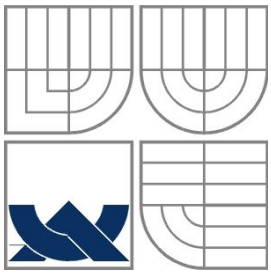
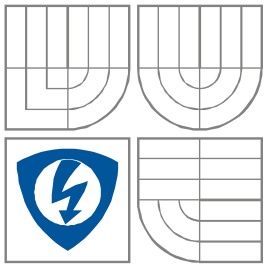


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

Použití mobilního robotu v inteligentním domě

MOBILE ROBOT IN SMART HOUSE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ KUPAROWITZ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR PETYOVSKÝ

BRNO 2013

Zadání diplomové práce

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Tomáš Kuparowitz	***ID:***	120789
Ročník:	2	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Použití mobilního robotu v inteligentním domě

POKYNY PRO VYPRACOVÁNÍ:

Výsledkem práce bude mobilní robot vybavený pro pohyb ve vnitřních prostorách, sloužící k vizualizaci nestandardních situací v prostoru inteligentního domu.

1. Seznamte se s možnostmi inteligentního domu v oblasti senzorických systémů, technologií zpracování získaných informací a možnostmi jejich využití autonomními roboty.
2. Nastudujte problematiku autonomních robotů v souvislosti s použitím v inteligentním domě.
3. Prozkoumejte trh a zvolte z dostupných řešení autonomních robotů takové, které vyhovují dané úloze, případně navrhněte vlastní řešení robotu.
4. Navrhněte způsoby lokálního vyhodnocování a reakce na podněty senzorů autonomního robotu.
5. Simulujte a následně realizujte SW ovládání robotu pomocí počítače.
6. Implementujte algoritmy ovládání robotu pomocí počítače a lokalizace robotu v prostředí domu.
7. Navrhněte a realizujte rozhraní pro spolupráci robotu s inteligentním domem (ovládacím serverem).
8. Zhodnoťte dosažené výsledky, možnosti aplikace vytvořeného robotu v inteligentních budovách, výhody a omezení zvolené koncepce a navrhněte další možná rozšíření pro navazující práci.

DOPORUČENÁ LITERATURA:

[1] Everett, H., R.: Sensors for Mobile Robots theory and application, 1995; ISBN: 978-1568810485
[2] Prassler, E.; Stopp, A. Eds.: Advances in Human-Robot Interaction, Springer 2005, ISBN: 978-3-540-23211-7

Termín zadání:	11.2.2013	***Termín odevzdání:***	9.8.2013

Vedoucí práce: Ing. Petr Petyovský
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

Abstrakt

Zadáním této práce je prozkoumat trh a vybrat vhodný autonomní robot pro spolupráci s inteligentním domem. Součástí práce je rešerše schopností inteligentních domů v oblasti senzorických systému, možnosti zpracování získaných dat a jejich následné využití autonomními roboty. Součástí práce je implementace ovládání robotu ve vývojovém prostředí Microsoft Robotics Studio (C#) a jeho simulace pomocí simulátoru Visual Simulation Environment. V práci je navrženo a realizováno komunikační rozhraní mezi robotem a inteligentním domem a komunikační rozhraní mezi robotem a uživatelem. Rozhraní robotu nabízí přímé ovládání a automatické navádění robotu v rámci inteligentního domu. Aplikace je vybavena pro navigaci a pohyb v dynamickém prostředí inteligentního domu. Robot je schopný registrovat nové překážky a pracovat s nimi.

Klíčová slova

inteligentní dům, autonmoní robot, robotický vysavač, bezdrátové ovládání, wi-fi wifi, iRobot Create, microsoft robotic developer studio, visual simulation environment, senzory inteligentního domu, lokalizace robotu, dead-reckoning, nejistoty lokalizace za použití metody dead-reckoning, dynamická mapa prostředí, lokalizace metodou vektorových polí, rozpoznání vzoru

Abstract

Aim of this thesis is to search the market for suitable autonomous robot to be used by smart house. The research in this work is partly done on the range of abilities of smart houses in matter of sensor systems, ability of data processing and their use by mobile robots. The output of this thesis is robotics application written using Microsoft Robotics Developer Studio (C#) and simulated using Visual Simulation Environment. Main feature of this application is the interface between robot and smart house, and robot and user. This interface enables employer to directly control robot's movement or to use automated pathfinding. The robot is able to navigate in dynamic environment and to register, interact and eventually forget temporary obstacles.

Keywords

smart house, digital home, autonomous robot, robotic vacuum cleaner, wireless controll, wi-fi wifi, iRobot Create, microsoft robotic developer studio, visual simulation environment, robot localization, dead-reckoning, localization error using dead-reckoning, mapping dynamic environment, vector field path planning, glyph recognition

Bibliografická citace:

KUPAROWITZ, T. *Použití mobilního robotu v inteligentním domě.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2013. 108 s. Vedoucí diplomové práce Ing. Petr Petyovský.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma *Použití mobilního robotu v inteligentním domě* jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **9. srpna 2013** ……

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Petru Petyovskému za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

Děkuji slečně Mgr. Elišce Poláčkové za zapůjčení netbooku, který sloužil jako ovládací stanice použitého robotu.

V neposlední řadě děkuji svému otci Ing. Václavu Velískovi za konečnou korekturu mé diplomové práce, a děkuji též autorům všech v mé práci citovaných zdrojů.

V Brně dne: **9. srpna 2013** ……

podpis autora

Obsah

1 ÚVOD

Bill Gates ve svém článku z roku 2006 nastínil myšlenku, že roboty operující v každé domácnosti budou v budoucnu naprosto běžným jevem [11]. Jeho vize se díky technologickému rozvoji pomalu stala realitou a současné technologie dokonce začínají tuto vizi postupně přesahovat. Nejenže jsou dnes autonomní roboty k vidění ve velkém množství domácností, ale některé plně inteligentní domy s centralizovaným ovládáním je začínají využívat jako prostředek k interakci s okolím. Postupné úplné propojení inteligentních domů s jejich autonomními roboty je přirozeně navazující krok.

V této práci bude provedena rešerše možností současných inteligentních domů a jejich spolupráce s autonomními roboty. Budou probrány samostatné schopnosti autonomních robotů a schopnosti robotů v interakci s inteligentním domem. Dále budou probrány způsoby detekce a způsoby vyhodnocení neobvyklých stavů, které mohou v inteligentním domě nastat.

Výstupem práce bude návrh možného kandidáta na domácí robot. Takový robot musí být vybaven pro pohyb ve vnitřních prostorách inteligentního domu, musí být ovladatelný centrálním počítačem domu a sloužit i k vizualizaci nestandardních a krizových situací uvnitř domu. Kromě těchto činností musí robot samozřejmě zastávat některou z domácích prací, ke které byl primárně určen.

Následně bude vybrána vhodná vývojová platforma pro programování tohoto robotu a vhodný simulátor, ve kterém tento robot bude možné testovat. Výstupem této práce bude tedy i program pro ovládání autonomního robotu, implementující funkce pro pohyb a navádění v domě, uživatelský výstup a živý kamerový přenos z pohledu robotu. Celý program bude přitom obsahovat rozhraní pro komunikaci s inteligentním domem a se vzdáleným uživatelem. Struktura a funkčnost i návod k instalaci bude shrnuta v těle této práce. Zdrojové kódy a binární soubory, potřebné k instalaci robotu, budou společně s některými vybranými referencemi a zajímavými dokumenty týkajícími se problematiky v elektronických přílohách této práce.

V závěru práce budou zmíněny některé z možností dalšího rozvinutí této práce. Nakonec budou rekapitulovány dosažené výsledky, možnosti aplikace vytvořeného řešení, výhody a omezení tohoto řešení.

2 INTELIGENTNÍ DŮM

S rychlým rozvojem elektrotechniky v druhé polovině dvacátého století vznikl expandující fenomén automatizace domácnosti. Především díky vzniku jednočipových počítačů v roce 1971 [1] je dnes naprosto běžné vlastnit například pračku schopnou vykonávat složité prací cykly, televizor ovládaný na dálku, či vysoce sofistikované bezpečnostní systémy. Všechna tato jednoúčelová elektronická zařízení tvoří základ automatizace domácnosti.

Prvním pokusem automatizovat dům byl nejspíše ambiciózní projekt inženýra Jamese Sutherlanda [2]. Ten v roce 1965 zkonstruoval domácí počítač z programovatelných obvodů, vyřazených jeho zaměstnavatelem firmou Westinghouse Electric. Tato společnost již v roce 1959 zkonstruovala víceúčelový počítač PRODAC IV, jehož pozůstatky Sutherland částečně využil pro vlastní aplikaci. Na základě názvu svého předchůdce byl první domácí počítač pojmenován ECHO IV. Ten sloužil jako jednoduchý psací stroj (byl opatřen několika klávesovými terminály a tiskárnou), pomáhal vytápět dům (jeho odběr činil neskutečných 3,5 kilowat, přičemž bylo možné omezit jeho externí chlazení) nebo ovládal natočení televizní antény v závislosti na prohlíženém kanálu. Než byl v roce 1976 vyřazen, měl Sutherland v plánu na něm implementovat aplikaci pro předpověď počasí a nástroj udržující přehled o skladovaných potravinách.

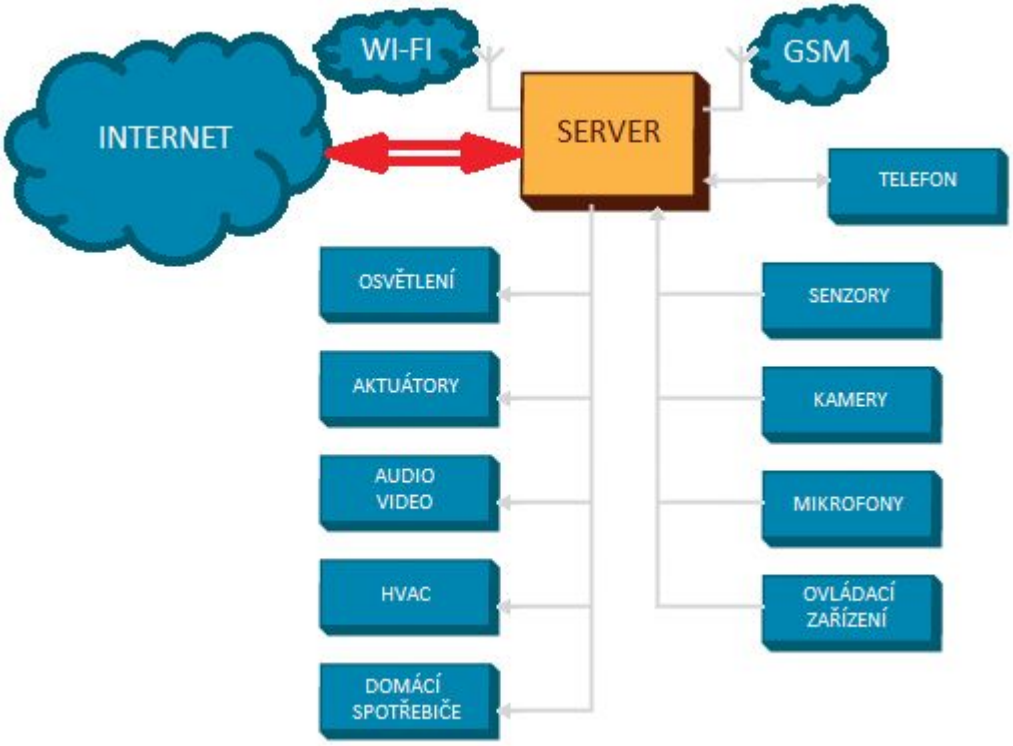
V dnešní době je již vlastnictví domácího počítače neodmyslitelnou samozřejmostí. Novinky implementované v domácnosti se tedy týkají převážně vylepšení starých konceptů. I přes vysoce vyspělou úroveň použité techniky se však u většiny automatizovaných domácností stále jedná pouze o pseudo-inteligenci.

O skutečně inteligentním domě mluvíme v případě, že jsou všechna jeho zařízení optimálně propojena a schopna spolupracovat při vykonávání konkrétně zadaných úloh. Prováděné kroky dům kalkuluje sám na základě požadavků uživatele nebo po vyhodnocení potřeby takové akce provést. Přímý přístup k zařízení domu a jejich použití uživatelem ovšem nesmí být omezeno; nechtěný nebo negativní dopad činnosti přístrojů na obyvatele domu musí být minimalizován. V rámci této práce je při použití pojmu inteligentní dům myšlen právě takovýto model.

2.1 Struktura inteligentního domu

Současné inteligentní domy jsou velice komplexní nástroje. Není důvod si myslet, že takovýchto domů nebude v budoucnu pouze přibývat. S rostoucím zájmem o stavbu nových inteligentních domů roste i nabídka a cenová dostupnost jejich vybavení. Přesto, že druhů a způsobů využití nových zařízení stále přibývá, jisté struktury a zákonitosti

systému inteligentního domu jsou již zavedeny. Při tvorbě přístrojů a především ovládacího software inteligentních domů je kladen důraz především na modularitu a kompatibilitu.



Obr 2.1: Struktura inteligentního domu

Jádrem inteligentního domu bývá domácí počítač. Ten je na Obr 2.1 označen jako server a je připojen ke všem zařízením v domě. Podrobněji je jeho funkce popsána v kapitole 2.6.

Přímo s uživatelem přijdou do styku výstupní zařízení domácnosti a senzory. Výstupním zařízením se myslí například audiovizuální prostředky nebo osvětlení domácnosti; podrobněji je jejich problematika popsána v kapitole 2.2. Senzory používané inteligentním domem jsou zevrubně popsány v kapitole 2.5.

Propojení mezi periferním zařízením a serverem je provedeno pomocí zavedených domácích sítí. V jednom inteligentním domě může být použito i více druhů domácích sítí - podrobněji jsou nastíněny v kapitole 2.4.

