cíl považován za platný, subautomat přejde do stavu 3(b)ii. Pokud cíl není platný, nebo se čeká na výsledek rozpoznávání déle než 10 s, dojde k návratu ze subautomatu na začátek stavu 3.

ii. `FSM_STATE(get_target_turn)`
Robot stojí blízko cíle a z dat rangefinderu se určí minimální naměřená vzdálenost v rozsahu ±45 ° a vypočte se úhel mezi dopřednou osou robotu a spojnicí mezi středem rangefinderu a naměřeným minimem.
V tento okamžik se vypne detekce překážek pomocí rangefinderu, aby byl umožněn pohyb robotu i v těsné blízkosti cíle.
Robot se otočí o vypočtený úhel a pokud vzdálenost k cíli je menší než 5 cm, subautomat pokračuje stavem 3(b)iv, jinak pokračuje stavem 3(b)iii.

iii. `FSM_STATE(get_target_touch)`
Robot popojede 2 cm vpřed směrem k minimu a vrací se do stavu 3(b)ii.

iv. `FSM_STATE(get_target_load)`
Robot stojí těsně u cíle. Prostřednictvím modulu *robot_orte* odešle příkaz do desky Eb board k naložení nákladu pomocí manipulátoru. Po dokončení pohybu manipulátoru a naložení nákladu subautomat přijme událost `EV_CRANE_DONE` a přejde do stavu 3(b)v.

v. `FSM_STATE(get_target_back)`
Robot popojede pozpátku 30 cm směrem od cíle tak, aby měl dostatek prostoru pro otočení. Opět se zapne detekce překážek pomocí range-finderu a dojde k návratu ze subautomatu a přechodu do stavu 5.

4. `FSM_STATE(move_around)`

   Je vygenerován náhodný bod v rámci povoleného prostoru pohybu robotu a robot jede na tento bod. Po dosažení nové pozice přejde do stavu 2.

5. `FSM_STATE(go_home)`

   Robot s naloženým nákladem jede na pozici, kde došlo k odstartování aplikace, po dosažení startovní pozice aplikace skončí.

5.3.6.2 Stavový automat řízení pohybu

Tento automat přijímá příkazy z *main fsm* a zajišťuje plánování trajektorie a pohyb robotu do zadané pozice. Stavový automat pro řízení pohybu robotu byl naprogramován v předchozích letech jinými členy týmu, ale pro účely demonstrační aplikace byly provedeny úpravy v následujících stavech *motion fsm*.

1. `FSM_STATE(movement)`

   Tento stav byl upraven tak, že pokud přijme 5× po sobě událost `EV_OBSTACLE`, odešle se signál `FSM_SIGNAL(MAIN, EV_MOTION_ERROR, NULL)` do *main fsm* a poté *motion fsm* přejde do stavu `FSM_STATE(wait_for_command)`.

   Tato úprava omezuje maximální počet pokusů pro přeplánování trajektorie do požadované pozice pokud je přístup k této pozici blokován překážkou. Tím se zabrání tomu, že robot se bude nekonečně dlouho snažit dojet do bodu, do kterého lze sice naplánovat trajektorii, ale kolem kterého není dostatek prostoru pro manévrování.

2. `FSM_STATE(wait_and_try_again)`

Tento stav byl upraven tak, že pokud plánovací algoritmus 3× po sobě vrátí parametr `TARGET_INACC`, *motion fsm* odešle signál `FSM_SIGNAL(MAIN, EV_MOTION_ERROR, NULL)` do *main fsm* a poté *motion fsm* přejde do stavu `FSM_STATE(wait_for_command)`.

Tato úprava zabraňuje trvalému zastavení a čekání robotu v případě, kdy dojde k naplánování trajektorie do bodu, který je překážka, ale v době plánování jako překážka označen nebyl. Například proto, že cílový bod byl mimo dosah range-finderu, nebo na tento bod byla překážka umístěna až po započatí pohybu po trajektorii.

5.3.6.3 Modul komunikace s vrstvou ORTE

Tento modul slouží k přijímání zpráv ze všech systémů robotu prostřednictvím vrstvy ORTE a obsah těchto zpráv pak předává do různých částí hlavní řídicí aplikace *demo*. Zároveň tento modul publikuje příkazy z *main fsm* a *motion fsm* pro řízení pohonných motorů a všech dalších aktuátorů robotu.

Modul *robot_orte* byl vytvořen již dříve ostatními členy týmu. Pro potřeby demonstrační aplikace byla do modulu implementována funkce pro kontrolu stavu všech systému robotu.

Pokud nějaký systém není plně připraven k odstartování demonstrační aplikace, nedovolí se odeslání signálu `FSM_SIGNAL(MAIN, EV_START, NULL)` do *main fsm* a odstartování *demo* aplikace.

5.3.6.4 Zpracování sdílené mapy

Mapa používaná pro plánování pohybu v prostoru je uložena ve paměti, která může být sdílena vice procesy, například aplikacemi *demo* a *robomon*.

Vytvoření mapy

Pro plánování pohybu v prostoru robot používá mapu, která odpovídá velikosti povoleného prostoru ve kterém se robot může pohybovat, tedy 5×5 m. Mapa je rozdělena na buňky o velikosti 10×10 cm, tím je určeno rozlišení s jakým jsou do mapy zakreslovány překážky. Každá buňka mapy je struktura obsahující dva parametry o velikosti 4 bytů, parametr `flags` a `detected_obstacle`.

Podle počtu buněk potřebných pro pokrytí celé mapy je poté v RAM paměti vytvořena sdílená paměť. Pro každou buňku mapy je tedy ve sdílené paměti vyhrazeno 8 bajtů. Při vytvoření sdílené paměti je uveden přístupový klíč k této paměti. Každá

další aplikace spuštěná na stejném stroji, která má k dipozici klíč k této sdílené paměti tak získává přístup k mapě.

Této vlastnosti je využíváno při vývoji a testování hlavní řídicí aplikace spuštěné na host PC, kdy je možné zobrazit mapu prostředí v aplikaci *robomon*, která je spuštěna na stejném stroj, viz obrázek 6.6.

Kreslení překážek do mapy

Pro zakreslování překážek do mapy jsou využita data z rangefinderu. V okamžiku, kdy modul *robot_orte* přijme z ORTE nová data z rangefinderu, zavolá funkci `update_map_hokuyo()` v modulu `map_handling` a předá jí naměřená data. Naměřené hodnoty jsou poté transformovány z relativních polárních souřadnic vůči středu rangefinderu na absolutní pozici v prostoru vůči počátku souřadnic hřiště, viz obrázek 5.8.

Pro každou transformovanou hodnotu se poté zjistí, jaké buňce v mapě odpovídá a této buňce jsou nastaveny odpovídající parametry. Pomocí parametru `flags` se nastavuje, že je na této pozici detekována překážka a zároveň se nastaví parametr `detected_obstacle`, který funguje jako čítač. Vlákno `forget_obstacle()` prochází všechny buňky v mapě a podle nastavených parametrů jako je perioda zapomínání a doba do úplného zapomenutí překážky postupně dekrementuje čítače detekovaných překážek.

Celý proces vytvoření sdílené mapy a aktualizace mapy podle dat z rangefinderu byl již dříve navržen a naprogramován ostatními členy týmu a byl přizpůsoben požadavkům soutěže Eurobot. Pro použití v demonstračním robotu musel být modul *map_handling* a princip zakreslování překážek upraven následujícím způsobem.

1. Byly nastaveny nové parametry pro vlákno zapomínající překážky v mapě. Perioda zapomínání překážek je 200 ms a maximální čas do zapomenutí překážky je 5 s. Původní hodnoty pro zapomínání byly pro použití v neznámém prostředí příliš krátké. Docházelo tak k příliš častému přeplánování trajektorie pokud se robot pokoušel objet velký objekt a došlo k zapomenutí velké části tohoto objektu dříve, než ho stačil robot objet. Nastavené parametry byly zvoleny jako dostatečný kompromis mezi příliš krátkou dobou a dobou při které by mapa byla přeplněna překážkami a robot by nemohl nikdy naplánovat trajektorii.