2.2 Úkoly inteligentních domů

Inteligentní domy sebou přirozeně nesou mnoho výhod pro jejich obyvatele. Převážně se jedná o způsoby zvýšení komfortu a snížení nároků na chod domácnosti. V této kapitole je rozebráno několik základních odnoží automatizace domácnosti, jejichž

členění je částečně převzato z [3]. V navazujících podkapitolách jsou postupně probrány jednotlivé systémy zobrazené na obrázku 2.1.

2.2.1 Klimatizace (HVAC)

Zabývá se správou ovzduší v domě. Řídí především teplotou, čistotu a vlhkost vzduchu. Práce s těmito třemi aspekty vzduchu bývá shrnuta anglickou zkratkou HVAC (*Heating, Ventilation and Air Conditioning*). [4]

K vytápění se obecně používá elektrických, teplovzdušných nebo vodních zařízení. Ta bývají instalována tak, aby působila ve studených částech místnosti. Vytápěcí cyklus bývá přizpůsoben požadované teplotě v každém okamžiku. Inteligentní dům, například z důvodu úspory energie, může změnit požadovanou teplotu v domě po odchodu uživatelů a poté opět těsně před jejich návratem. [4]

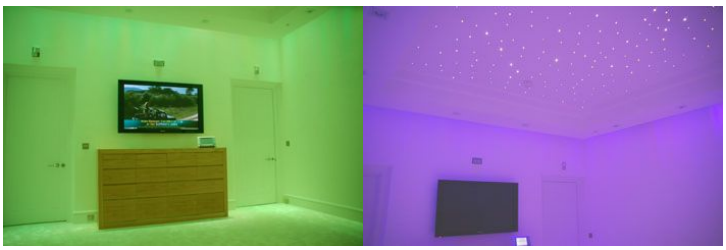
Primární funkcí vzduchové ventilace je výměna vzduchu v domě. Ta slouží k odstranění prachu, bakterií, oxidu uhličitého a jiných cizích nebezpečných částic. Venkovní vzduch bývá přiváděn přes filtry pomocí ventilátorů nebo využitím tahu komínového efektu (otevřením správných průduchů v domě). Inteligentní dům dokonce může obsahovat senzory pro měření kvality vzduchu a na základě jejich výstupu rozhodnout o intenzitě ventilace. [4]

K ochlazení vzduchu se používá většinou odvodnění vzduchu aplikací cyklu komprese, kondenzace a vypařování. Snížením vlhkosti ovzduší se poté efektivně sníží i jeho teplota, protože většina tepla zůstane vázána v odebrané vodě. [4]

2.2.2 Osvětlení

V inteligentních domech bývá běžné centrální ovládání veškerých světelných a elektrických zdrojů. Jeho použití snižuje náklady na provoz a zvyšuje komfort. [3]

Běžně lze měnit odstín a intenzitu osvětlení (náladu pokoje) [3] v závislosti na požadavcích uživatele nebo automaticky podle údajů senzorů či ovladačů.



Obr 2.2: Příklad změny „nálady“ pokoje [5]

Centrální ovládání umožňuje například na dálku vypnout zapomenutá světla nebo zásuvku se zapojenou žehličkou. Dále je možné ve spolupráci s různými detektory

automaticky vypínat světla při nepřítomnosti obyvatel a po jejich návratu je opět zapínat.

Časované spínání a vypínání světel může taktéž sloužit jako součást prevence proti vniknutí zlodějů do domu. Případně je možné všechna světla rozsvítit při detekci narušitele nebo použít blikání jako světelné sirény k přilákání pozornosti kolemjdoucích.

K ovládání osvětlení v domě lze mimo zapínání a vypínání světla použít také jeho potlačení pomocí různých žaluzií, rolet, závěsů nebo LCD stínidel [3].

2.2.3 Audiovizuální systém

Audiovizuální systém, mnohdy v kombinaci se změnou osvětlení, umožňuje sledování oblíbených televizních programů, poslech hudby a podává celkový přehled o chodu domácnosti. [3]

Obecně bývá zvukový a grafický systém použit jako výstupní konzole inteligentního domu. Její pomocí je například možné sledovat kamery, získat přehled o nákladech na chod domu, sledovat stav teplé vody atd.

Další zajímavou funkcí audiovizuálního systému v inteligentním domě může být automatické zaznamenání oblíbených televizních nebo rozhlasových pořadů v případě, kdy obyvatelé nejsou přítomni, nebo přehrávání zvolené hudby při pohybu člověka domem vždy jen ve správné místnosti, případně propojení systému s uživatelovým počítačem či telefonem.

2.2.4 Zabezpečovací a dveřní systém

Zabezpečovací a dveřní systém umožňuje v prvé řadě ochranu před neoprávněným vniknutím cizích osob do domu. Dálkové ovládání oken a dveří poskytuje způsob jak eliminovat možné cesty vniknutí (zapomenutá otevřená okna) nebo naopak ve spolupráci s částečnou deaktivací zabezpečovacího systému umožnit vstup jiným osobám do domu v nepřítomnosti vlastníka (doručovatel může složit zásilku v garáži a návštěva nemusí čekat před domem). [3]

Zabezpečovací systém musí být nejspolehlivější a nejbezpečnější součástí inteligentního domu. Bývá velmi komplexní a může využívat všech ostatních systémů. Například může zprostředkovat přístup ke kamerám přes internet, což umožní uživateli sledovat aktivitu uvnitř a okolo domu z práce nebo na dovolené.

Součástí bezpečnostního systému bývají také senzory kouře, případně senzory úniku vody a plynu. Systém detekce zranění (přivolání lékařské pomoci) může být součástí bezpečnostního systému v případech, kdy dům obývají starší osoby. [3], [12]

2.2.5 Komunikační systém

Komunikační systém spojuje využívání služeb telefonu, „surfování“ po internetu, interkomu a vzájemného dorozumívání uvnitř velkého domu. Pomocí inteligentního audiovizuálního systému lze například nepřerušovaně telefonovat při pohybu domem, ověřit totožnost osoby přede dveřmi odkudkoliv z domu, nebo telefonovat z místnosti do místnosti v rámci domu. Samozřejmostí je schopnost uskutečňovat videohovory. [3]

Inteligentní dům může sám zvolit způsob komunikace s vyžádanou osobou například v závislosti na ceně hovoru tak, že nejprve zkusí telefonovat přes internet a následně využije specifikované telefonní číslo.

2.3 Motory a aktuátory

Motory a aktuátory slouží primárně k manipulaci s jinými objekty. Používají se k automatizovanému otevírání a zavíraní vrat, dveří, oken, krytu bazénu a podobně. Je vhodné, aby motor či aktuátor vracel údaj o své výchylce. Díky této zpětné vazbě je možné mít přehled o stavu manipulovaného předmětu (dveří) bez použití jiných externích senzorů.

2.3.1 Kuchyňské a jiné spotřebiče

Lednice, pračka, kávovar, krmítko domácích zvířat, zavlažovací systém nebo široká škála jiných inteligentních přístrojů včetně robotů, může být ovládána inteligentním domem. Převážně jde o možnost plánování úloh tak, aby byly provedeny v nepřítomnosti obyvatel domu. [3]

Propojení těchto nástrojů s centrálním serverem inteligentního domu umožňuje především jejich snadné používání s možností měnit nastavení ze vzdáleného stanoviště a slouží k eliminaci jejich nesprávné interakce s jinými systémy domu. Jako příklad lze uvést nebezpečí kolize bezpečnostního systému a pohybu automatického vysavače nebo funkce inteligentního kávovaru, kdy může dojít k aktivaci pyrosenzoru nebo detektoru pohybu.

Přestože do této části logicky spadá i problematika využití robotů v inteligentním domě, bude vzhledem k jejímu významu v této práci probrána odděleně v kapitole 3.

2.4 Domácí sítě

Domácí sítě slouží k propojení periferních zařízení inteligentního domu a hlavního domácího počítače. Přestože je vývoj inteligentních domů zatím v počátku, vzniklo již velké množství standardů a druhů domácích sítí. Pro spojení používají rozvodné kabely

elektrické energie (*Power Line Communication*), vlastní druh kabeláže nebo bezdrátovou komunikaci.

V této kapitole je čerpáno převážně ze zdrojů [3] a [6].

2.4.1 PLC

Power Line Communication, neboli komunikace přes rozvodnou síť elektrické energie, využívá již instalovaných drátů normálního napětí pro zasílání dat a kontrolních signálů [7]. Data jsou přenášena pomocí vysokofrekvenčních impulsů, které jsou na jedné straně domácí rozvodné sítě elektrického napětí modulovány a na druhé zpětně dekódovány pomocí modemů.

Původně sloužilo PLC pro komunikaci mezi výměníky elektrické energie, rozvodnami a elektrárnami. V současné době se toto velmi jednoduché, dostupné a robustní řešení používá s oblibou i v inteligentních domech. Přestože přenosové rychlosti těchto dat nejsou nijak vysoké (přesné hodnoty rychlostí pro jednotlivé architektury PLC jsou popsány níže), jejich šířka pásma přenosu zpravidla stačí pro správnou kontrolu domácích zařízení a přijímání dat ze senzorů. Obliba této technologie souvisí také s faktem, že rozvod elektrické energie je již zaveden a to zpravidla s několika výstupy v každé místnosti domu.

Problémy s používáním PLC souvisí především s potřebou robustní napěťové modulace, velkým vlivem rušivých signálů, odrazy ve vedení, saturací signálu, útlumem signálu a s přeslechy. U starších rozvodů mají velký vliv na kvalitu přenosu také odlišné impedanční vlastnosti různých částí elektrického vedení. Mnohdy tak nastane chvilkový výpadek v síti z důvodu rozsynchronizování jednotlivých komunikujících stran a výjimečně vznikne i nekontrolovaný systémový pád, který lze vyřešit pouze ručním restartem všech modemů. [6]

Sítě X-10

Mezinárodní víceúčelový protokol X-10 je otevřeným standardem pro PLC. Bývá použit pro všechny aspekty inteligentního domu včetně zabezpečovacích a monitorovacích systémů. [6]

Komunikace probíhá modulováním 120 kHz signálu na vodič při každém průchodu napětí nulou. Díky tomu, že v tomto okamžiku prakticky neteče proud vodiči ani jedním směrem, je snažší eliminovat rušivé vlivy. Komunikace potom probíhá rychlostí 2 bitů za jednu periodu domácího napětí (tj. 0,1 kb/sec v Evropě).

Systémy používající technologie X-10 mají zpravidla jedno ovládací zařízení s vysílačem a několik přijímacích zařízení. Každý přístroj na sběrnici X-10 má vlastní identifikační číslo, které slouží k jeho adresaci.

X-10 je považován za spolehlivý protokol, i když bylo prokázáno, že určitá domácí zařízení mohou komunikaci pomocí této sběrnice rušit.

Jedná se o velmi levné řešení, nevyžadující instalaci žádných přebytečných rozvodů. Nevýhodou tohoto protokolu je jednosměrná komunikace master-slave, což znamená, že vysílací zařízení nemůže být informováno o úspěšnosti přenosu [6].

INSTEON

Novější technologie INSTEON vychází ze standardu X-10 a tvůrci z firmy SmartLabs ji vytvořili za účelem jeho nahrazení. Vyznačuje se nízkou složitostí i spotřebou. Na rozdíl od svého předchůdce používá kombinaci komunikace po napěťovém vodiči a rádiových signálů, spojených ve speciálním protokolu. Výhodou propojení těchto dvou systémů je především bezpečnost přenosu, nicméně jejich aplikace často naráží na řadu problémů. Instalace bývá zdlouhavá a vyžaduje složitou kalibraci. [6]

INSTEON využívá ke komunikaci signály na frekvenci 131,65 kHz s BPSK (Binary Phase Shift Keying) modulací [8] pomocí rozvodné sítě a zároveň signály na frekvenci 904 MHz s FSK (Frequency Shift Keying) modulací [8] přes rádiové vlny. Stejně jako u technologie X-10 dochází i zde k zasílání impulsů přes rozvodnou síť v okamžiku nulového napětí. Přenosová rychlost se pohybuje okolo 38 kb/sec. [6]

Narozdíl od X-10 je komunikace pomocí technologie INSTEON obousměrná, přičemž dochází k potvrzování přijatých a přeposílaných paketů. Pro jednoduchou komunikaci mají všechna zařízení nastavené hardwarové identifikační číslo, které slouží k jejich adresaci.

LonWorks

Mezi nejatraktivnější druh komunikace mezi zařízeními v inteligentním domě patří v dnešní době síťová platforma LonWorks vyvinutá firmou Echelon v devadesátých letech dvacátého století. Konkrétně se jedná o protokol LonTalk, navržený pro robustní komunikaci v ovládaných systémech. LonTalk zahrnuje standardy pro komunikaci pomocí PLC, kroucené dvojlinky, optických kabelů nebo rádiových vln. [18]

Stále více společností používá tuto architekturu v široké škále svých výrobků, přestože vysoká cena její aplikace zůstává stále primární překážkou (každý kontrolní bod sítě LonTalk standardně obsahuje tři mikrokontrolery, což tuto alternativu činí mnohem dražší, než použití například protokolu X-10). Obyčejně se u sítí LonTalk využívá hvězdicové topologie, i když je možné použít kruhovou, stromovou, hybridní a některé další. Výhodou LonWorks je dále především použití vrstveného OSI modelu, přičemž jsou pro různé úrovně komunikace použity jednotlivé mikročipy. První mikrokontrolér je využit pro práci s fyzickou a MAC vrstvou (1. a 2. vrstva). Druhý je zodpovědný za adresování a směrování komunikace (3. až 6. vrstva OSI modelu) a třetí

je vyhrazen pro zprovoznění samotné funkce „inteligentního“ výrobku (aplikační vrstva OSI modelu). [6],[18]

Na rozdíl od starších technologií je protokol LonTalk navržen tak, aby komunikace mezi řídicí entitou a řízeným zařízením probíhala obousměrně. Přestože v aplikaci LonTalk na kroucené dvojlince je možné dosáhnout rychlosti komunikace až 80 kbit/s, při použití PLC je dosaženo maximálně 3,6 kbit/s (pro 50 Hz). Přestože oficiální zdroje nespecifikují maximální možný dosah pro komunikaci přes PLC, ujišťují o možnosti komunikovat na „velké“ (řádově desítky metrů) vzdálenosti. [6]

PLC-BUS

Power Line Comunication Bus je výrobkem holandské firmy ATS [6]. Umožňuje obousměrnou komunikaci mezi všemi jeho zařízeními. Využívá technologie PPM [9] a dosahuje rychlostí komunikace až 0,2 kb/sec.

Přesto, že má tato technologie přinášet velmi dobré výsledky v oblasti spolehlivosti a bezpečnosti, bývají výzkumníci z oblasti aplikace PLC skeptičtí z důvodu nedostatku literatury popisující tento protokol [6].

Ostatní PLC

Mezi zbývající technologie využívající PLC patří například systémy EHS, KNX, S-Bus a UPB.

2.4.2 Externí komunikační vedení

Externí komunikační vedení zpravidla využívá jednoho, dvou nebo čtyř párů kroucených vodičů nebo koaxiálních kabelů, přičemž použití tohoto druhu propojení může být finančně nákladné. Výhodou je ale bezpečnost a jistota přenosu. Díky uzavřenému okruhu je více omezen způsob záměrného napadnutí inteligentního domu ze strany třetích osob; slabým článkem systému může být použití PLC nebo WI-FI.

Komunikace na fyzické vrstvě je specifická k použitému médiu (Ethernet, Homeplug, HomePNA, USB, FireWire). Vzhledem k faktu, že je na trhu velké množství konkurenčních protokolů, které mohou být vzájemně kompatibilní nebo naprosto odlišné, následuje výčet některých z nich, a jeden vybraný protokol je popsán podrobněji. Mezi protokoly používané při automatizaci domácnosti patří BACnet, Bus SCS, C-Bus, CEBus, KNX, LonWorks a S-Bus. [3]

BACnet

Komunikační protokol BACnet je navržený pro přímé ovládání domácích zařízení. Umožňuje používat několik druhů služeb jako jsou Who-Is, I-am, Who-Has, I-Have,

Read-Property, Write-Property, které umožňují komunikaci mezi jednotlivými objekty na sběrnici. [10]

2.4.3 Bezdrátové sítě

Bezdrátové propojení domácích zařízení přináší mimo jiné výrazně nižší náklady na zavedení. Dále umožňuje flexibilní umístění zařízení a povoluje jejich pohyb. Pomocí bezdrátové sítě mohou být ovládány například pohyblivé roboty nebo připojen přenosný ovládací panel. Bezdrátové sítě v domácnosti mohou sloužit k napojení jiných uživatelských zařízení jako jsou telefony nebo přenosné počítače, a mohou být použity k distribuci sítě internet.

Mezi nevýhody bezdrátových sítí lze považovat riziko jejich infiltrace zvenčí a vyšší náklady na jejich chod. Problémem může být také jejich vzájemné rušení při použití více různých standardů.

Dobře známé a často používané bezdrátové technologie jsou: Wi-Fi, Bluetooth, IrDa, FireWire, Z-Wave, GSM a ZigBee. [3] Přehled jejich parametrů a rychlostí se nachází v [47]. Ve zbytku této podpitoly budou nastíněny dva nejpoužívanější z nich.

Bluetooth

Bluetooth je zařízení pro sériovou komunikaci do maximální vzdálenosti 10 metrů s maximální šířkou pásma 1 Mbps. Skutečná propustnost se ale pohybuje kolem 0,7 Mbps (při použití asymetrického přenosu). Výhodou použití Bluetooth je nízká spotřeba elektrické energie. Nevýhodou je relativně krátký dosah (limitovány jsou především větší domy) a nízká přenosová rychlost pro potřeby streamování videa. Spolehlivost přenosu dat také není vysoká, protože Bluetooth sdílí své frekvenční pásmo s řadou jiných domácích výrobků. Šifrování a kontrola kvality přenosu sice možné jsou, ale musí být implementovány zvlášť. [21]

Wi-Fi

Přestože je spotřeba elektrické energie o něco vyšší než u jiných metod, splňuje tento standard všechny požadavky na propojení robotu s domem. Především jde o rychlost, bezpečnost a kvalitu přenosu. Z těchto důvodů je právě toto řešení preferované pro komunikační rozhraní robotu s domem.

2.5 Senzorické systémy

Pro dosažení schopnosti *inteligentně* reagovat na jakoukoliv změnu situace potřebuje mít dům nějakou formu zpětné vazby. Ani základní úkoly, jako je například regulace teploty v místnosti, nelze bezpečně bez použití zpětné vazby realizovat.

Světelné senzory

Světelné senzory slouží k měření intenzity a barvy světla. Pomáhají při správné volbě osvětlení, ať už se jedná o sílu světla nebo barevnou náladu pokoje.

Senzory pohybu, PIR a tříštění

Senzory pohybu jsou tradiční součástí zabezpečení proti vniknutí. Inteligentní dům může navíc tento druh snímačů použít k dohledání pozice obyvatele v domě, což lze následně využít například pro úsporu energie tak, že v neobsazených místnostech jsou zhasnuta světla [3].

PIR (*Passive InfraRed sensor*) měří energii vyzařované objekty v infračerveném spektru. Pomocí PIR senzoru je možné detekovat pohyb i v případě, kdy se v jeho dosahu kříží předměty o různé teplotě.

Senzor tříštění je velmi jednoduchý senzor zvuku. Na základě určitých zvukových shluků je schopný detekovat tříštění skla. Je vhodné jej použít jako součást zabezpečovacího systému nebo pro detekci krizové situace (obyvatel rozbije sklenici, což může být signál pro povolání uklízecích robotů).

Senzory teploty, tlaku a vlhkosti

Tyto senzory jsou neodmyslitelnou součástí každého inteligentního HVAC systému.

Magnetické senzory

Bývají součástí bezpečnostního systému. Používají se při detekci otevřených oken a dveří. Magnetické senzory lze také využít pro bezkontaktní ověření spuštění motorů. A v robotice jako kompasový senzor.

2.6 Mozek inteligentního domu

V samém jádru inteligentního domu bývá zpravidla domácí počítač neboli server. Ten je připojen ke všem externím zařízením a ovládá je. Vzhledem k faktu, že druhů a stupňů autonomnosti u externích zařízení je velké množství, existuje i velké množství způsobů ovládání těchto zařízení.

Komplexita domácího počítače roste s počtem zařízení k němu připojených a s požadavky na jeho inteligenci. Logicky platí, že čím větší má počítač přístup k periferiím, tím složitější bývá ovládací algoritmus.

Centrální počítač může mimo samotné automatizace domácnosti pracovat také jako zprostředkovatel sítí. Server je totiž s největší pravděpodobností již připojen k internetové síti a poskytuje v domácnosti rozvodné sítě LAN, Wi-Fi nebo Bluetooth. Je možné jej tedy použít například jako firewall, proxy, mail-server nebo FTP-server.

3 ROBOT V INTELIGENTNÍM DOMĚ

Bill Gates, spoluvlastník největší softwarové firmy světa Microsoft [19] (údaj převzatý z hodnocení časopisu Forbes pro rok 2012), ve svém článku A *Robot in Every Home* z roku 2006 [11] předpovídá, že stejně jako domácí počítače před třiceti lety, nachází se dnes robotika v zárodcích. Přestože Gates neuvádí přesné datum, kdy podle jeho názoru budou autonomní roboty skutečně v každé domácnosti, vyjadřuje názor, že se tomu tak neodvratně v dohledné době stane. Dnes, šest let po vydání tohoto článku, trend zavádění robotů do domácností pouze zrychlil a bude dále zrychlovat kvůli snižující se ceně hardwaru a zvyšující se konkurenci mezi výrobci.

Dnes již existuje celá řada pomocných, sociálních a vzdělávacích robotů [13], které jsou koncipovány pro pobyt v domech. Je ale podstatný rozdíl mezi robotem v domě pracujícím a robotem, který s domem spolupracuje. Spolupracující robot může nadále vykonávat ty činnosti, ke kterým byl primárně určen, ale může tak činit aniž by omezoval chod domu, a v některých případech může být využit domem při vykonávání složitějších úloh.

Spojení domu a autonomního robotu může vést k vzájemné symbióze, kdy obě strany těží z výhod toho druhého. Jako příklad lze uvést humanoidní sociální robot [12]. Ten může být naprogramován tak, aby pomocí senzorů v domě vyhledával obyvatele domácnosti a dělal jim společnost. Inteligentní dům potom může například použít kamerový systém robotu pro nahlédnutí do slepých míst.

V současné době jsou ve spojení s inteligentními domy aplikovány především roboty, vytvořené na pomoc handicapovaným a seniorům. [14]

3.1 Role domácích robotů

S rozvojem techniky přibývají stále další lidské úkony, které mohou roboty zastávat. Domácí roboty mohou v současnosti provádět mnoho většinou časově velmi náročných a monotónních prací. Rozsah úkolů, které provádí, sahá od prostého vysávání podlahy až po složité hlídání dětí. Zažité členění domácích robotů pracuje se třemi základními skupinami. Jsou to pracovní, sociální a vzdělávací roboty [13], přičemž existují i různé kombinace těchto skupin.

3.1.1 Robot služebník

Skupina robotů zastávající jednu nebo více lidských prací v domácnosti je bezkonkurenčně nejširší. Patří do ní stacionární i pohyblivé; humanoidní i diferenciálně řízené stroje. Jejich primárním účelem je ulehčit člověku od některé z domácích prací.

V této skupině najdeme roboty pro práci uvnitř i vně domu. Rozdíl mezi nimi je přirozeně v zaměření, ale především v robustnosti konstrukce a pořizovací ceně.

Robotická sekačka trávy

Robot zahradník většinou používá infračervené nebo fyzické překážky k vytyčení svého teritoria. V pravidelných intervalech projíždí zahradu a seká trávu na zadanou výšku. Vzhledem k tomu, že se jedná o venkovní robot, a že většinou pracuje na velké ploše, bývají tyto sekačky na trávu vybaveny GPS přijímači, solárními panely a dálkovým ovládáním. Pro eliminaci nebezpečí převrácení na nerovném terénu dokonce některé modely využívají vyvažovacích ramen [15].

U venkovních robotů obecně hrozí nebezpečí jejich odcizení. Nové modely proto bývají vybaveny různými bezpečnostními prvky jako jsou GPS lokátory, výstražné zařízení nebo dokovací stanice vysílající unikátní kód [16].

Výrobci robotických sekaček jsou například Zucchetti [15] nebo Husqvarna [16].

Robotický uklízeč bazénu

Tento druh robotu slouží k odstranění napadaného listí a jiného nepořádku ze dna bazénů. Některé modely pracují i na stěnách a vodním povrchu. Obecně se jedná o zařízení používající sání filtru bazénu k odstraňování nečistot a pro domácí automatizaci nejspíše nejsou vhodné vůbec.

Robotické vysavače a mopy

Robotické vysavače a mopy jsou zaměřeny na uklízení ploch uvnitř domů. Využívají algoritmů pro hledání trasy a pro mapování terénu. Některé modely je možné ovládat i bezdrátově nebo dokonce programovat. Pro vytyčení obhospodařované plochy využívají tato zařízení většinou infračervené majáky.

Robotické vysavače, stejně jako mnoho dalších robotů, mohou pracovat naprosto spolehlivě i bez inteligentního domu. Ale propojení s inteligentním domem může vytvořit oboustranně výhodnou vazbu.

Problematika robotických mopů se velice blízce týká problematiky robotických vysavačů. Praktický rozdíl je jen v rozsahu jejich působení. Jelikož se poměr ploch s koberci a ploch s jiným druhem povrchu liší dům od domu, jsou oba druhy robotů rovnocenní adepti pro práci v inteligentním domě.

Největší a nejrozšířenější výrobce těchto zařízení je firma iRobot [17]. Jejich řady Scooba i Roomba patří mezi špičku a jejich program Create umožňuje vytvoření vlastního robotu na jejich platformě.

Každý větší výrobce elektroniky zkusil štěstí na tomto poli. Mezi ně patří například Siemens, Elektrolux, Samsung a LG.

Robotický číšník

Na rozdíl od předešlých druhů robotů se robotický číšník neobejde bez toho, aniž by byl ovládán inteligentním domem nebo přímo člověkem. Jeho práce spočívá v pohybu po vytyčených trasách. Při detekci změny naložení nebo po příkazu odjede robot na předem určené místo „přeložit“.

Robotické číšníky je možné použít nejen k transportu potravin a pití, ale také k přenášení například zpráv ať již mluvených či psaných. Možnost využití tohoto druhu zařízení pro inteligentní dům je velice vysoká a v budoucnu se bude jednat o jeho nedílnou součást. V současnosti ale bohužel toto odvětví robotiky naráží na příliš mnoho problémů a žádný skutečně vhodný robotický číšník zatím nebyl zkonstruován.

3.1.2 Robot kamarád

Sociální roboty jsou jedním z velkých experimentů současné techniky. Úroveň umělé inteligence sice ještě nesahá dostatečně daleko k vytvoření skutečného kamaráda, nicméně již hodně úsilí bylo věnováno pochopení a aplikaci schopnosti pseudoemocí. [12]

Vývoj robotických přátel pro sociální interakci jde ruku v ruce s vývojem umělé inteligence. Na této problematice pracují přední světové robotické týmy.

Využívání tohoto druhu robotu pro inteligentní dům se v současnosti děje pouze jako pomocníka starým a handicapovaným lidem [14]. Většinu ostatních aplikací robotických kamarádů, jako jsou například pes Aibo (Sony) není možné použít specificky pro práci v inteligentním domě.