2. Byl změněn princip zakreslování překážek do mapy. Ve verzi pro soutěž Eurobot se vždy předpokládalo, že detekovaná překážka je robot soupeře a jeho předpokládaná velikost je válec o průměru 30 cm. Vždy když došlo k detekci překážky

v určitém bodě, byla za tento bod do mapy zakreslena kruhová překážka o průměru 30 cm.

V situaci demonstračního robotu ovšem nelze učinit žádný předpoklad o velikosti detekované překážky. Proto je do mapy zakreslena překážka vždy jen do místa kde došlo reálně k detekci a kolem tohoto bodu je do mapy zakreslena bezpečnostní oblast, ve které nesmí být naplánována trajektorie. Poloměr bezpečnostní oblasti odpovídá velikosti demonstračního robotu.

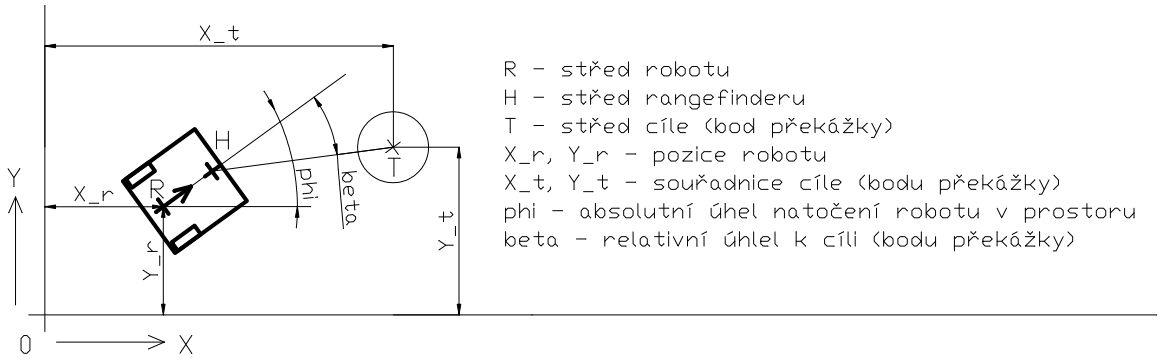
5.3.6.5 Detekce cíle pomocí rangefinderu

Pro detekci cíle pomocí rangefinderu je využita knihovna *shape_detect*, jejímž autorem je Martin Synek [18].

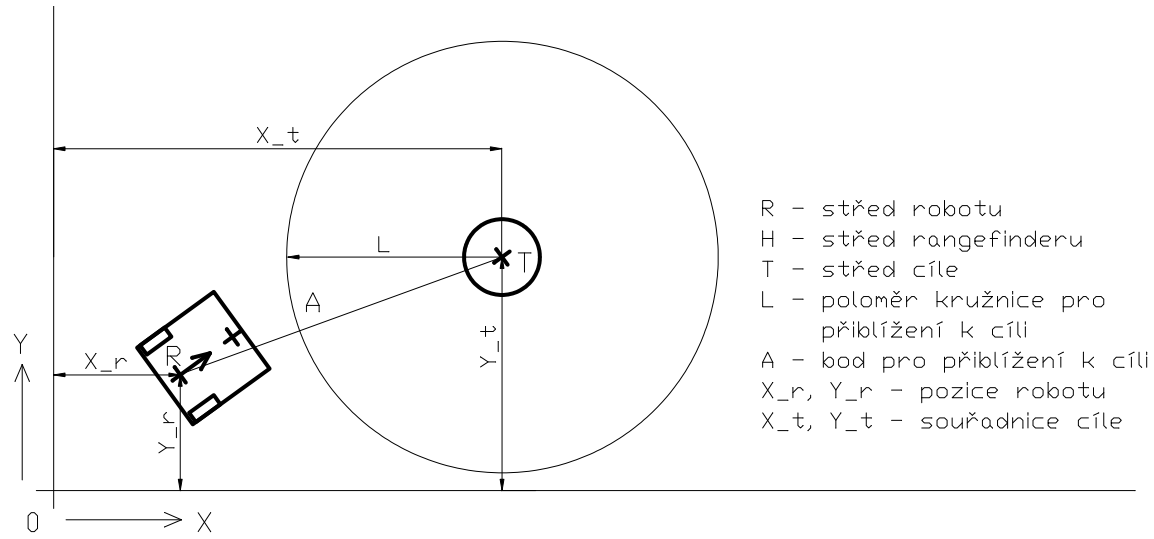
Knihovna *shape_detect* je software určený pro detekci oblouků a úseček v datech získaných pomocí rangefinderu. Vzhledem k relativně špatnému úhlovému rozlišení a malé přesnosti měření vzdálenosti rangefinderu, který je v robotu nainstalován, lze s uspokojivou spolehlivostí detekce knihovnu *shape_detect* použít pouze pro detekci oblouků o poloměru alespoň 10 cm. Proto byla velikost cílového podstavce pro náklad zvolena právě na tuto hodnotu. I v takovém případě dochází často k falešné detekci například na rozích nábytku a v hodně členitých oblastech jako jsou podstavce kancelářských židlí a podobně. Následující body popisují použití této knihovny pro nalezení cíle a určení bodu pro přiblížení k cíli.

1. Základní funkce knihovny `arc_detect(arcs)` vrací vektor obsahující souřadnice středů všech detekovaných objektů v daném scanu relativně vůči středu rangefinderu. Tato pozice je poté transformována na absolutní pozici detekovaného objektu v prostoru, viz obrázek 5.8. Vyhodnotí se, zda se detekovaný objekt nachází v rámci povoleného prostoru. Pokud ano, jsou jeho souřadnice přidány do vektoru s detekovanými cíli, jinak se ignorují.

2. Pozice na kterou robot jede při přiblížení k cíli je určena jako průsečík mezi úsečkou vedoucí z aktuální pozice robotu do středu detekovaného cíle a kružnicí se středem v cíli o poloměru L = (poloměr cíle + 3×velikost buňky mapy + poloměr otáčení robotu), viz obrázek 5.9. Takto určená vzdálenost od cíle je dostatečně velká na to, aby *motion fsm* nehlásil překážku v cestě při pohybu robotu na tento bod a zároveň je robot dostatečně blízko pro prozkoumání cíle pomocí kamery.



Obrázek 5.8: Vztah mezi souřadným systémem hřiště, robotu a rangefinderu



Obrázek 5.9: Určení pozice pro přiblížení k cíli

5.4 Simulátor robomon

Simulátor *robomon* je grafická aplikace využívající knihovny Qt a slouží především k testování navrhovaných strategií a stavových automatů pro soutěž Eurobot bez potřeby spouštět testovanou aplikaci na reálném robotu. Testovanou aplikaci lze spustit přímo na PC programátora a v simulátoru sledovat průchod aplikace jednotlivými stavy a s tím spojený pohyb robotu v simulovaném prostředí. Simulátor má pomocí klíče přístup ke sdílené paměti s mapou a může podle parametrů nastavených v mapě vykreslit detekované překážky a naplánovanou trajektorii. V simulátoru je také možné umístit do prostoru hřiště simulovanou překážku a simulovat měření rangefinderu. Takto lze například testovat nastavení detekce překážek a algoritmus plánování trajektorie.

Protože mapa je uložena v RAM paměti na stroji kde je spuštěna testovaná aplikace, nelze mapu zobrazit v případě, kdy je aplikace spuštěna na reálném robotu a simulátor je spuštěn na PC.

Základ simulátoru *robomon* byl naprogramován v minulých letech ostatními členy týmu a každý rok je upravován podle aktuálních požadavků a pravidel soutěže Eurobot.

Pro potřeby testování demonstrační aplikace byl v simulátoru upraven způsob zobrazování sdílené mapy a později byla tato úprava také převzata do výchozí verze simulátoru pro soutěž Eurobot.

V simulátoru *robomon* byla provedena následující úprava.

Původní princip vykreslování mapy v simulátoru byl ten, že se podle počtu buněk v mapě do obrazovky simulátoru nakreslil odpovídající počet čtverců. Poté se postupně procházela sdílená paměť a jednotlivým čtvercům byla nastavena barva podle toho jaký příznak byl na daném políčku mapy nastaven. Slabým článkem v tomto principu bylo vykreslování mnoha čtverců pokaždé, kdy měla být mapa na obrazovce aktualizována novými daty z rangefinderu.