3.1.3 Robot učitel

Podobně jako sociální roboty je i oblast vzdělávacích strojů silně závislá na poli umělé inteligence. Autonomní jednání se u těchto strojů téměř nevyskytuje a robotičtí učitelé jsou převážně naprogramováni jen na plynulý přednes dané látky.

Využití vzdělávacích robotů v inteligentním domě se nyní zdá být plýtváním prostředků. V budoucnu tuto roli s největší pravděpodobností převezme jiný druh strojů.

3.1.4 Humanoidní robot

Nejuniverzálnější domácí robot je teoreticky takový, který je schopný zastoupit člověka v jakémkoliv aspektu. Jako řešení se nabízí humanoidní roboti. Ti jsou vyvíjeni například společností Honda (Asimo) [35] nebo v NASA (Robonaut) [36]. Přestože by využití humanoidního robotu v inteligentním domě bylo možné, současná vyspělost technologií a finanční nákladnost tomuto cíli zatím stojí v cestě.

3.2 Použití domácího robotu

Úkolem této práce je vybrat vhodné roboty pro použití v inteligentním domě. Cílem je zvolit takový robot, který zvládá autonomní provádění jistých činností potřebných v domě, a navíc má kvality, které je schopen dům využít i jinak.

Vhodnou volbou by mohl být například robotický vysavač se zabudovanou kamerou, ovladatelný pomocí bezdrátové sítě. Takový přístroj by ve spojení s inteligentním domem mohl pracovat vždy v nepřítomnosti obyvatel nebo v době, kdy by je nerušil při spánku či sledování televize. Interakce s člověkem by tedy probíhala až v okamžiku, kdy by dům signalizoval nutnost vyprázdnit koš vysavače. Pro uživatele by tak odpadla potřeba plánovat činnost stroje.

Dům by měl mít přehled o práci tohoto robotu, aby se dalo zamezit nechtěnému spuštění poplachu v době, kdy je dům střežen. Robot by také mohl být vysílán na potřebná místa v případě detekce náhlého znečištění podlahy.

V případě vzniku krizové situace jakou je například detekce požáru, únik tekutin nebo vniknutí zlodějů by dům kromě informování daných autorit vyslal také robot s kamerou kvůli rozšířenému vyhodnocení problému. Ten by ve spolupráci se systémem dveří a světel mohl sloužit jako nástroj pro vzdálené zjištění situace. Přestože některé sériově vyráběné modely již umožňují přistupovat k jejich interní kameře přes internet, úplné spojení s domem je myšlenka nová a její implementace bude součástí navazující práce.

3.3 Požadavky na autonomní robot

V předchozí kapitole byl navržen autonomní robot vhodný pro použití v domácnosti. Přestože samotný robot, a především jeho primární funkce, jsou otázkou výběru, některé vlastnosti musí robot nezbytně splňovat automaticky.

Komunikace

V první řadě je třeba, aby existovala možnost propojení robotu a inteligentního domu. Tento druh komunikace musí být schopný pracovat spolehlivě v reálném čase a nesmí existovat místo v dosahu domu, kde by se komunikace mohla přerušit. V případě výpadku, v důsledku krizové situace, se robot musí sám aktivně dopravit do dosahu bezdrátové komunikace.

Vhodné druhy bezdrátového propojení jsou Bluetooth a Wi-Fi. Jelikož první zmiňovaný nemusí mít dostatečný dosah, vhodnou kapacitu nebo rozšíření, je rozumné klást požadavek na podporu Wi-Fi. Zvolený robot se tedy musí být schopen připojit k domovnímu serveru přes zabezpečenou bezdrátovou síť Wi-Fi, která s největší pravděpodobností bude v budoucnu součástí každého inteligentního domu.

Senzory

Kromě schopnosti připojení k centrálnímu počítači je rozumné také požadovat možnost nějakého datového vstupu z robotu. Přestože některé druhy senzorů jsou u robotu samozřejmostí, ne všechny mají pro inteligentní dům význam. Například údaje z dotekového senzoru jsou pro dům téměř bezvýznamné.

Od zvoleného robotu je tedy rozumné požadovat, aby byl vybaven kamerou a aby údaje z ní byl schopen zprostředkovávat. Kamera musí také být namířena směrem od robotu do místnosti tak, aby byl pokrytý prostor co největší.

Své uplatnění nalezne také skupina senzorů využitelných pro zabezpečení. A nemusí se jednat pouze o zabezpeční proti vniknutí. Velkou roli mohou hrát například detektory kouře nebo vyplavení.

Pohyb

Z důvodu jednoduchosti a pohodlnosti programového řešení by měl mít vybraný robot diferenciálně řízený podvozek. Stroj s tímto druhem pohybu je navíc jednodušší simulovat pomocí generických knihoven diferenciálních podvozků v zvoleném simulátoru, jak je popsáno v kapitole 7.3.7.

Další funkcionalitou, která by měla být u vybraného robotu implementována, je možnost posunu na relativní lokaci. Tím se myslí, že robot dostane instrukci kam má přijet, a sám navrhne nejjednodušší trasu v závislosti na předem vytvořené mapě terénu. Tato funkce bývá často implementována právě u robotických vysavačů, které se po cyklu dobíjení vrací automaticky na pozici, kde přestaly vysávat.

Autonomnost

Nezbytnou vlastností vybraného robotu musí být to, že v případě přerušení spojení s centrálním serverem bude po co nejdelší dobu pokračovat v automatické činnosti. Optimálně musí být schopen dokončit zadanou úlohu a sám se vrátit na dobíjecí stanoviště.

Výhodou je, pokud zůstane většina výpočetní náročnosti, týkající se řešení zadaných úkolů, na robotu samotném. Znamená to, že server nebude muset ovládat robot přímo, ale prostřednictvím obecných instrukcí. Důvodem tohoto požadavku je, aby aplikace ovládající robot zabírala co nejméně strojového času domácího počítače. Domácí počítač bude nadále vyhodnocovat navrácená data, ale již nebude zatížen přesným výpočtem požadovaných výkonů jednotlivých motorů.

4 VOLBA VHODNÉHO ROBOTU

Úkolem této práce je prozkoumat trh a vybrat vhodný robot pro použití v inteligentním domě. Kritéria pro výběr robotu jsou popsána v kapitole 3.3. Na základě těchto požadavků, informací z oficiálních stránek výrobců a různých necitovaných recenzí nalezených na internetu, bylo vybráno několik možných přístrojů, které je v souladu se zadáním možné použít v inteligentním domě.

V této kapitole je nejprve provedena a zdůvodněna volba určitého typu robotického pomocníka v domácnosti. Poté je na základě korelace všech požadovaných prerekvizit vybráno a popsáno několik sériově vyráběných robotů, které je k vytyčenému úkolu možno použít. Konečně je navržen i jeden „udělej si sám“ balíček. V závěru kapitoly jsou shrnuta vybraná řešení.

4.1 Výběr specializace robotu

Jako zaměření robotu byl zvolen robot s funkcí vysavače. Moderní vysavače totiž poskytují nejvíce z požadovaných prerekvizit a je možné je pořídit v relativně přijatelné cenové relaci. Funkce vysavače je navíc požadována v každé domácnosti, inteligentní dům nevyjímaje.

Všechny sériově vyráběné robotické vysavače používají diferenciální pohon a je možné je snadno programovat. Podmínka jednoduchého ovládání je tedy splněna. Většinu robotických vysavačů je navíc možné ovládat dálkově a ty z nich, které mají k dispozici kameru, mají většinou i Wi-Fi připojení.

Vysavače navíc obsahují laserové nebo infračervené senzory, které dokáže inteligentní dům detekovat, a tím zjistit přesnou pozici robotu. Interní tvorba map a následný přístup k nim je u novějších aplikací již dnes standardem.

Jelikož jsou vysavače primárně vybaveny pro autonomní chod, obsahují dostačující výpočetní výkon pro schopnost orientace v terénu s minimálním vytížením centrálního počítače. Možnost automatického návratu do dobíjecí stanice je také samozřejmostí.

Posledním faktorem, který ovlivnil výběr druhu robotu, je fakt, že na trhu existuje široká řada výrobků a konkurence pozitivně ovlivňuje jejich cenu. Například ve srovnání s robotickými čísníky se stejnými schopnostmi je cena vysavačů daleko nižší.

4.2 Dostupné výrobky

Přestože je v současnosti na trhu mnoho konkurenčních výrobků, jen roboti některých firem splňují požadované prerekvizity a zároveň mají dobré reference a kvalitu. V následujících podkapitolách jsou roboti řazeni podle výrobce a ke každému je uveden krátký popis včetně orientační ceny (ceny jsou platné k datu odevzdání této práce).

Technické informace jsou přejaty ze stránek výrobce a některé informace pochází z necitovaných zdrojů (převážně z fór a recenzí k výrobkům).

4.2.1 Výrobce LG

Přestože firma LG vstoupila na trh s robotickými vysavači relativně nedávno, produkuje nyní jedny z nejlepších výrobků. Je to právě řada robotů Hom-Bot.

Všechny nové výrobky LG jsou stavěny na technologii Thinq, která umožňuje jejich propojení pomocí domácí sítě za použití Wi-Fi. Tento fakt dává příslib možnosti tvorby vlastních aplikací pro roboty řady Hom-Bot. Přestože již LG vydalo několik dokumentů k funkčnosti a k použití této sítě, její implementace závisí na použití některého z výrobků firmy LG.

LG Hom-Bot VR5900

LG Hom-Bot LrV5900 patří k nadstandardu mezi robotickými vysavači. Jedná se již o třetí a zatím marketingově nejúspěšnější generaci vysavačů od LG. Jeho součástí je nástavec pro mopování a pořizuje mapu prostorů, ve kterých se pohybuje pomocí technologie DualEye. Samozřejmostí je použití technologie LG Thinq, která umožňuje propojení Hom-Botu s ostatními domácími zařízeními pomocí Wi-Fi.

Obr 4.1: LG VR5900 [44]

Robot je oficiálně dostupný na korejském, americkém a evropském trhu s cenou pohybující se okolo 800$. Rozměry tohoto robotu jsou 36 cm v průměru a 9 cm na výšku s hmotností 3,2 kg. Hluk přístroje při vysávání je pouze 60 dB. Splněny jsou i podmínky programovatelnosti a možnost propojení s inteligentním domem, přičemž součástí robotu je dokovací stanice.

Senzorický systém obsahuje přední nárazník, ultrazvukový senzor a čidla poškození.

LG Hom-Bot VR6800 (Hom-Bot 2.0)

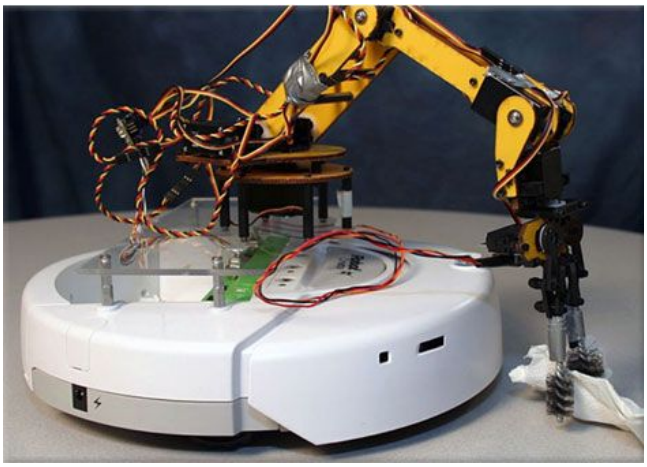
LG Hom-Bot 2.0 je čtvrtou generací robotů Hom-Bot. Mapuje okolí pomocí technologie TripleEye a oproti předchozí verzi navíc obsahuje detekci nízkých průchodů. Jeho rozměry jsou shodné s předchozí verzí. Cena výrobku, který je dostupný pouze na evropském a korejském trhu, se pohybuje okolo 750$.

Obr 4.2: LG VR6800 [44]

Na trh tento výrobek vstoupil v roce 2011. Novou funkcí, představenou tímto modelem, je například možnost ovládání hlasem. Hom-Bot 2.0 obsahuje také reproduktory a mikrofon, takže pomocí něj může probíhat obousměrná komunikace.

4.2.2 Výrobce iRobot

Rozšiřitelný balíček iRobot Create dovoluje postavit vlastní robot na platformě vysavačů Roomba. Jeho primárním účelem je provádění experimentů a další vzdělávání v oblasti robotiky.



Obr 4.3: iRobot Create [17]

Robot samotný sice nemá funkci vysavače, ale lze na něm testovat změny, které je posléze možné aplikovat na skutečný robot Roomba. Platforma obsahuje pouze

komunikační port pro ovládání základních funkcí robotu, jako jsou standardní senzory, a pohon. Stejné rozhraní lze nalézt i u robotů Roomba. Pomocí rozšíření jde na platformu robotu přidat i Wi-Fi modul a další požadované příslušenství. Na obrázku 4.3 je příklad projektu s robotickým ramenem pro sběr odpadků.

Cena základního balíčku, obsahujícího platformu, příkazový modul, dvě virtuální stěny, dokovací stanici a další obvyklé příslušenství přijde na zhruba 300$. Zbytek zařízení včetně programového rozhraní si ale musí uživatel dodat sám. K robotu Create existují Open Source knihovny pro práci s vestavěnými senzory a prvky robotu.

4.3 Vybrané řešení

Při rešerši a volbě vhodného autonomního robotu byly zvažovány i roboty firem Samsung, Chinavasion, Elektrolux, Siemens a další, všechny však byly následně zamítnuty. Některé z nich nevyhovovaly svou specifikací a některé byly zavrženy z důvodu špatného hodnocení uživatelů.

Jako vhodné roboty pro použití v inteligentním domě byly shledány LG Hom-Bot 2.0 a robotický balíček iRobot Create. Přičemž programování robotů firmy LG stojí pouze na ujištění výrobce o existenci použitelných knihoven. V opačném případě by bylo nutné všechny podpůrné knihovny napsat znovu. Přestože existují videa fungujících výrobků, bez přístupu ke skutečnému hardware není možné ověřit jejich autenticitu.