V případě hřiště o velikosti 3×2 m, jaké se používá v soutěži Eurobot je počet buněk mapy ještě dostatečně malý a tomu odpovídající rychlost překreslování okna simulátoru snesitelná.

V případě demonstrační aplikace je nastavená velikost hřiště o mnoho větší (5×5 m). Množství buněk tak stoupá kvadraticky a s tím i klesá rychlost překreslování okna simulátoru. Pro takovou velikost hřiště byla rychlost překreslování okna nedostačující a byl navržen nový princip vykreslování mapy pomocí bitmapového obrazu.

Nejprve je vytvořena prázdná bitmapa o velikosti X $\times$ Y pixelů tak, aby počet pixelů odpovídal velikosti mapy. Poté se prochází sdílená paměť a jednotlivým pixelům v bitmapě je nastavena RGBA barva podle příznaku v buňce. Po vyplnění všech pixelů je celá bitmapa zobrazena najednou. Tím je výrazně snížen počet operací kreslení na obrazovku a rychlost překreslování okna simulátoru je velmi dobrá.

5.5 Úprava firmware pro řízení BLDC motorů Driver board

Firmware v procesorovém modulu Hitachi byl vytvořen v minulých letech pro účely řízení motorů v robotu a autorem zdrojového kódu je Michal Sojka. Pro řízení BLDC motorů je v kódu použita knihovna PXMC [7], která je produktem firmy PiKRON [26] a jejím autorem je Ing. Pavel Píša ve spolupráci s dalšími vývojáři.

Základní popis řízení BLDC motorů s využitím PXMC

Aby se BLDC motor roztočil, musí magnetické pole statoru směřovat kolmo k poli rotoru. Software v PXMC slouží mimo jiné k tomu, aby určilo jakým směrem má pole statoru směřovat. Sekvence rozjezdu motoru funguje s použitím HALL senzorů následovně (Michal Sojka, 2011).

1. Přečte se natočení rotoru podle HALL senzorů a aplikuje se pole kolmé k tomuto údaji (rozlišení HALL senzorů v motoru je 60 °).

2. V okamžiku, kdy se motor pootočí tak, že se změní signál z HALL senzoru, zpřesní se informace o natočení rotoru a pole statoru.

   Přečte se aktuální hodnota IRC snímače motoru, zapamatuje se jako offset a pole statoru se začne generovat na základě signálu z IRC snímače a uloženého offsetu. Tento stav je v software signalizován příznakem PTI.

3. V okamžiku, kdy se motor pootočí tak, že software dostane signál z indexové značky IRC snímače, upřesní se hodnota offsetu na základě znalosti pozice indexové značky. To je signalizováno příznakem PHA.

Úprava firmware

Protože v pohonných motorech robotu došlo ke zničení HALL senzorů, viz sekce 3.3.1, musel být firmware pro řízení motorů upraven tak, aby bylo možné rozjet motory a regulovat polohu i bez HALL senzorů. Bez signálu z HALL senzoru by nefungoval krok 1 a nikdy bychom se nedostali přes krok 2.

Ve spolupráci s Pavlem Píšou a Michalem Sojkou byla provedena následující úprava firmware a definovány podmínky, za jakých dojde k bezpečnému sfázování motoru s IRC snímačem.

1. Přívod energie do výkonového můstku buzení motorů musí být před zapnutím robotu přerušen stop tlačítkem. Tím je také umožněn volný pohyb kol robotu, která nejsou brzděna zkratovaným vinutím statoru.

2. Po inicializaci procesoru je nutné popojet robotem tak, aby se rotory obou motorů otočily alespoň o jednu otáčku.

3. V okamžiku, kdy se motory pootočí tak, že software dostane signál z indexové značky IRC snímače, nastaví se hodnota offsetu na základě znalosti pozice indexové značky. To se signalizováno příznakem PHA a zároveň je nastaven příznak PTI.

Firmware byl dále upraven tak, že dokud nedojde ke správnému sfázování motoru s IRC snímačem, ve stavové zprávě posílané po sběrnici CAN je odeslán příznak informující o chybě sfázování. Na základě tohoto příznaku je pole na displeji signalizující stav desky Driver board rozsvíceno oranžovou barvou.

V okamžiku, kdy dojde ke sfázování motoru s IRC snímačem, je příznak zrušen a signalizace stavu Driver board svítí zeleně.

Kapitola 6

Manuál ovládání demonstračního robotu

Tato kapitola popisuje návod k obsluze demonstračního robotu. Obsluhou se rozumí následující operace.

1. Příprava prostoru, ve kterém bude demonstrace probíhat, sekce 6.1.
2. Zapnutí robotu, sekce 6.2.
3. Kontrola stavu elektronických systémů, sekce 6.3.
4. Odstartování aplikace, sekce 6.4.

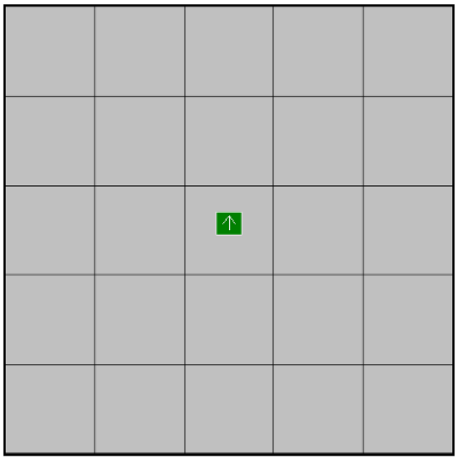
6.1 Příprava před spuštěním

1. **Výběr prostoru pro demonstraci**

   K provozu demonstrační aplikace postačuje podlaha jakékoliv místnosti. Podlaha musí být hladká a bez větších nerovností. Nejvhodnějším povrchem je PVC nebo koberec s krátkým vlasem. Není příliš vhodná dlažba, na které dochází k zasekávání předních podpěrných kuliček ve spárách mezi dlaždicemi.

2. **Umístění robotu**

   Akční rádius robotu je přibližně 2,5 m. Rozměr volného prostoru pro pohyb robotu by tedy měl mít velikost přibližně 5×5 m. Po zapnutí aplikace je lokalizační systém robotu nastaven tak, že je robot umístěn ve středu tohoto prostoru, viz obrázek 6.1.



Obrázek 6.1: Umístění robotu v prostoru

Obrázek 6.2: Cílový podstavec

Obrázek 6.3: Náklad

3. **Umístění cíle**

Pro předvedení demonstrační aplikace umístěte v mezích akčního rádiusu robotu cílový objekt. Pro účely demonstrační aplikace je cílem kruhový podstavec o průměru 200 mm a výšce 50 mm, viz obrázek 6.2.

Na podstavec umístěte náklad, který robot vyzvedne. Nákladem je například plechovka nebo jiný kovový magnetický válec o průměru maximálně 80 mm a výšce maximálně 100 mm, obrázek 6.3.

Podstavec i náklad *musí* být po celém obvodu označen čárovým kódem, podle kterého robot pomocí kamery rozhoduje, zda se jedná o hledaný cíl. Čárový kód je k dispozici v příloze B, verze vhodná pro tisk je na CD v adresáři `/barcode`.

6.2 Zapnutí robotu

1. **Akumulátor**

   Před zapnutím robotu je nutné připojit napájecí akumulátor. Ten se vkládá do prostoru za dvířky na levé straně robotu, viz obrázek 6.4(a). Připojte akumulátor k volnému kabelu s protikusem konektoru jaký je na přívodním kabelu akumulátoru. Konektor neumožňuje přepólování kontaktů. Napětí připojeného akumulátoru v nabitém stavu by mělo být v rozsahu 12 V–14 V.

   Nepřipojujte robot na větší napájecí napětí než 18 V DC!

2. **Bezpečnostní tlačítko**

   Robot je na horním panelu vybaven červeným bezpečnostním tlačítkem, viz obrázek 6.4(b), které přerušuje přívod energie do pohonných motorů. Před zapnutím napájení robotu se ujistěte, že červené bezpečnostní tlačítko je stlačené a nehrozí náhodné rozjetí robotu.