Za těchto podmínek byl vybrán jako nejvhodnější robot firmy iRobot. Tato firma vystupuje na trhu s výrobky několika druhů. Je to například robotický vysavač Roomba, robotický mop Scooba, leštič podlahy Mint, čistič bazénů Verro, vymetač okapů Looj a dokonce i řada robotů pro vojenské použití, například SUGV a Warrior [17]. Výhodou je podobné programové rozhraní pro většinu z těchto robotů [20]. Případná reimplementace vytvořených kódů z jednoho výrobku na druhý by tedy byla výrazně snažší.

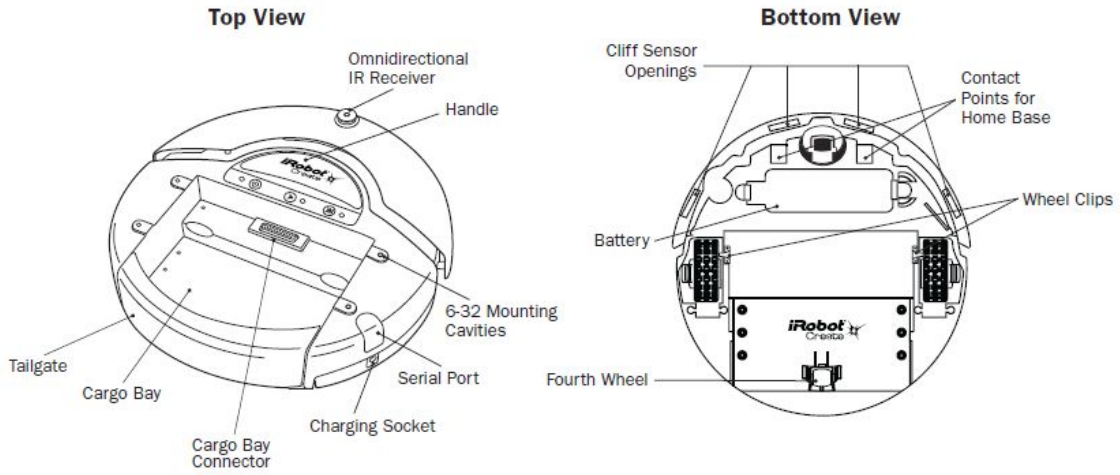
Platforma robotu Create je velice podobná platformě vysavače Roomba. Robotický vysavač iRobot Roomba je jedním z nejprodávanějších na trhu [21]. Nejedná se sice o nejlevnější řešení, ale robot Create je v cenové relaci srovnatelný s jinými běžnými roboty podobného zaměření. Na rozdíl od nich je ale možné jej programovat v široké škále vývojových prostředí. Patří mezi ně například prostředí pro vývoj robotických aplikací MRDS (*Microsoft Robotics Developer Studio 4*), které bude použito pro programování a simulaci robotu (kapitola 7.3). Nevýhodou tohoto robotu je fakt, že oproti jeho funkčnímu protějšku není vybaven žádným sacím ani jiným podobným zařízením, nicméně tento nedostatek nepředstavuje žádnou překážku v jeho využití pro potřeby této práce.

5 SENZORY A HARDWARE

Tato kapitola se zabývá různými druhy senzorů robotu Create, dále senzory, které je možno k robotu přidat pro zlepšení jeho funkčnosti, a senzory běžnými pro inteligentní dům, které mohou být při ovládání robotu využity. Následně jsou probrány možnosti pro realizaci interakce robot-dům.

5.1 Standardní hardware zvoleného robotu

Create obsahuje řadu vnitřních i vnějších senzorů. Většina těchto senzorů je stejná nebo velmi podobná těm, které jsou implementované v jiných robotech společnosti iRobot. Jedná se například o IR senzory nebo o enkodéry kol. Na Obr 5.1 je vyobrazeno vnější zpracování robotu [22]. V následujících podkapitolách bude popsána většina částí referovaných v tomto obrázku.



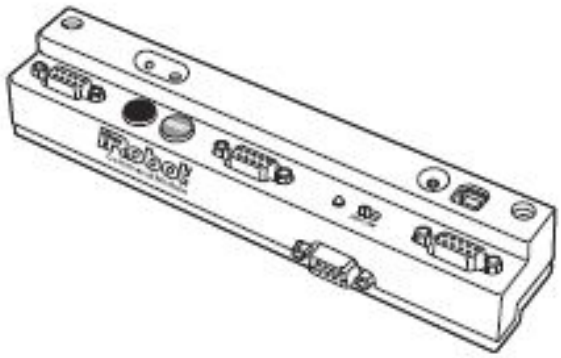
Obr 5.1: Vnější rozložení robotu Create [23]

Informace v této podkapitole vychází téměř všechny ze specifikace robotu a návodů publikovaných výrobcem, jedná se o reference [22] a [23].

5.1.1 Cargo Bay

V části *Cargo Bay* se u robotu Create nachází prostor vhodný pro umístění různého specializovaného hardwaru. Je opatřena konektorem typu DB-25 [24]. U robotických vysavačů Roomba, ze kterých Create konstrukčně vychází, tento prostor a konektor není. Místo toho jsou v zde umístěny propriety pro úklid. Konektor DB-25 (*Cargo Bay Connector*) má 25 pinů, z nichž jsou 4 digitální a analogové vstupy, 3 digitální výstupy, 3 vysokoproudové výstupy (vhodné pro ovládání dodaných motorů nebo osvětlení), indikaci stavu baterie, sériový port a 5V napájení.

Ke konektoru DB-25 je možné připojit specializovaný iRobot Command Module Obr 5.2. Ten se musí zakoupit zvlášť a je vhodný pro programování robotu v jazyce C/C++. Obsahuje 4 DB-9 konektory, vhodné pro snadné rozšíření robotu o řadu senzorů.

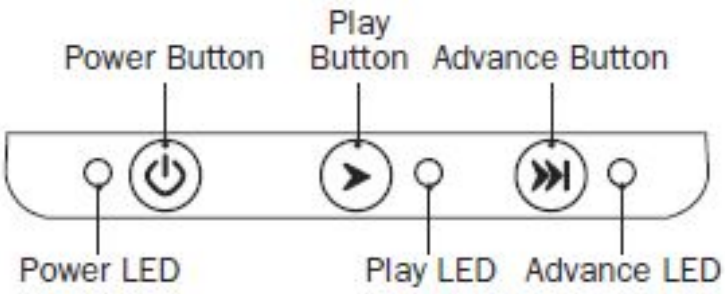


Obr 5.2: iRobot Command Module [23]

Tyto DB-9 konektory (sériové konektory) je možné využít pro standardní komunikaci přes sériovou linku nebo jako analogové či digitální převodníky. Modul je výpočetní kapacitou shodný s některými zařízeními Arduino.

Zajímavostí *Cargo Bay* je možnost připojení kolečka zespodu robotu. To slouží k lepší stabilitě robotu v případě, že je těžiště posunuto směrem za osu postranních kol vlivem vysoké hmotnosti zařízení, zasazených v oblasti *Cargo Bay*. Toto dodatečné kolečko bývá součástí zakoupeného balíčku iRobot Create.

5.1.2 Ovládací tlačítka



Obr 5.3: Ovládací tlačítka iRobot Create [23]

Na povrchu robotu se nachází malý kontrolní panel se třemi tlačítky Obr 5.3. Při běžné konfiguraci robotu je možné jej využít k omezenému ovládání robotu. Jde například o spouštění vysávacího cyklu nebo přehrávání nakonfigurovaných melodií. Výstup dvou pravých tlačítek je možné programově zachytit a využít pro ovládání robotu [23].

Dále se na tomto panelu nachází tři diody pro indikaci různých stavů. Například indikace nabití robotu. Na této diodě je možné programově nastavit různou intenzitu a barvu světla. Ostatní dvě LED diody jsou zelené a lze je pouze přepnout do stavu zapnuto.

5.1.3 Dotekové senzory

Přední část robotu Create je vybavena dvěma dotekovými senzory. Ty jsou připojeny na přední blok robotu a mohou být použity k detekci nárazu do překážky. Fakt, že jsou aplikovány v páru, umožňuje indikaci přibližné polohy překážky. Detekovaná překážka se nachází podle kombinace stisknutých senzorů buď vlevo, vpravo nebo uprostřed před robotem.

5.1.4 Detektor převisu

Na předním bloku robotu jsou umístěny 4 detektory převisu. Jedná se o jednoduché senzory vzdálenosti. Tyto snímače mohou sloužit k detekci propasti, nacházející se v oblasti před robotem, jako jsou schody nebo okraj stolu či podložky, po které se robot pohybuje. Dále mohou být použity pro detekci zvednutí robotu například pokud s robotem manipuluje uživatel.

5.1.5 Vícesměrové IR zařízení

Na předním nárazníku robotu se nachází *omnidirekcionální IR receiver*. Ten bývá použit k zachycení příkazů dálkového ovládání robotu.

Toto zařízení bývá také použito pro detekci dobíjecího stanoviště. Stanoviště vysílá do stran světlo v různých spektrech, podle kterých je robot schopný určit přibližnou polohu vůči stanovišti. Určit přesnou vzdálenost od stanoviště pomocí tohoto senzoru schopen není. Robot je standardně vybaven funkcí *„force dock“* [23], která nalezne a provede připojení robotu k dobíjecí základně. V případě, že se robot nenávratně ztratí, jedná se o poslední možnost sebelokalizace (s velice vysokou pravděpodobností úspěchu).

Další možností využití tohoto senzoru je detekce tzv. virtuální stěny, což je přístroj, vysílající IR paprsky vymezující robotu hranici prostoru, kam smí cestovat v daném směru.

5.1.6 Senzor stěny

V pravé části předního nárazníku se nachází takzvaný senzor stěny. Jedná se o infračervený vysílač a přijímač. Ten bývá použit pro určení vzdálenosti od překážky,

nacházející se napravo od robotu, což slouží k zpřesnění navádění robotu podél stěn. Detekce vzdálenosti tímto senzorem je velice nepřesná, protože závisí na barvě a odrazivosti světla od překážky. Senzor tak bývá spíše než k určení vzdálenosti stěny použit pro udržování požadované vzdálenosti od ní na základě předchozích měření.

5.1.7 Prověšení kola

Všechna tři standardní kola jsou vybavena snímačem detekujícím jejich prověšení. To nastane v případě, pokud robot přejede přes hranu plochy, po které se pohybuje, nebo pokud je zvednut ze země.

5.1.8 Enkodéry kol

Pravé a levé pohonné kolo je opatřeno rotačním enkodérem. Pomocí těchto senzorů robot zjišťuje pootočení svých kol. Nepříjemnou zvláštností je fakt, že příkazové rozhraní robotu neumožňuje číst přesné hodnoty z těchto senzorů. Namísto toho je možno z robotu vyžádat informaci o vzdálenosti a natočení robotu. Při každém vyžádání se tyto informace ve vnitřních registrech robotu resetují. Z těchto dvou informací je poté možno snadno vypočítat potřebné pootočení kol. Tento postup je dále popsán v kapitole 6.1.

Nevýhodou enkodérů kol je fakt, že se jedná o jednořadé enkodéry. Z toho plyne praktická nemožnost získat přesnou výchylku kol, pokud je s robotem manipulováno z vnější. Robot vrací správné hodnoty pouze pokud na něm právě probíhá příkaz pohybu.

5.1.9 Snímání výkonu

Create obsahuje několik senzorů pro snímání různých charakteristik, souvisejících s elektrickým výkonem. Prvním z nich je senzor nabití baterie. Ten udává stav baterie a předpokládanou výdrž. Při procesu dokování je také možné zjistit stav dobíjení baterie. Další senzory sledují proudové přetížení levého a pravého motoru. To slouží k ochraně motorů před nechtěným poškozením. Nakonec obsahuje Create snímač přetížení vnitřního napěťového převodníku.

5.1.10 Výstupní zařízení

Již bylo zmíněno, že robot Create je řízený diferenciálně. Obsahuje tedy dva relativně silné motory pro ovládání krajních kol. Dále robot obsahuje malý reproduktor, pomocí kterého je možné přehrávat některé předdefinované a uložené zvuky. Přehrávání jiných než předem uložených zvuků není standardně možné.

5.2 Přidaný hardware

Tato podkapitola se zabývá možností implementace některých senzorů a zařízení, které nejsou standardně součástí robotu. Tyto senzory je možné připojit k robotu buď pomocí jeho vícevstupového portu, nebo je implementovat zvlášť jako modul a ovládat je nezávisle na robotu. Druhé řešení má tu výhodu, že následně je možné takový modul implementovat i na jiné roboty firmy iRobot, jako je například Roomba, bez nutnosti rozsáhlejších změn. Z tohoto důvodu bude implementace veškerých venkovních sonzorů připojena k řídicí jednotce robotu a nikoliv k robotu samotnému

I když nemusí být všechny zmiňované senzory v robotu implementovány, jsou v následujících podkapitolách alespoň zmíněny společně s možnostmi jejich použití.

5.2.1 Mikrokontroler

V závislosti na množství úloh prováděné samotným robotem je třeba zvolit vhodný kontroler připojený k robotu. Základní požadavek na toto zařízení je podpora komunikace s robotem pomocí sériové linky. Dále je nutná komunikace robotu s inteligentním domem, která musí být z důvodu mobility robotu vedena pomocí bezdrátového zařízení. Vhodnými kandidáty pro bezdrátovou komunikaci jsou Bluetooth, Wi-Fi, FireWire a ZigBee (kapitola 2.4.3). Poslední dva nelze snadno využít se zvoleným hardware (kapitola 4.3), a proto jsou zde pouze zmíněny. Další požadavek na mikrokontroler je schopnost přenášet streamované video z kamery robotu. V neposlední řadě by měl být kontroler vybaven převodníky pro čtení dat z přidaných senzorů. Velikost samotného mikrokontroleru s přidanými moduly by neměla přesáhnout rozměry robotu a v ideálním případě by se měl celý aparát vejít do hardwarové přihrádky robotu. [1], [3], [21]

Vzhledem k faktu, že je robot Create vybaven akumulátorovou baterií s vysokou kapacitou, je omezení výkonu řídicího obvodu robotu malé. Z tohoto důvodu je proponováno řešení bezdrátového spojení pomocí Wi-Fi. Wi-Fi má podstatně větší dosah, než ostatní zmiňované alternativy, a navíc podporuje mnohem rychlejší a šifrovaný přenos.

V případě, že bude všechen výpočetní výkon ležet na inteligentním domě, je možné využít některý z malých mikrokontrolerů. Možnými alternativami jsou například Arduino Due v kombinaci s Arduino WiFi Shield. Nevýhodou této platformy ale zůstává fakt, že pro propojení s kamerou nemá čip dostatečný výkon pro streamování videa. Přenos videa by tedy musel být veden externě (například pomocí Raspberry Pi). [25]

Jiná alternativa pro vytvoření embeded minimálního systému je použití platformy Electric Imp. Jedná se o procesor s Wi-Fi modulem velikosti SD karty. Naneštěstí podobně jako u Arduina je přenos videa v současné době značně omezen (20 fps v minimálním rozlišení). [26]

Za předpokladu, že bude robot koncipován pro vyšší výpočetní zátěž, kterou by jinak musel provádět domácí server, je možné namísto mikrokontroleru použít složitější zařízení na straně robotu. V takovém případě by možnou alternativou mohly být některé složitější chipsety a malé počítače s vlastním operačním systémem. Možností by mohlo být vyžití vyřazeného nebo částečně nefunkčního notebooku/netbooku. Jasnými kandidáty pro operační systém jsou Linux a, z důvodu lepší dostupnosti a kompatibility driverů, některá z distribucí Windows. Výhodou takovéto koncepce je nižší náročnost na vytvořený kód (zvolený OS bude obsahovat knihovny pro přístup k Wi-Fi, kameře, atd.) a menší datový tok (signály ze senzorů jsou vyhodnocovány lokálně a je tedy například přenášen pouze údaj o poloze namísto všech čtení enkodérů kol). Nevýhodou tohoto řešení je, že kontrolní obvod situovaný na robotu bude celkem velký. Vhodným hardwarem je v zásadě kterýkoliv AMD nebo Intel chipset standardu nano-ITX nebo pico-ITX, s připojenou USB kamerou a Wi-Fi. Jiným řešením je úprava inteligentního telefonu nebo PDA. [27]

5.2.2 Kamera

Uživatel může pomocí robotu vybaveného kamerou získávat okamžité snímky, a to i z úhlů a pozic, ve kterých nejsou žádné kamery či jiné senzory inteligentního domu.

Nevýhodou použití kamery ve spojení s technologií Bluetooth je to, že při současném pohybu robotu (a třeba i snímání zvuku) může být překročena kapacita bezdrátového spojení. [25] Kamera vhodná pro toto použití je například PK 465 DN (Obr 5.4).

Obr 5.4: PBC PK 465 DN [45]

5.2.3 Mikrofon

Senzor pro nahrávání zvuku by měl být nedílnou součástí robotu. Může být využit jak pro záznam zvuku v místech, kde inteligentní dům žádný senzor doposud nemá, tak pro přijímání zvukových příkazů. Důležitý požadavek na mikrofon je, aby byl schopný určit

směr příchozího zvuku. K tomu je nejsnažší využít pole směrových mikrofonů. Postačující jsou tři mikrofony. Detekce směru potom může probíhat na principu diferenčního srovnání intenzity zvuku ze všech vstupů nebo na principu fázového posuvu na jednotlivých mikrofonech.

5.2.4 Ultrazvukový senzor

Ultrazvukový senzor pro měření vzdálenosti k překážkám se sice může jevit jako zbytečný, protože robot již obsahuje sám o sobě jiné detektory překážek, ale jeho použití může být spojeno s funkčností nastíněnou v podkapitole 5.2.3 - Mikrofon. Nebo při použití jiného robotu, který takovými senzory nedisponuje.

5.2.5 Senzor náklonu

Pro přesnější vyhodnocení pozice robotu je vhodné získávat informace o jeho náklonu. Tato data mohou být použita například pro určení, zda se robot nachází na některé spojovací rampě domu, nebo pro detekci zvednutí robotu ze země.

Gyroskop

Podobně, jako bylo zmíněno v minulé podkapitole, je pro měření změny náklonu oproti referenčnímu stavu možné použít gyroskop. Nevýhodou tohoto řešení je chyba způsobená opakováním měření. K měření absolutní orientace vůči zemi je sice gyroskop vhodný, ale vzhledem k tomu, že nezprostředkovává informace o pohybu, nejedná se o nejlepší senzor pro absolutní lokalizaci.

Akcelerometr

Pro měření přesného náklonu robotu bez kumulativní chyby může být použit akcelerometrický senzor. Ten na základě zjištění vektoru působení gravitačního zrychlení země dovede určit přesný náklon robotu v každém daném okamžiku. Ve spojení se snímačem náklonu (probraným v předchozí podkapitole) mohou být výstupy tříosého akcelerometru, kompasu a gyroskopu použity k velice přesné lokalizaci robotu [31].

5.2.6 Senzor natočení

Pro přesnou lokalizaci robotu v domě je vhodné znát aktuální úhel natočení vůči referenční hodnotě. Přestože tuto funkcionalitu již obstarávají enkodéry kol robotu, při získávání úhlu natočení dochází k integraci chyb způsobených nepřesností enkodérů a prokluzem kol. Ve standardní implementaci není vůbec řešena situace, kdy je robot zvednut ze země.

Kompas

Vhodným senzorem pro detekci natočení robotu je elektronický kompas. Nevýhodou tohoto senzoru je případná nepřesnost, způsobená elektromagnetickým polem motorů robotu. Výhodou je to, že při použití kompasu získává robot hodnotu natočení nezatíženou kumulativní chybou enkodéru.

5.2.7 Satelitní navigace

Zejména zařízení standardů GPS, Galileo a GLONASS. Ty jsou zde ale uvedeny pouze pro úplnost. Praktické využití pro navigaci v interiéru inteligentního domu toto řešení neposkytuje. Přesnost této navigace se totiž ve venkovním prostoru pohybuje v řádech metrů a fakt, že robot je uvnitř uzavřené budovy, ji pouze zhoršuje.

5.2.8 Detekce stavu prostředí

Jedním z hlavních úkolů autonomního robotu v inteligentních domech, je detekce nestandardních a krizových situací. Mezi nestandardní situace patří například požár nebo vniknutí cizí osoby. Pro jejich detekci je možné využít následující tři typy senzorů.

Kouř

Kouř může být prvním příznakem vzniklého požáru.

Teplota

V případě, že se k robotu nedostane kouř, může zvýšená teplota upozornit na oheň. Inteligentní dům bude samozřejmě měření teploty provádět nezávisle na robotu, ale v případě zjištění vysoké teploty může robot vyrazit směrem k místu incidentu a opakovaným měřením upřesnit stav. Zvolený iRobot Create obsahuje interní měřič teploty.

Pro detekci vniknutí cizích osob se běžně používají pyrodetektory. Senzor připojený přímo na robot může být použit při prohledávání domu. Jeho použití má také tu výhodu, že se pro přesnou detekci nemusí robot příliš přibližovat k potenciálně nebezpečnému místu.

5.3 Hardware inteligentního domu

Komplexní senzorický systém běžného inteligentního domu je při použití mobilního robotu nutno rozšířit o hardware potřebný k jeho přesné lokalizaci a práci s ním. Dále je třeba zajistit dostatečné výpočetní schopnosti inteligentního domu a vhodný komunikační kanál.

5.3.1 Komunikační rozhraní

Nejdůležitějším prvkem inteligentního domu ve spojení s autonomním robotem je jeho komunikační rozhraní s robotem. Možnými kandidáty na komunikační protokol jsou Bluetooth a Wi-Fi, jak bylo zmíněno v podkapitole 2.4.3.

5.3.2 Hardware pro lokalizaci

Důležitou funkcí domu ve spolupráci s robotem je jeho asistence při lokalizaci robotu. Lokalizace může probíhat více způsoby, které jsou teoreticky popsány v následujících odrážkách.

- **Infračervené, Ultrafialové a Ultrazvukové emitory**

Za předpokladu, že je robot vybaven senzory daného vlnění, mohou být emitory těchto typů rozmístěny v prostorách inteligentního domu. Robot si při potřebě lokalizace vyžádá jejich spuštění a ty se na krátkou dobu postupně aktivují. Robot poté na základě jejich zaměření určí buď svoji přesnou polohu (například pomocí triangulace) nebo alespoň určitou oblast domu ve které se nachází.

- **Pasivní prvky**

Pro lokalizaci robotu mohou být využity různé druhy magnetických nebo jiných pasivních značek uvnitř domu. Možností jsou různé druhy natištěných vzorů v široké škále barevných spekter (člověku viditelné i neviditelné) a tvarů.

- **Domácí kamery**

V hypotetickém případě, kdy by byl inteligentní dům vybaven množstvím kamer, může být jeho pozice přesně stanovena na základě jejich výstupu při použití počítačového vidění. Robot je v tomto případě vhodné vybavit například sadou infračervených LED emitorů, pomocí kterých bude v potřebný okamžik pro dům snadno viditelný. Nevýhodou tohoto přístupu je vysoká výpočetní náročnost a složitá kalibrace takového systému (nepatrné pootočení kamery obyvatelem domu může způsobit velkou chybu při lokalizaci). Navíc, jak již bylo zmíněno v předchozí kapitole, není běžné vybavovat každou místnost inteligentního domu kamerou, aby nebyl příliš narušen pocit soukromí obyvatel domu.

6 LOKALIZACE A PLÁNOVÁNÍ TRASY

Pro správnou funkci robotu v inteligentním domě je potřeba znát přesné údaje o jeho aktuální poloze. V této kapitole jsou popsány možné způsoby lokalizace robotu a návrh orientace robotu s pomocí mapy prostředí.

6.1 Dead reckoning

Název tohoto druhu určení polohy robotu vznikl zkrácením slov deduced (dedukovaný, neboli určený původní polohou a informací o její změně) a recognition (rozpoznání). Jak název napovídá, vychází tento přístup z apriorní znalosti polohy robotu a z informací o její změně. Nevýhodou tohoto způsobu lokalizace je, že je zatížen kumulativní chybou. To znamená, že malé chyby v odhadu změny pozice robotu se sčítají a po delší době bez kalibrace polohy mohou dávat velice zkreslený údaj. [28]

Enkodéry diferenciálních motorů

Za předpokladu, že je známa změna natočení kol diferenciálního podvozku, je možné pomocí následujících tří jednoduchých rovnic vypočítat změnu polohy robotu. [29] Výpočet zanedbává prokluz mezi koly robotu a podložkou.

Pro změnu natočení robotu vůči výchozí poloze (poloha při resetování enkodérů) platí:

$$\Delta\theta = \frac{2\pi}{T}\frac{r}{d}(T_r - T_l)[rad] \tag{1}$$

kde r ... *poloměr kola robotu,*
d ... *vzdálenost mezi koly robotu,*
T ... *počet tiků enkodéru při jednom úplném otočení kola,*
$\Delta\Theta$ ... *změna natočení robotu v radiánech, kladné hodnoty – otočení doprava,*
T_l ... *počet tiků enkodéru levého kola,*
T_r ... *počet tiků enkodéru pravého kola.*

$$\Delta x = \frac{\pi r}{T}\cos(\theta)(T_l + T_r)$$
$$\Delta y = \frac{\pi r}{T}\sin(\theta)(T_l + T_r) \tag{2}$$

kde Θ ... *aktuální natočení robotu vzhledem k souřadné ose (NE změna natočení!),*
Δx ... *změna polohy robotu v ose x souřadné soustavy (jednotka je jako u r),*
Δy ... *změna polohy robotu v ose y souřadné soustavy (jednotka je jako u r).*

Pomocí standardních příkazů robotu Create bohužel nelze zjistit změnu natočení enkodérů. Namísto toho je možné vyžádat pomocí příkazu *Angle* změnu natočení robotu od posledního volání robotu ve stupních (rovnice 1 převedená z radiánů na stupně). A pomocí příkazu *Distance* údaj o ujeté vzdálenosti v milimetrech.[23] Pro převedení ujeté vzdálenosti ze stupňů na rozdíl v souřadné ose můžeme použít následující rovnice, které vychází z kosinové věty:

$$\Delta x = D\cos(\theta)[mm]$$
$$\Delta y = D\sin(\theta)[mm] \quad (3)$$

kde D ... ujetá vzdálenost

Po srovnání rovnic 2 a 3 je patrné, že výstupní údaj funkce *Distance* je součet natočení obou kol v milimetrech, vydělený dvěma. A výstup z funkce *Angle* je dvojnásobek součtu výchylek obou kol vydělený průměrem robotu, převedený z radiánů na stupně. Nevýhodou tohoto přístupu je, že se ve výsledku jedná pouze o aproximaci pohybu po křivce úsečkou. Nicméně, za předpokladu dodržení podmínek popsaných v [37], je možné počítat s minimální chybou. Jedním z těchto kritérií je zejména nutnost redukovat maximální natočení mezi dvěma polohami na méně nežli 10 stupňů.

Akcelerometry, gyroskopy a kompas

Pomocí dynamického snímání informací z těchto senzorů je možné přesně vypočítat změnu polohy robotu, a to i v případě, že robot ztratí kontakt s podložkou. Praktický příklad s důkladným vysvětlením principu je v [28] a [31].

6.1.1 Chyby metody DeadReckoning

Lokalizace pomocí metody Dead-Reckoning je zatížena dvěma zdroji chyb [32]. Ty lze označit za systematické a nesystematické. Vzhledem k faktu, že jsou oba zdroje chyb způsobeny nekonstantními vlivy, je nutné výpočet jejich hodnot (kalibraci) provést pro každý použitý robot samostatně (velikost chyb se liší kus od kusu). Tyto zdroje chyby lokalizace jsou probrány v následujících podkapitolách.