3. **Pojistka**

   Mezi bezpečnostním tlačítkem a hlavním vypínačem je umístěno pojistkové pouzdro pro automobilovou pojistku, viz obrázek 6.4(b). Před zapnutím robotu se ujistěte, že je v pouzdře vložena pojistka s tavným proudem maximálně 20 A.

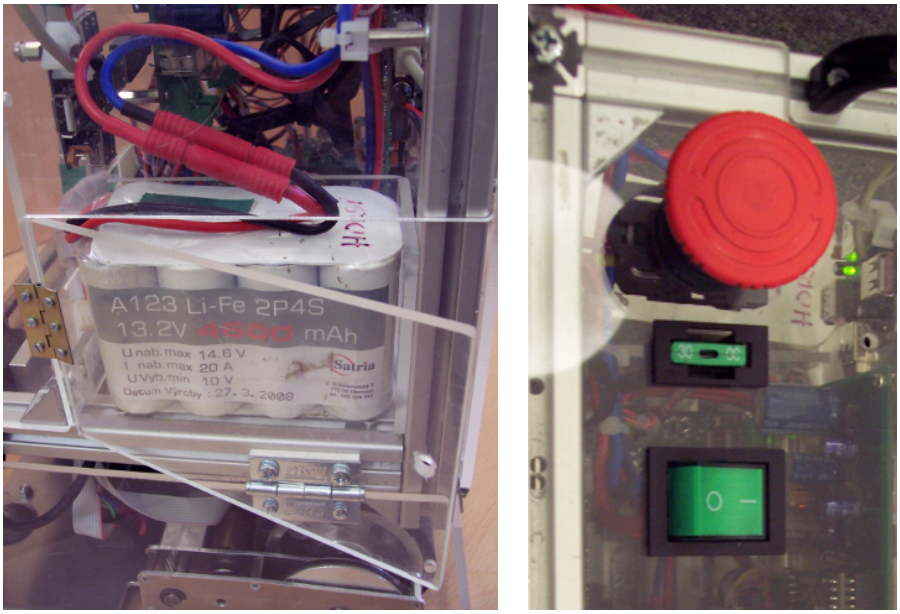
   Nevkládejte do pojistkového pouzdra pojistku s větším tavným proudem než 20 A, ani nepropojujte kontakty pomocí jiného vodiče!

4. **Hlavní vypínač napájení**

   Pro zapnutí napájení robotu se používá zelený kolébkový vypínač vedle pojistkového pouzdra, viz obrázek 6.4(b). Po kontrole bezpečnostního tlačítka a pojistky zapněte hlavní vypínač.

6.3 Kontrola stavu systémů robotu

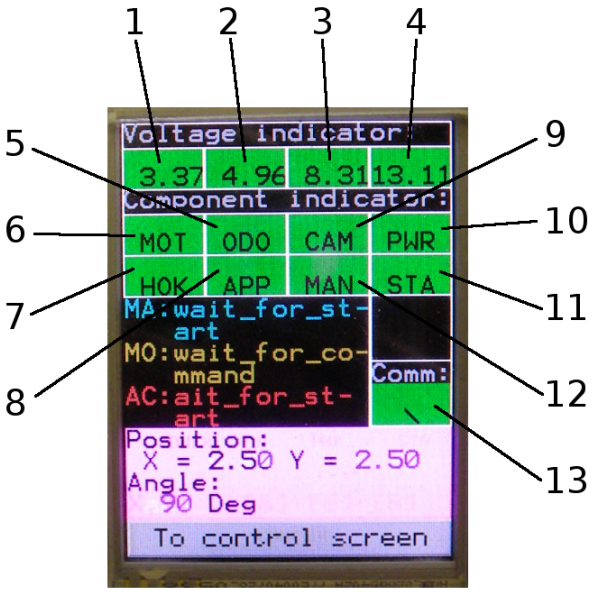
Na horním panelu robotu je umístěn grafický displej zobrazující stav všech elektronických systémů robotu, viz obrázek v tabulce 6.1. Po zapnutí hlavního vypínače vyčkejte cca 10 sekund než dojde k nastartování operačního systému hlavního řídicího počítače. Po nastartování OS zkontrolujte stav ostatních systémů robotu podle následujících bodů.



(a) Uložení akumulátoru

(b) Část ovládacího panelu s bezpečnostním tlačítkem, pojistkou a hlavním vypínačem

Obrázek 6.4: Napájení robotu



1	stav napájení 3,3 V
2	stav napájení 5 V
3	stav napájení 8 V
4	stav akumulátoru
5	stav odometrie
6	stav Driver board
7	stav appl. *hokuyod*
8	stav appl. *demo*
9	stav appl. *barcam*
10	stav PWR board
11	stav start tlačítka
12	stav Eb board
13	stav komunikace displeje

Tabulka 6.1: Popis diagnostického displeje

1. **Stav komunikace s displejem**

   Stav, kdy displej komunikuje s hlavním počítačem a správně zobrazuje údaje o stavu robotu je indikován zeleně rozsvíceným políčkem *Comm* v pravém dolním rohu displeje a otáčejícím se kurzorem uvnitř tohoto pole, viz 13 v tabulce 6.1.

2. **Stav napájení**

   Na displeji zkontrolujte hodnotu napětí akumulátoru a napětí napájecí větve 3,3 V, 5 V a 8 V ve skupině *Voltage indicator*, viz 1, 2, 3, 4 v tabulce 6.1.

   akumulátor Pokud je akumulátor dostatečně nabit a napětí akumulátoru je větší než 12,5 V, pole s napětím akumulátoru svítí zeleně. Pokud dojde k vybití akumulátoru pod úroveň 12,5 V, pole s napětím akumulátoru ze rozsvítí oranžovou barvou a akumulátor by měl být v nejbližší době vyměněn, ale provoz robotu je stále možný. V případě kritického vybití akumulátoru, napětí menší než 11 V, svítí pole s napětím akumulátoru červeně. Dojde k automatickému odpojení napájecí větve 3,3 V a 5 V. V tomto stavu by robot neměl být dále používán a akumulátor by měl být co nejdříve znovu nabit, aby se zabránilo jeho poškození a ztrátě kapacity.

   větev 8 V Pokud se napětí na této napájecí větvi pohybuje v rozsahu 7,5–10 V, pole svítí zeleně. Pokud napětí překročí tento rozsah, dojde k odpojení příslušné napájecí větve a pole se rozsvítí červeně.

   větev 5 V Pokud se napětí na této napájecí větvi pohybuje v rozsahu 4,5–5,5 V, pole svítí zeleně. Pokud napětí překročí tento rozsah, dojde k odpojení příslušné napájecí větve a pole se rozsvítí červeně.

   větev 3,3 V Pokud se napětí na této napájecí větvi pohybuje v rozsahu 3–4 V, pole svítí zeleně. Pokud napětí překročí tento rozsah, dojde k odpojení příslušné napájecí větve a pole se rozsvítí červeně.

3. **Stav systémů na CAN sběrnici**

   MOT – řízení motorů Pole **MOT** signalizující stav desky Driver board , viz 6 v tabulce 6.1, má následující význam. Pokud řídicí aplikace dostává od modulu zprávy a žádná chyba není signalizována, pole svítí zeleně. Pokud pole svítí oranžově, řídicí aplikace dostává od modulu zprávy signalizující chybu sfázováni motoru s IRC senzorem. V takovém případě je potřeba stisknout stop tlačítko a popojet robotem tak, aby se motory otočily alespoň o jednu

otáčku. Pokud signalizace chyby nezhasne ani po otočení motorů, jedná se o chybu, která musí byt diagnostikována připojením sériové linky/USB a robot nelze dále používat. Pokud řídicí aplikace nedostává od modulu žádné zprávy, pole svítí červeně.

ODO – odometrie Pole **ODO** signalizující stav desky Odometry board, viz 5 v tabulce 6.1, svítí zeleně, pokud řídicí aplikace dostává od modulu zprávy. Pokud dojde k selhání tohoto modulu a řídicí aplikace nedostává zprávy od modulu, pole svítí červeně.