Systematické chyby

Systematické chyby jsou dány nepřesností a špatnou kvalitou hardware diferenciálního podvozku. Běžně se hodnota těchto chyb v průběhu navádění nemění a lze je minimalizovat aproximací výpočtu polohy. Jedním z možných způsobů jejich výpočtu je metoda *UMBmark* [32]. Metoda zvolená v této práci z ní koncepčně vychází, ale je komplikovanější (podkapitola 8.3). Zdroje systematických chyb metody dead-reckoning jsou:

1. Různé rozměry kol diferenciálního podvozku.
2. Položení kol mimo jednotnou osu.
3. Směr pohybu kol není rovnoběžný.
4. Nedostatečné rozlišení nebo vzorkovací perioda enkodérů kol.

Nesystematické chyby

Nesystematické chyby vznikají vlivem prostředí, ve kterém se robot pohybuje. Na rozdíl od systematických chyb je jejich výskyt nepředpovídatelný a nelineární. Obecně je jejich detekce velmi složitá a jejich výpočet je tedy omezen na výpočet pravděpodobnosti jejich vzniku v závislosti na ujeté vzdálenosti. Zdroje nesystematických chyb jsou především:

1. Nerovnosti podložky, po které se diferenciální podvozek pohybuje.
2. Prokluz kol.

Hodnotu nesystematické chyby lze aproximovat. Pro potřeby této práce lze uvažovat, že pravděpodobnost jejího vzniku je po dobu běhu aplikace konstantní a že se robot pohybuje po hladké neměnné podložce. Chyba bude převedena na informaci o ploše v okolí robotu, ve které se robot pravděpodobně nachází. Tato plocha bude vyjádřena poloměrem r_e , který údavá kruh v okolí předpokládaných souřadnic robotu.

Pro zjištění změny Δr_e v závislosti na ujeté vzdálenosti lze použít rovnici (5). Pro její správnou aplikaci je důležité, aby byl výpočet polohy zatížen systematickými chybami co nejméně. Je tedy vhodné nejprve provést detekci a eliminaci systematických chyb metodou popsanou v podkapitole 8.3.1.

K výpočtu nesystematické chyby lze použít metodu popsanou v [32]. V tomto článku se píše: „vliv nesystematické chyby robotu lze nejlépe vyjádřit jako průměrnou chybu natočení robotu“. To je proto, že u diferenciálního podvozku chyba natočení ku celkovému natočení odpovídá chybě vzdálenosti výchozí a koncové polohy ku uražené vzdálenosti, ale snáze se zjišťuje. Platí vztah

$$\epsilon_e \simeq \frac{\epsilon_\sigma}{\sigma} \simeq \sqrt{\left(\frac{\epsilon_x}{d}\right)^2 + \left(\frac{\epsilon_y}{d}\right)^2} \tag{4}$$

kde ϵ_e ... *nesystematická chyba pohybu,*
ϵ_σ ... *chyba natočení diferenciálního podvozku,*
σ ... *celkový úhel otočení podvozku,*
ϵ_x ... *chyba polohy v ose x kartézské souřadné soustavy,*
ϵ_y ... *chyba polohy v ose y,*
d ... *celková uražená vzdálenost.*

Použitá metoda funguje na principu měření průměrné chyby natočení robotu $\epsilon_{\sigma avrg}$ po projetí čtverce o délce hrany 4 m. Robot se pohybuje 4 m rovně a následně zatočí v úhlu 90°, to se opakuje čtyřikrát. Jeho očekávané výsledné natočení je shodné s natočením výchozím. Hodnota $\epsilon_{\sigma avrg}$ je vypočtena jako průměr rozdílu jeho očekávaného a skutečného natočení na konci trasy (při opakovaném měření). V jednom cyklu měření polohy robot urazí 16 m a otočí se o 360°. Chybu změny r_e v závislosti na ujeté vzdálenosti a úhlu natočení od posledního měření lze vypočítat jako

$$\Delta r_e = \epsilon_e \Delta d + \epsilon_e \Delta \sigma \tag{5}$$

kde Δr_e *... změna poloměru chyby polohy podvozku v metrech,*
Δd *... vzdálenost ujetá od posledního měření v metrech,*
$\Delta \sigma$ *... změna úhlu natočení od posledního měření ve stupních.*

Hodnota Δr_e je tedy hodnotou změny poloměru kruhu (se středem v odhadované poloze robotu), ve kterém se robot nachází. Pro přidání dodatečné jistoty je k tomuto výsledku vhodné přičíst 10% z jeho hodnoty. Je nutné poukázat na to, že pravděpodobnost rozložení polohy robotu není kruhová, ale eliptická [42]. Nicméně v aplikaci rovnice (5) jde o zjištění maximální deviace robotu od očekávané polohy, k čemuž aproximace kružnicí postačuje.

6.2 Triangulace

Pomocí jednoduchého použití Pythagorovy věty je možné ze znalosti vzdálenosti a směru alespoň dvou známých pozic v mapě určit polohu robotu. K tomu musí být inteligentní dům vybaven emitory jistého druhu a robot jejich přijímačem s možností určit jejich směr a vzdálenost jak bylo popsáno v podkapitole 5.3.2. Za předpokladu známé vzdálenosti nebo směru tří těchto bodů je možno provést triangulaci bez znalosti druhého parametru.

6.3 Rozpoznání obrazu

Pomocí metod rozpoznání obrazu je možné získat pozici robotu. Možné přístupy k řešení tohoto problému jsou následující.

Rozpoznání prostředí

Rozpoznání prostředí je výpočetně nejnáročnější způsob lokalizace robotu v inteligentním domě. Jeho implementace je ztížena především velkou dynamikou domu, ve kterém žijí lidé. Použití metody rozpoznání prostředí je nad rámec této práce.

Rozpoznání robotu

Pokud je inteligentní dům vybaven rozsáhlým kamerovým systémem, mohou být kamery využity k nalezení robotu v domě. K tomu je třeba, aby byl robot zřetelně označen barevným vzorem, nebo při nutnosti lokalizace intenzivně svítil například v infračerveném spektru. Tento druh lokalizace je ale vysoce náročný na kalibraci domácích kamer a jejich instalace v domě může narušit pocit soukromí jeho obyvatel.

Rozpoznání kódu

Robot vybavený kamerou může za pomocí technik rozpoznání obrazu detekovat značky umístěné v inteligentním domě a z nich vyvodit svoji pozici. Tento způsob je funkčně obdobný předchozí metodě, ale řeší problém pocitu soukromí obyvatel, protože se v domě musí nacházet pouze jedna kamera, která je pevně spojena s robotem. Tento přístup je úzce svázán s metodou lokalizace popsanou v minulé podkapitole a možný způsob jeho implementace je popsán v následujících sekcích této kapitoly.

6.3.1 Vhodný vzor

Pro absolutní lokalizaci robotu v trojrozměrném prostředí lze použít detekci různých kontrastních vzorů (v různých barevných spektrech) za pomoci počítačového vidění. Tyto vzory mohou být buď planární (obecně tvar polygonu nebo kruhové výseče na rovné ploše) nebo plastické libovolného tvaru. Vzhledem k nízké kvalitě kamery, použité při řešení této práce, je vzor vhodné volit co nejjednodušší. Zároveň by však mělo existovat dostatečné množství variací pro pokrytí prostoru inteligentního domu.

Poloha vzoru

V optimálním případě by v inteligentním domě byla vzorem opatřena každá plocha. Prakticky ale stačí, v závislosti na nábytku v domě, jeden až dva umístěné na stropě. Výhoda vzorů na stropě je, že je sníženo riziko jejich nechtěného zakrytí. Nevýhodou je, že za běžného provozu robotu musí jeho kamera strop zabírat. Z tohoto důvodu jsou v této práci použity ploché značky, primárně navržené pro horizontální připevnění.

V závislosti, na mechanismu získání polohy při detekci vzoru v místnosti, musí být pro úspěšnou lokalizaci vždy viditelný jeden, dva nebo tři z nich. Vzory je tedy vhodné umístit naproti frekventovaným trajektoriím robotu (dveře), s co největší vzdáleností mezi nimi, a v zorném poli robotu za běžného provozu.

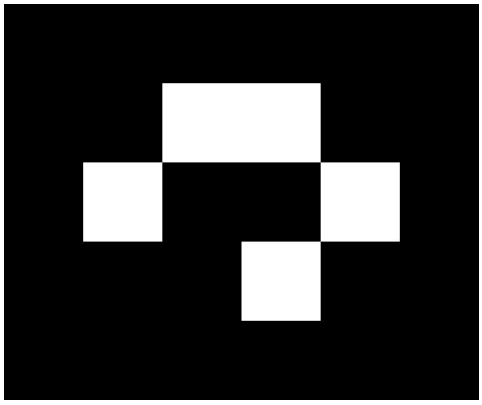
Vzhled vzoru

Pro lokalizaci robotu je v této práci použita metoda, která vyžaduje záběr alespoň jedné značky (kapitola 6.3.3). Z tohoto důvodu je nutné, aby byl vzor a jeho jednotlivé detekované součásti dostatečně velký, vzhledem ke kvalitě použitého hardware.

Pro spolehlivější použití technik počítačového vidění je vhodné volit dvoubarevný, kontrastní vzor. Z tohoto důvodu jsou použity černobílé značky.

Pro snížení negativního vlivu „zprůměrování“ hodnot sousedících pixelů, a protože většina digitálních kamer snímá obraz ve čtvercovém rámci, je vhodné volit hlavní kontrastní linie vzoru svisle a vodorovně.

Všem kladeným požadavkům vyhovuje návrh vzoru, použitý v [48]. Jedná se o obdélníkovou matici černých a bílých čtverečků. Krajní čtverce této matice jsou všechny vyplněné černou barvou a je kladen požadavek na dostatečně široké pásmo bílé barvy v okolí vzoru. Různou konstelací černých a bílých výplní uvnitř vzoru je dána jeho „hodnota“, kterou může uživatelský program použít k identifikaci. Příklad možného vzoru se nachází na Obr 6.1.



Obr 6.1: Možná varianta vzoru

Hodnota vzoru

Požadavky na „hodnotu“ značky (sekvenci barev vnitřních čtverců vzoru) jsou:

1. Kód nesmí obsahovat shluky stejných barev v řádku nebo sloupci. Za shluk jsou pro potřebu této práce považovány tři čtverečky stejné barvy.
2. „Hodnota“ značky by měla být horizontálně (a případně i vertikálně a rotačně) variantní. To zamezí falešné detekci například v zrcadle.
3. Jednotlivé, současně použité, kódy by se měly lišit vždy alespoň o jeden bit.
4. Vzor může obsahovat bity vyhrazené pro *checksum* nebo *paritu* značky, podobně jako je tomu u čárového kódu.
   Pozn.: Checksum ani parita nebyly při řešení implementovány.

Vzory, použité v této práci, mají rozměr 5x6 čtverečků. To odpovídá datové oblasti vzoru 4x3 čtverečky. Maximální počet vzorů, vyhovujících 3. požadavku, je 2^12. Pokud jsou na tuto sadu aplikovány požadavky 1 a 2, bude obsahovat 186 prvků. Pokud tuto množinu navíc omezíme pouze na vzory, které se liší alespoň o 2 bity, bude výsledek obsahovat 13 řešení. To může být pro potřebu inteligentního domu dostatečné, a zároveň se tím sníží riziko falešné detekce.

GlyphMaker

Volba různých vhodných vzorů, především v případě jejich velkého množství, může být složitý úkol. Z tohoto důvodu byl pro potřebu této práce vytvořen program **GlyphMaker**, který automaticky generuje vzory v obrázkovém formátu *png*. Navíc také vytvoří textové šablony pro použití k inicializaci programu počítačového vidění.

Program implementuje metody 1-3 popsané v předchozí podkapitole. Uživatel je při jeho spuštění vyzván k zadání jednotlivých parametrů výstupních vzorů.

Program **GlyphMaker** je psán v jazyce C# a jeho zdrojové kódy jsou v elektronické příloze této práce. Vnitřní interpretace „hodnoty“ vzoru je provedena pomocí typu *ulong*. To omezuje maximální velikost vnitřní matice vzoru na rozměr 8x8, 4x16, atd. V současné podobě provádí program **GlyphMaker** permutaci všech možných řešení a vyřazuje ty, které nevyhovují zadaným pravidlům. To může být zdlouhavé již při velikosti kódu 5x5. Možným způsobem pokračování v této práci by mohla být optimalizace tohoto programu.

6.3.2 Detekce vzoru

Pozn.: Implementace některých metod a nastavení hodnot konstant, zmiňovaných v této kapitole, jsou podrobněji vysvětleny v kapitole 7.3.5 a ve zmiňované referenci a ve zdrojových kódech v elektronické příloze této práce.

K rozpoznání lokalizační značky, popsané v předchozí kapitole, od barevných shluků vyskytujících se v běžném domě, je použit postup popsaný v [48]. Vychází z předpokladu, že je vzor čtyřúhelník s černým okrajem, obklopený bílou barvou. A že jej kamera zabírá celý.

Nalezení vzoru v obrazu

Pro detekci vzoru je nejprve nutné nalézt ve vstupním obrazu místa, na kterých se může značka nacházet s vysokou pravděpodobností. Krok popsaný v této podkapitole odpovídá „vybrání rysů“ (*feature extraction*) v modelu počítačového vidění. Postup detekce vzoru v obrazu je následující:

1. Převedení obrazu na „černobílý“.
2. Aplikace diferenčního hranového detektoru pro získání prudkých přechodů kontrastu v obrazu.
3. Prahování obrazu konstantním koeficientem z důvodu eliminace nevýrazých pechodů kontrastu (hran).