PWR – řízení napájení Pole **PWR** signalizující stav desky PWR board, viz 10 v tabulce 6.1, svítí zeleně, pokud řídicí aplikace dostává od modulu zprávy. Pokud dojde k selhání tohoto modulu a řídicí aplikace nedostává zprávy od modulu, pole svítí červeně.

MAN – řízení manipulátoru Pole **MAN** signalizující stav desky Eb board, viz 12 v tabulce 6.1, svítí zeleně, pokud řídicí aplikace dostává od modulu zprávy. Pokud dojde k selhání tohoto modulu a řídicí aplikace nedostává zprávy od modulu, pole svítí červeně.

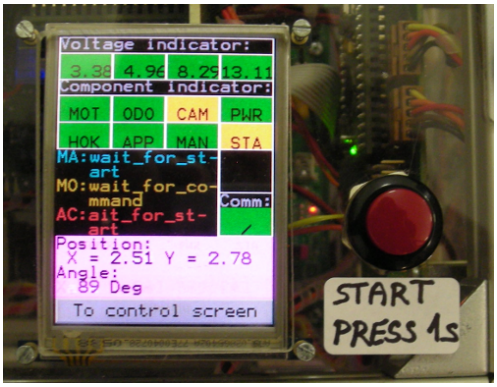
Vzhledem k problému s řadiči CAN sběrnice v procesorech LPC2119, dochází příležitostně k selhání komunikace s deskou PWR board a Eb board. Vzhledem k absenci resetovacího tlačítka na těchto deskách lze problém vyřešit pouze vypnutím a zapnutím hlavního vypínače, viz `http://www.nxp.com/documents/errata_sheet/ES_LPC2109_19_29_01.pdf`

4. **Stav systémů na USB sběrnici**

HOK – rangefinder Pokud řídicí aplikace dostává data od aplikace *hokuyod* zpracovávající data z rangefinderu, pole **HOK** signalizující stav aplikace, viz 7 v tabulce 6.1, se rozsvítí zeleně. Pokud řídící aplikace nedostává žádná data od aplikace *hokuyod*, pole svítí červeně.

CAM – kamera Pokud řídicí aplikace dostává data od aplikace *barcam* zpracovávající obraz z webové kamery, pole **CAM** signalizující stav aplikace, viz 9 v tabulce 6.1, se rozsvítí zeleně. Pokud řídicí aplikace nedostává žádná data od aplikace *barcam*, pole svítí červeně.

Pokud dojde z jakéhokoliv důvodu k selhání a ukončení některé z předchozích aplikací, operační systém po několika sekundách tuto aplikaci automaticky spustí znovu.



Obrázek 6.5: Umístění startovacího tlačítka a zobrazení stavu na displeji

6.4 Odstartování demonstrační aplikace

Pro odstartování demonstrační aplikace se používá malé červené tlačítko vpravo od displeje, viz obrázek 6.4. Před odstartováním aplikace je potřeba zkontrolovat, že je demonstrační aplikace připravena. To je signalizováno zeleně svítícím polem s nápisem **APP**, viz 8 v tabulce 6.1. Pokud ikona **APP** svítí oranžově, některý systém robotu není připraven. Pokud ikona svítí červeně, aplikace byla ukončena. Během několika sekund by však mělo dojít znovu k automatickému spuštění aplikace. Sekce 6.5 popisuje chování robotu po odstartování aplikace. Odstartování *demo* aplikace poté probíhá v následujících krocích.

1. Po zapnutí robotu je demonstrační aplikace v předstartovním stavu, signalizace **APP** svítí oranžově a pole signalizující stav startovacího tlačítka **STA**, viz 11 v tabulce 6.1, svítí také oranžově.

2. Po krátkém stisku startovacího tlačítka dojde k zavření manipulátoru do výchozí pozice, pokud byl tento při zapnutí napájení v jiné pozici. Signalizace startovacího tlačítka se rozsvítí zeleně a signalizace **APP** se také rozsvítí zeleně.

3. K odstartování demonstrační aplikace dojde po stisku a držení startovacího tlačítka po dobu 1-2 sekund. Signalizace startovacího tlačítka i aplikace se znovu rozsvítí oranžově.

4. Krátký stisk startovacího tlačítka kdykoli po odstartování robotu způsobí ukončení demonstrační aplikace. Operační systém aplikaci automaticky znovu spustí a ta přejde do stavu čekání na start a signalizace startovacího tlačítka i aplikace se rozsvítí zeleně.

6.5 Popis chování demonstrační aplikace

Následující body popisují chování robotu tak jak probíhá po odstartování aplikace.

1. (a) Robot stojí na místě a hledá cíl v okolí pomocí rangefinderu.
   (b) Pokud robot nedetekuje cíl, otočí se na místě o 120 ° a detekuje znovu.
   (c) Takto robot prozkoumá celý rozsah 360 °.
   (d) Pokud robot z aktuální pozice nedetekuje žádný cíl, pokračuje bodem 2.
   (e) Pokud robot detekuje cíl, jede k cíli na bezpečnou vzdálenost asi 0,5 m a pokračuje od bodu 3.
2. (a) Robot vygeneruje náhodnou pozici v rámci povoleného prostoru a jede na tuto pozici a pokračuje bodem 1.
   (b) Pokud se robotu nepodaří dosáhnout nové pozice během tří pokusů, například kvůli překážce, zastaví se a pokračuje od bodu 1.
3. (a) Robot stojí nedaleko od cíle a pomocí kamery rozhoduje zda se jedná o hledaný cíl.
   (b) Pokud je cíl považován za platný, pokračuje bodem 4.
   (c) Pokud je cíl neplatný, pokračuje bodem 2.
4. (a) Robot stojí nedaleko cíle a pomocí rangefinderu určí pod jakým úhlem vůči dopředné ose robotu je umístěn cíl.
   (b) Robot se otočí tak, aby cíl byl umístěn v dopředné ose.
   (c) Robot popojede 2 cm vpřed k cíli.
   (d) Robot opakuje kroky 4b a 4c dokud nestojí maximálně 5 cm od cíle, poté pokračuje bodem 5.
   (e) Během operace přibližování k cíli nesmí ve vzdálenosti 1 m od robotu stát žádná osoba!
5. (a) Robot stojí těsně u cíle a pomocí manipulátoru naloží náklad umístěný na cílovém podstavci.
   (b) Po naložení cíle robot couvá asi 0,3 mm směrem od podstavce. Pokud při couvání robot narazí do překážky, zastaví se a čeká dokud překážka není odstraněna a pokračuje bodem 6

6. (a) Robot stojí s vyzvednutým nákladem u cílového podstavce.

   (b) Robot převeze náklad na startovní pozici a aplikace skončí. V závislosti na typu povrchu po kterém se robot pohybuje a vzhledem k nepřesnosti měření odometrie se robot nemusí vrátit přesně do stejné pozice.

Pokud kdykoli během provozu demonstračního robotu dojde k neočekávanému a nebezpečnému chování robotu nebo ohrožení osob, ihned stiskněte červené bezpečnostní tlačítko nebo úplně vypněte robot hlavním vypínačem!

6.6 Použití simulátoru robomon

Simulátor *robomon* je program pro osobní počítač určený k testování aplikací před jejich spuštěním na robotu. Zároveň slouží pro monitorování a vizualizaci skutečného stavu robotu, zobrazení dat naměřených rangefinderem a zobrazení vytvořené mapy prostředí podle které robot plánuje pohyb v prostoru.

Podmínkou pro spuštění simulátoru *robomon* a souvisejících aplikací je použití PC s operačním systémem Linux. Všechny programy jsou zkompilované pro platformu x86/32 bit. Lze je tedy spouštět jak na 32 bitovém tak 64 bitovém systému.

Aplikace *robomon* je společně s dalšími programy k dispozici na přiloženém CD v adresáři /bin–x86. Před použitím programů je nutné nainstalovat některé sdílené knihovny na kterých jsou programy závislé. Seznam těchto závislostí je uveden v tabulce C.1 v příloze, tabulka C.2 obsahuje seznam Debian/Ubuntu balíků, které obsahují potřebné knihovny.

1. **Offline spuštění demo aplikace na PC**

   Simulátor *robomon* lze použít pro částečné (nedochází k rozpoznávání a vyzvedávání nákladu) předvedení demonstrační aplikace bez použití robotu následujícím postupem.

   (a) V terminálu spusťte program *ortemanager* s parametrem `-e`.

   ```
   $ ortemanager -e
   ```

   (b) Spusťte program *robomon*. Po jeho spuštění by v terminálu se spuštěným ortemanagerem měl přibýt jeden řádek v následujícím formátu.

   ```
   $ ortemanager -e
   application 0xc0a80104-0xc09e01 was accepted
   ```

(c) V dalším terminálovém okně spusťte program *demo*. Po jeho spuštění by v terminálu se spuštěným ortemanagerem měl přibýt další řádek.

```
$ ortemanager -e
application 0xc0a80104-0xc09e01 was accepted
application 0xc0a80104-0xc09e01 was accepted
```

Při spuštění *demo* aplikace v simulátoru aplikace nečeká na startovací tlačítko a ihned je odstartována. Ve výpisu v terminálovém okně lze sledovat průchod aplikace jednotlivými stavy.

```
$ demo
fsm  mot: init(EV_ENTRY)
fsm  mot: init(EV_EXIT)
fsm  mot: wait_for_command(EV_ENTRY)
fsm MAIN: init(EV_ENTRY)
fsm MAIN: init(EV_EXIT)
fsm MAIN: wait_for_start(EV_ENTRY)
fsm MAIN: wait_for_start(EV_START)
fsm MAIN: wait_for_start(EV_EXIT)
fsm MAIN survey EV_ENTRY: survey
fsm MAIN survey EV_ENTRY: no target
fsm  mot: wait_for_command(EV_NEW_TARGET)
fsm  mot: wait_for_command(EV_EXIT)
fsm  mot: movement(EV_ENTRY)
fsm  mot: movement(EV_TRAJECTORY_DONE_AND_CLOSE)
fsm  mot: movement(EV_EXIT)
fsm  mot: wait_for_command(EV_ENTRY)
fsm MAIN survey EV_ENTRY: survey
```

(d) Nastavení simulátoru.

i. V okně simulátoru lze v menu *View* zapnout použití OpenGL a zlepšit tak rychlost překreslování simulátoru. Toto nastavení není výchozí, neboť nefunguje správně s některými ovladači grafických karet.

ii. Kliknutím kurzoru myši do vymezeného prostoru hřiště dojde k umístění simulované překážky a zapnutí simulace měření rangefinderu.

iii. V menu *View* lze dále zapnout zobrazení sdílené mapy.

Ukázka spuštění *demo* aplikace na PC v simulátoru *robomon* je na obrázku 6.6.

2. **Online monitorování *demo* aplikace spuštěné na robotu**

   Simulátor *robomon* lze použít ke sledování stavu aplikace spuštěné na reálném robotu následujícím postupem.

   (a) Připravte robot ke startu podle návodu v sekcích 6.1–6.4.

   (b) Připojte PC k bezdrátové Wi–Fi síti robotu.

SSID	CTU_demo
zabezpečení	WPA & WPA2 Personal
heslo	demorobot

   (c) V terminálu spusťte program *ortemanager* s parametrem `-p 10.1.1.1 -e`. Pokud je PC správně připojeno k Wi–Fi robotu, po krátké chvilce *ortemanager* akceptuje všechny aplikace spuštěné na robotu.

   ```
   $ ortemanager -p 10.1.1.1 -e
   application 0xc0a80104-0xdd4401 was accepted
   application 0xc0a80104-0x883a01 was accepted
   application 0xc0a80104-0x8b7401 was accepted
   application 0xc0a80104-0xcb0f01 was accepted
   ```

   (d) Spusťte program *robomon*.

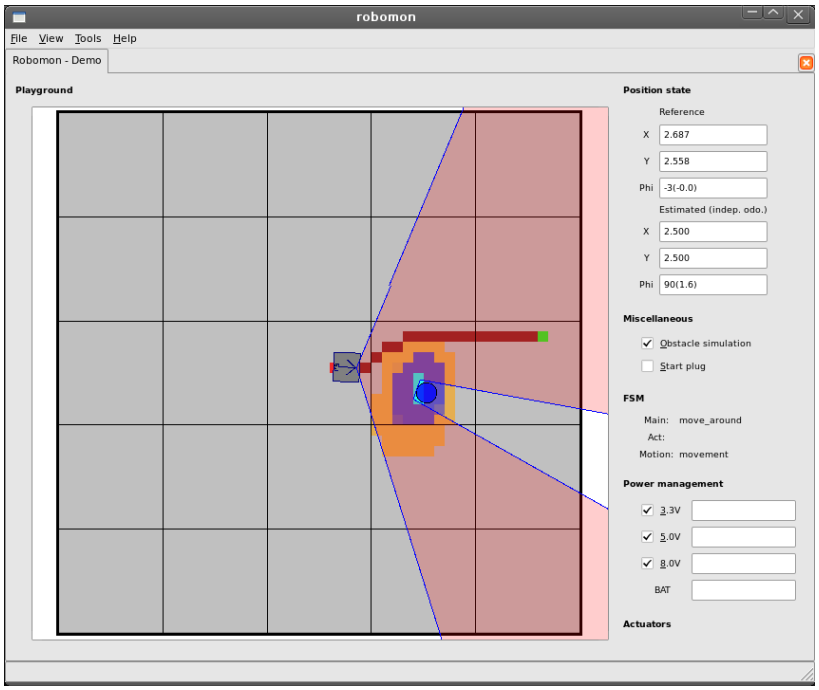
   (e) Nastavení simulátoru.

      i. V okně simulátoru lze v menu *View* zapnout použití OpenGL a zlepšit tak rychlost překreslování simulátoru.

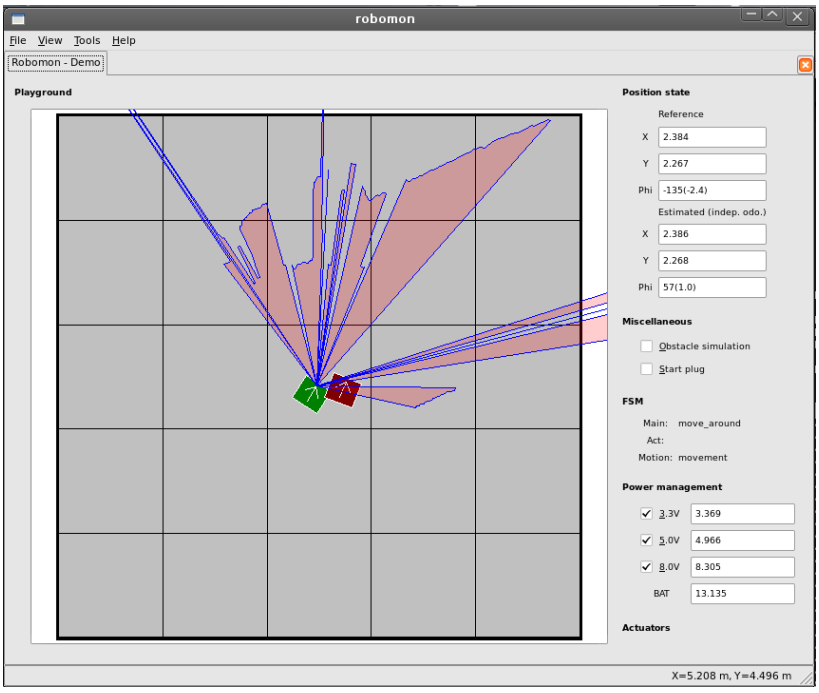
      ii. V menu *View* lze dále zapnout zobrazení měření vlečné odometrie a měření IRC snímačů na motorech.

   Ukázka sledování *demo* aplikace spuštěné na robotu pomocí simulátoru *robomon* je na obrázku 6.7.

   V případě spuštění* demo *aplikace na robotu nelze v robomonu zobrazit sdílenou mapu!



Obrázek 6.6: Spuštění *demo* aplikace v simulátoru *robomon*



Obrázek 6.7: Online monitorování *demo* aplikace spuštěné na robotu v simulátoru *robomon*

Kapitola 7

Závěr

Cílem této práce bylo vytvořit mobilní robot vhodný pro prezentaci ČVUT při různých příležitostech jako jsou dny otevřených dveří a podobně.

Při vytváření tohoto demonstračního robotu jsem vycházel z již postaveného robotu, vytvořeného robotickým týmem Flamingos pro soutěž Eurobot v roce 2010. Tento robot jsem kompletně rozebral a veškeré komponenty zkontroloval a otestoval. Téměř na všech součástech původní elektroniky jsem nalezl nějaké nedostatky. Pokusil jsem se o opravu všech závad a nedostatků které jsem na elektronice objevil a také jsem veškeré nalezené chyby zdokumentoval na wiki týmu Flamingos, aby byly přístupné i budoucím členům týmu v dalších letech.

Otestované a opravené komponenty jsem následně znovu nainstaloval do původní kostry robotu tak, aby byly snadno přístupné pro případnou budoucí opravu nebo výměnu. Všechny elektronické komponenty jsem propojil novou kabeláží. Kladl jsem velký důraz na kvalitu provedení této práce, jelikož právě problémy s vypadávajícími konektory a nekvalitně zaletovanými kabely byly dříve častou příčinou selhání všech robotů používaných při soutěži Eurobot.

Pro demonstrační robot jsem navrhl a naprogramoval novou řídicí aplikaci řešící zajímavou a potenciálně užitečnou úlohu. Při vytváření této aplikace jsem také bohatě využil software napsaný dalšími členy týmu, mnohdy již před mnoha lety, pro účely soutěže Eurobot.

Pro vytvořený demonstrační robot jsem také vytvořil návod k obsluze, aby mohl být snadno používán i zaškolenými laiky.

Na všech částech robotu a v software jsem provedl takové změny a opatření, aby byl výsledný systém spolehlivý, jednoduchý na obsluhu a k provozu bylo potřeba pouze minimum pomůcek. Výsledný robot funguje velmi dobře a může být okamžitě použit pro prezentaci univerzity.

Literatura

[1] **Jiří Kubias, Hardware robota pro soutěž Eurobot**, Diplomová práce, ČVUT, 2010 [*online*].
<http://support.dce.felk.cvut.cz/mediawiki/images/d/d5/Dp_2010_kubias_jiri.pdf> cit. 11/2011.

[2] **Filip Jareš, Implementace stavových automatů pro soutěž Eurobot 2009**, Bakalářská práce, ČVUT, 2010 [*online*].
<http://support.dce.felk.cvut.cz/mediawiki/images/5/50/Bp_2010_jares_filip.pdf> cit. 11/2011.

[3] **Jaroslav Šach, Vývoj elektroniky pro mobilní roboty**, Diplomová práce, ČVUT, 2011 [*online*].
<http://measure.feld.cvut.cz/cs/system/files/files/cs/vyuka/zaverecne_prace/DP_2011_Sach_locked.pdf> cit. 11/2011.

[4] **Jan Benda, CANOpen komunikace pro mobilního robota**, Diplomová práce, ČVUT,2008 [*online*] .
<http://support.dce.felk.cvut.cz/mediawiki/images/e/e4/Dp_2008_benda_jan.pdf> cit. 12/2011.

[5] **Petr Beneš, Plánování hladké trajektorie pro mobilní roboty**, Bakalářská práce, ČVUT,2009 [*online*] .
<http://support.dce.felk.cvut.cz/mediawiki/images/5/59/pr_benes_p.pdf> cit. 12/2011.

[6] **Petr Kubizňák, Kamerové rozpoznávání konfigurace herních prvků v soutěži Eurobot**, Bakalářska práce, ČVUT,2010 [*online*] .
<http://cyber.felk.cvut.cz/research/theses/papers/197.pdf>
cit. 12/2011.

[7] **Konrad Skup, Motion Control for Mobile Robots**, Diplomová práce, ČVUT, 2007 [*online*] .
<http://epubl.ltu.se/1653-0187/2007/061/LTU-PB-EX-07061-SE.pdf>
cit. 12/2011.

[8] **Martin Žídek, Software for Eurobot Competition and its Timing Analysis**, Diploma thesis, ČVUT, 2009 [*online*] .
<http://rtime.felk.cvut.cz/~sojka/students/Dp_2009_zidek_martin.pdf> cit. 12/2011.

[9] **João Xavier, Marco Pacheco, Daniel Castro, António Ruano, Urbano Nunes, Fast Line, Arc/Circle and Leg Detection from Laser Scan Data in a Player Driver**, in Proc. of the IEEE Int. Conference on Robotics & Automation ICRA'05, 2005 [*online*] .
<http://w3.ualg.pt/~dcastro/a1738.pdf> cit. 11/2011.

[10] **Eurobot**, mezinárodní amatérská robotická soutěž [*online*].
<http://www.eurobot.org/eng/presentation.php> cit. 12/2011.

[11] **progeCAD**, konstrukční 2D&3D program [*počítačový program*].
<http://www.solicad.com/progecad> cit. 12/2011.

[12] **ArtCAM Insignia**, 3D Design&CNC Machining Software [*počítačový program*].
<http://www.artcaminsignia.com/> cit. 12/2011.

[13] **Google SketchUp**, software pro 3D modelování [*počítačový program*].
<http://sketchup.google.com/> cit. 12/2011.

[14] **Robotic wiki**, wiki pro účely robotického týmu Flamingos, ČVUT, 2007-2011 [*online*].
<http://rtime.felk.cvut.cz/robot/index.php> cit. 12/2011.

[15] **Michal Vokáč, Comagrav NT Profi**, návod pro obsluhu CNC frézky na wiki týmu Flamingos, ČVUT, 2011 [*online*].
<http://rtime.felk.cvut.cz/robot/index.php/Comagrav_MT_Profi>
cit. 11/2011.

[16] **Mikroklima**, vyroce modulu MIDAM Shark [*online*].
<http://www.midam.cz/products/MIDAM-SHARK.html> cit. 12/2011.

[17] **H8eurobot**, popis modulu pro řízeni BLDC motorů na robotické wiki, ČVUT, 2007-2011 [*online*].
<http://rtime.felk.cvut.cz/robot/index.php/H8eurobot> cit. 11/2011.

[18] **Martin Synek, Shape detection library**, popis knihovny pro detekci tvarů v datech naměřených pomocí laserového dálkoměru, stránka s automaticky generovanou dokumentací zdrojového kódu, ČVUT, 2011 [*online*].
<http://rtime.felk.cvut.cz/dragons/doc/group__shapedet.html>
cit. 11/2011.

[19] **Hokuyo**, výrobce elektronických komponent [*online*].
<http://www.hokuyo-aut.jp/> cit. 12/2011.

[20] **Boa5200 HOWTO**, návod na wiki popisující instalaci a nastavení OS Linux pro desku Boa5200, ČVUT,
2005-2011 [*online*].
<http://rtime.felk.cvut.cz/hw/index.php/HOWTO> cit. 12/2011.

[21] **Petr Smolík, OCERA Real-Time Ethernet**, ČVUT [*online*].
<http://www.ocera.org/download/components/WP7/orte-0.3.1.html> cit. 12/2011.

[22] **OCERA**, Open Components for Embedded Real-time Applications [*online*].
<http://www.ocera.org/index.html> cit. 12/2011.

[23] **Flamingos**,webové stránky robotického týmu založeného na katedře řídicí techniky, ČVUT, 2010-2011 [*online*].
<http://flamingos.felk.cvut.cz> cit. 12/2011.

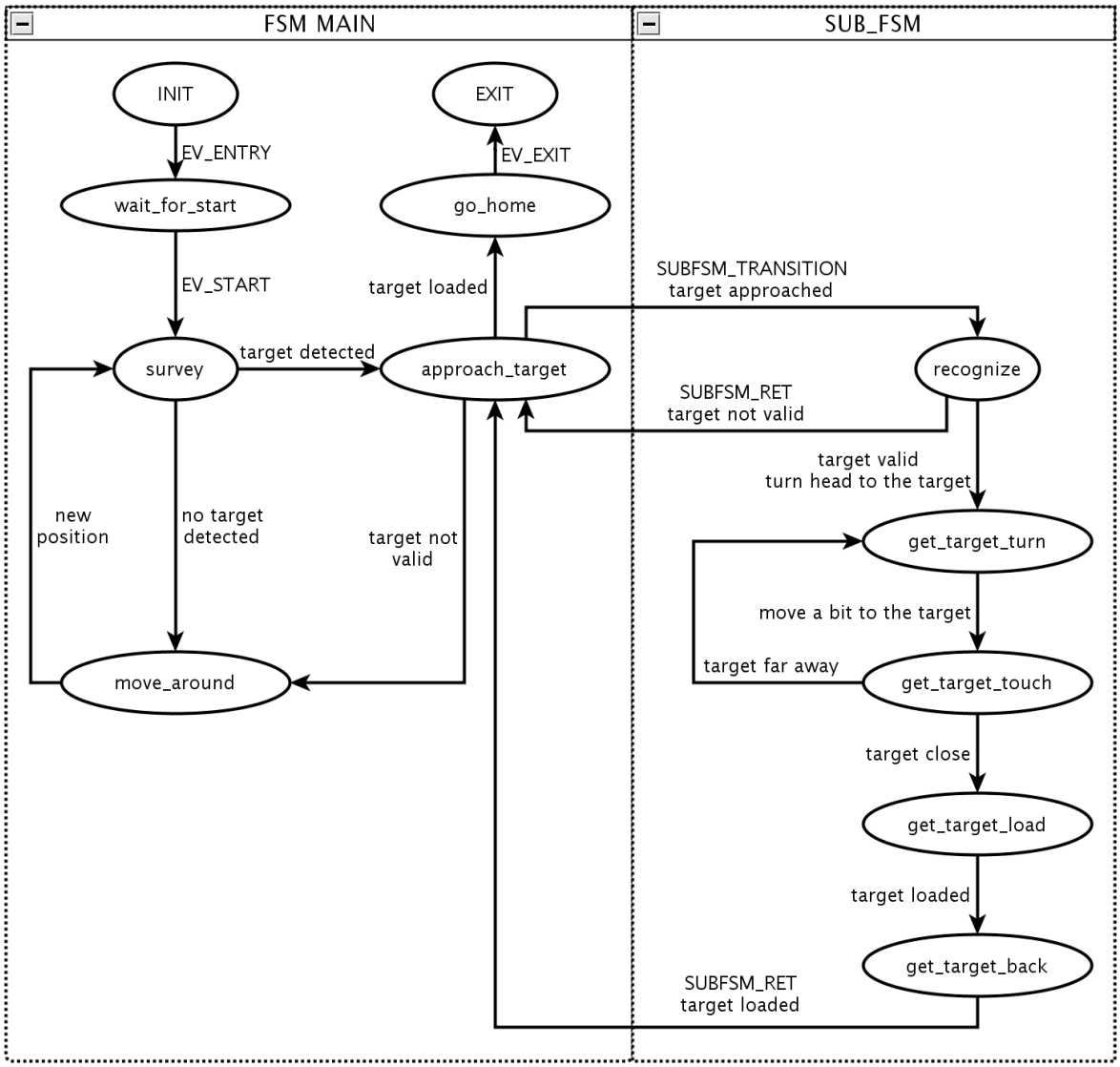
[24] **NXP errata**, dokument popisující známé problémy procesorů řady LPC21x9, NXP, 2011 [*online*].
<http://www.nxp.com/documents/errata_sheet/ES_LPC2109_19_29_01.pdf> cit. 12/2011.

[25] **Gymšpit**, webové stránky robotického týmu, [*online*].
<http://robot.gymspit.cz/new/en/uncategorized/> cit. 12/2011.

[26] **PiKRON**, Ing. Pavel Píša, firma Pikron [*online*].
<http://www.pikron.com/> cit. 12/2011.

Příloha A

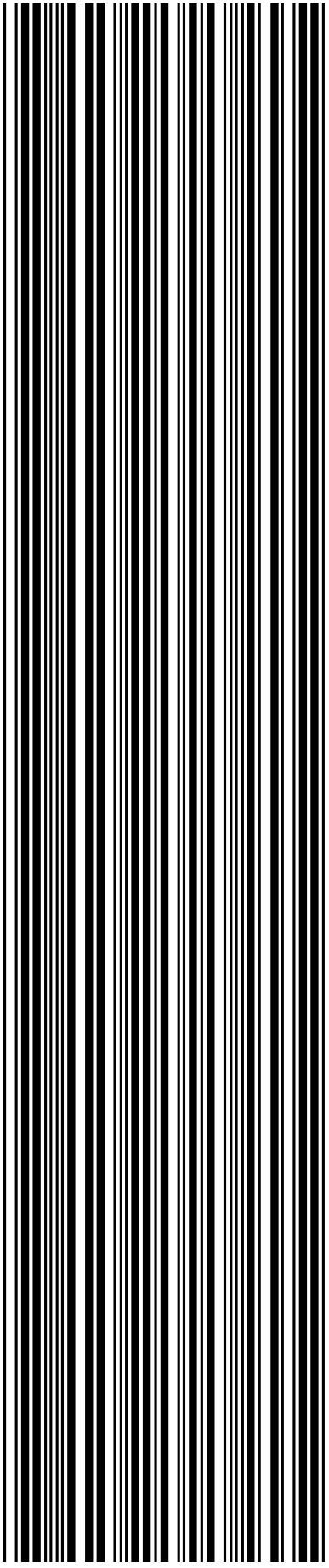
Vývojový diagram hlavního stavového automatu



Obrázek A.1: Vývojový diagram stavů hlavního automatu demonstrační aplikace

Příloha B

Čárový kód pro označení cílového podstavce



Obrázek B.1: Čárový kód pro označení cílového podstavce

Příloha C

Závislost PC aplikací na sdílených knihovnách

robomon
linux-gate.so.1
libQtOpenGL.so.4
libQtGui.so.4
libQtCore.so.4
libGLU.so.1
libGL.so.1
libpthread.so.0
libstdc++.so.6
libm.so.6
libgcc_s.so.1
libc.so.6
libfreetype.so.6
libdl.so.2
libXrender.so.1
libX11.so.6
libfontconfig.so.1
libaudio.so.2
libglib-2.0.so.0
libpng12.so.0
libz.so.1
libgobject-2.0.so.0
libSM.so.6
libICE.so.6
libXext.so.6
libgthread-2.0.so.0
librt.so.1
libXxf86vm.so.1
libXdamage.so.1
libXfixes.so.3
libdrm.so.2
/lib/ld-linux.so.2
libxcb.so.1
libexpat.so.1
libXt.so.6
libXau.so.6
libpcre.so.3
libuuid.so.1
libXdmcp.so.6

ortemanager
linux-gate.so.1
libpthread.so.0
libc.so.6
/lib/ld-linux.so.2

demo
linux-gate.so.1
libpthread.so.0
librt.so.1
libstdc++.so.6
libm.so.6
libgcc_s.so.1
libc.so.6
/lib/ld-linux.so.2

Tabulka C.1: Tabulka závislostí PC aplikací na sdílených knihovnách

robomon
libqt4-opengl
libqtgui4
libqtcore4
libglu1-mesa
libc6
libstdc++6
libgcc1
libfreetype6
libxrender1
libx11-6
libfontconfig1
libaudio2
libglib2.0-0
libpng12-0
zlib1g
libsm6
libice6
libxext6
libglib2.0-0
libxxf86vm1
libxdamage1
libxfixes3
libdrm2
libxcb1
libexpat1
libxt6
libxau6
libpcre3
libuuid1
libxdmcp6

ortemanager
libc6

demo
libc6
libstdc++6
libgcc1

Tabulka C.2: Ubuntu/Debian balíky obsahující sdílené knihovny

Příloha D

Obsah CD

/bin-x86 Binární programy zkompilované pro platformu x86/32bit.

robomon

ortemanager

demo

/dp Text této diplomové práce ve formátu .pdf.

/barcode Obrázek s čárovým kódem pro označení cílového podstavce, verze vhodná pro tisk na list A4.