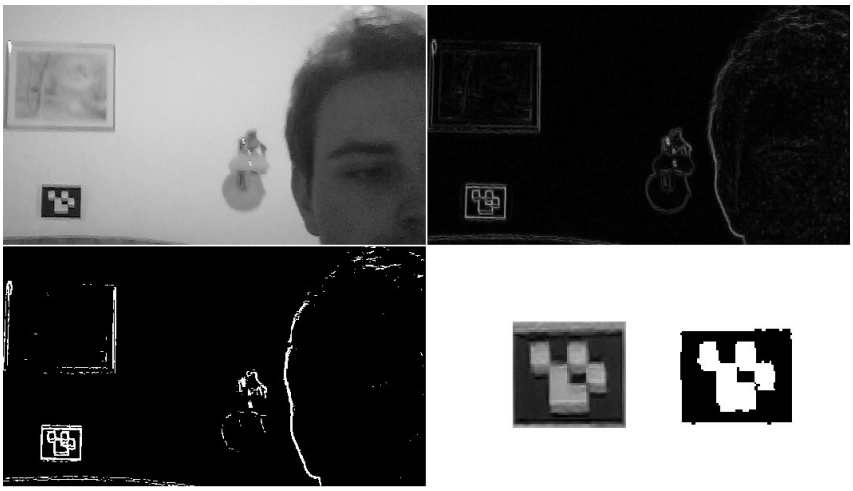
Tím vznikne obraz shodných rozměrů s původním snímkem, který obsahuje souvislé bílé křivky v místech, kde je přechod z tmavé barvy na světlou dostatečně prudký. Následně jsou v těchto křivkách rozpoznány oblasti, které mohou v původním snímku obsahovat hledaný vzor:

4. Detekce uzavřených křivek v obrazu.
5. Pro každou křivku se ověří, zda tvoří čtyřúhelník.

Všechny uzavřené křivky, vyhovující bodu 5, mohou být hledaný vzor. Pro každou z křivek je v její blízkosti vypočtena průměrná barva pixelů uvnitř a vně (v originálním obrazu). Pokud je poměr vnitřní a vnější průměrné barvy větší než nastavená hodnota, nachází se na daném místě v původním obrázku tmavý čtyřúhelník se světlým okrajem.

6. Každý nalezený čtyřúhelník je transformován do nového obrazu, který rozměrem odpovídá násobku rozměru (počet čtverečků) hledaného vzoru.
7. Transformovaný čtyřúhelník je prahován, čímž vyniknou bílé a černé plochy odpovídající jednotlivým čtvercům vzoru.

Pro představu postupu vysvětleného v této podkapitole, se na Obr 6.2 nachází přehled výstupu některých kroků. Při snímání bylo kamerou záměrně třeseno.



Obr 6.2: Proces rozpoznání vzoru. Zleva-doprava, zvrchu-dolů: krok 1, krok 2, krok 3, krok 6, krok 7

Detekce hodnoty nalezeného vzoru

V obrázku, vytvořeném v kroku 6 předchozí podkapitoly, se nachází transformovaný binární obraz. Teto snímek je potenciálně hledaný vzor.

8. V transformovaném prahovaném obrázku jsou detekovány okraje o šířce jednoho čtverce vzoru. Ty musí být u platné značky vždy černé.
9. Postupně jsou zjišťovány hodnoty barev vnitřních čtverců obrazu z předchozího bodu. Na základě hodnoty průměrné barvy uvnitř čtverce lze usoudit, že se jedná o černý, bílý nebo neurčitý čtverec.

Pokud při detekci v bodu 9 není žádná barva neurčitá, lze hodnoty získaného vzoru porovnat s databází vzorů. V případě identity lze usoudit, že se jedná o správný vzor.

6.3.3 Výpočet polohy robotu

Ze znalosti pozice několika vzorů (jejich referenčních bodů) lze určit jednoduchou triangulací pozici robotu. V této práci je ale použita metoda estimace polohy robotu ze znalosti jediné lokalizační značky. Pokud je v jednom snímaném obrázku nalezeno více vzorů, lze pozici určit jejich zprůměrováním.

Vzájemné posunutí a rotaci kamery vůči rozpoznanému rovinnému vzoru lze určit za pomoci metody *Coplanar POSIT* [49]. Prvními vstupními daty této metody jsou vzájemné polohy referenčních bodů na vzoru. Tyto body musí ležet v jedné rovině a více než dva z nich nesmí ležet na jedné přímce. Druhou vstupní informací jsou souřadnice těchto bodů tak, jak jsou zobrazeny na snímku. Souřadnice bodů jsou relativní vůči optické ose kamery (středu snímku). Poslední hodnota, důležitá pro funkci této metody, je hodnota ohniskové vzdálenosti kamery.

Ze znalosti referenčních soustav dům-vzor, vzor-kamera a kamera-robot je za pomoci homogenních transformací možno určit polohu robotu. Platí potom vztah:

$$P_r = P_v * P_{vk} * P_{kr} \tag{6}$$

kde P_r *... je matice pozice a rotace robotu,*
P_v *... je matice pozice a rotace detekovaného vzoru,*
P_{vk} *... je matice referenční soustavy vzor-kamera,*
P_{kr} *... je matice referenční soustavy kamera-robot.*

Všechny uvažované vztažné soustavy musí mít shodné pořadí os. Vzhledem k usnadnění při počítání s obrázky jsou v této práci použity levotočivé soustavy prostorových kartézských souřadnic. Ukázka referenčních systémů souřadnic robotu, kamery a vzoru je na Obr 8.5 na straně 91.

6.4 Mapování

K vytváření mapy prostředí využívají roboty firmy iRobot implicitně metodu Monte Carlo. O této metodě je možno zjistit více v publikaci [43]. Výhodou aplikace autonomního robotu v inteligentním domě je, že robot má apriorní znalost o prostoru, ve kterém se pohybuje. Dále je relativně snadné provést lokalizaci robotu jak je popsáno v předchozích podkapitolách. V případě naprosté ztráty robotu v domě může robot začít náhodně prozkoumávat terén a získané znalosti korelovat s modelem domu. Nevýhodou tohoto přístupu je ale opět vysoká dynamika překážek v domácnosti, ve které žijí lidé.

6.5 Návrh trasy

Pro optimální návrh trasy je třeba rychle a efektivně vypočítat trasu spojující dva body nebo místnosti. K tomu je využito trasování na dvou úrovních. Na globální úrovni se provede obecný návrh trasy tj. zjistí se, kterými místnostmi musí robot z lokální pozice projet, aby dorazil do místnosti cílové. Na druhé, lokální úrovni, robot vyhodnocuje způsob, jak danou místností efektivně projet tak, aby se vyhnul všem překážkám.

Vzhledem k faktu, že se robot pohybuje v rovině, je možné představit si mapu jako dvourozměrnou plochu. Polohu všech objektů v této mapě je pak možné specifikovat pomocí souřadnic x a y. Informace o prostoru domu jsou na základě jeho znalosti uloženy ve třech různých datových blocích. Jsou to:

1. **Mapa:** Jména, rozměry a polohy jednotlivých místností domu
2. **Přechody:** Informace o vzájemném propojení jednotlivých místností, tento datový blok je dynamicky vypočítán na základě mapy.
3. **Překážky:** Informace o známých statických překážkách v domě (stěny, sloupy, schodiště, dveře, atd.)

Algoritmus navádění, popsaný v následujících podkapitolách, je na robotu použit a jeho implementace je podrobněji vysvětlena v kapitolách 7.3.2 a 7.3.3.

6.5.1 Globální návrh trasy

Mapa se sestává z dvourozměrných polygonů, představujících jednotlivé místnosti domu. Tyto polygony je možno kvůli budoucím odkazům pojmenovat. Z důvodu eliminace minim při lokálním vyhodnocování (kapitola 6.5.3) je vhodné polygony volit tak, aby nebyly konkávní. Jedna místnost může sestávat z několika stejně pojmenovaných polygonů. V místě, kde se tyto polygony překrývají, se nachází přechody mezi jednotlivými místnostmi. Přechody je možné detekovat například pomocí některého nástroje pro polygon clipping. Z jednotlivých přechodů mezi místnostmi se utvoří struktura **přechodů**, pomocí které se bude vyhledávat možná cesta mezi jednotlivými body v mapě.

Ve struktuře **přechodů** jsou ke každé místnosti uloženi její sousedi a vzdálenost k nim. Dále je v této struktuře uložena informace o tom, na kterých souřadnicích se přechod nachází. V okamžiku, kdy robot vyhodnotí potřebu dorazit do místnosti identifikované jejím jménem, provede se v této struktuře **přechodů** vyhledání nejkratší cesty do cílové místnosti pomocí rekurzivního A^* trasovacího algoritmu. Více o A^* algoritmu je k nalezení v [33]. Po nalezení nejkratší sekvence místností, vedoucí k cílovému bodu, se tato sekvence předá lokálnímu plánování trasy, které po ní vyrazí. Výpočet možných spojení ale neustává, protože robot může dorazit do situace, kdy je jeden z přechodů nemožný.

6.5.2 Lokální návrh trasy

Na základě příkazu globálního plánovače trasy robot vyrazí k prvnímu spojujícímu bodu na cestě mezi současnou a cílovou pozicí. K výpočtu trasy na této úrovni je použito plánování trasy pomocí vektorových polí, popsané v podkapitole 6.5.3. Informace o poloze a vlivu překážek jsou uloženy v datovém bloku **překážky**. Při detekci uvíznutí, neboli neschopnosti se na dané místo dostat, je provedena korekce lokálního minima vložením nové „virtuální“ překážky před robot. Pokud opakovaná korekce nevede k nalezení trasy mezi robotem a cílovým bodem, je na základě místa uvíznutí (což je přechod mezi místnostmi, který se lokálnímu plánovači nepovedl) proveden backtracking k následující možné trase, která tento přechod nevyužívá.

Pokud lokální plánovač narazí na dříve neznámou překážku, uloží si informace o její poloze. Takto nalezené překážky jsou uloženy do paměti jako dočasné. Když při opakovaném pokusu (v horizontu několika týdnů) robot na tuto překážku znovu narazí, je hodnota perzistence překážky inkrementována. V jednodenních intervalech je hodnota periodicky snižována a překážka je úplně odstraněna ze seznamu v případě, kdy je její existence vyvrácena (robot je nucen daným úsekem projet a na překážku nenarazí).

6.5.3 Vektorová pole

Pozn.: Tato podkapitola je částečně převzata z mé bakalářské práce [34].
Prostředí pro pohyb robotu si lze představit jako soustavu elektrických nábojů ve dvou či vícerozměrném prostředí. Z fyziky víme, že shodné náboje se odpuzují a opačné přitahují, a elektrické pole je orientováno směrem od kladných nábojů k záporným. Tohoto faktu využívá metoda vektorových polí. Dle zvolené konvence budou překážky kladně nabité a cíl, ke kterému budu směřovat, bude mít náboj záporný. V prostředí, kde překážky a cíl mají takto zvolenou polaritu nábojů, stačí z jakéhokoliv místa sledovat siločáry, které vedou do požadovaného cíle. Tyto náboje se liší svojí intenzitou a tudíž i intenzitou pole, které kolem sebe vytváří. Ve skutečném světě tato intenzita klesá se čtvercem vzdálenosti od pozice náboje (je předpokládán bodový náboj).

Vycházíme z elementárních vztahů elektrostatiky. Na základě Coulombova zákona:

$$\vec{F} = \frac{1}{4\pi\epsilon_0}\frac{q_1 q_2}{r^3}\vec{r} \tag{7}$$

a vztahu mezi intenzitou elektrického pole a elektrickou silou

$$\vec{E} = \frac{\vec{F}}{q} \tag{8}$$

kde $\vec{E}$ *… je intenzita elektrického pole částice,*
q *… je náboj částice,*
$\vec{F}$ *... je elektrická síla působící na částici.*

lze definovat vztah pro elektrické pole bodového náboje

$$\vec{E} = \frac{1}{4\pi\epsilon_0}\frac{q}{r^3}\vec{r} \tag{9}$$

kde ϵ_0 *… je elektrická konstanta,*
r *… je vzdálenost od náboje.*

Tato rovnice definuje elektrické pole kolem osamoceného náboje. Pokud v jeho okolí existuje jeden nebo více dalších bodových nábojů, využíváme principu superpozice pro stanovení intenzity elektrického pole pro určité místo. Výsledná intenzita elektrického pole je potom logicky jen vektorový součet intenzit jednotlivých nábojů.

$$\vec{E} = \sum \vec{E}_i \tag{10}$$

$$\nabla \vec{E} = \frac{\rho}{\epsilon_0} \tag{11}$$

kde ρ *... je objemová hustota náboje*

Coulombův zákon je pouze speciální případ obecnějšího Gaussova zákona. Pomocí Gaussova zákona je možno elektrické pole vypočítat jako souvislé rozložení náboje v prostoru.

Gaussův zákon pro elektrické pole je jednou ze čtyř Maxwellových rovnic, které popisují makroskopickou teorii elektromagnetického pole.

Pro zjednodušení výpočtu je možno upravit rovnici (9) tak, že položíme konstantní část rovnice rovnu 1.

$$4\pi\epsilon_0 = 1 \tag{12}$$

Vztah pro elektrickou intenzitu se zjednoduší nasledujícím způsobem:

$$\vec{E} = \frac{q}{r^3}\vec{r} \tag{13}$$

Nyní je možné vypočítat pole elektrického náboje, jehož intenzita se zmenšuje se čtvercem vzdálenosti a začíná velikostí náboje q . Kdyby tedy bylo třeba vypočítat, jakou silou na daný bod působí určitý náboj, stačí za r dosadit vzdálenost bodu od náboje a za q hodnotu náboje.

V souladu s předestřenou konvencí je tedy cíl označen záporným nábojem a překážky kladným nábojem. Ukazuje se však, že se stoupající vzdáleností od cíle se zvyšuje i negativní dopad překážek na požadovanou trajektorii. Tuto nevýhodu je nutné obejít použitím homogenního pole.

Homogenní elektrické pole je pole, které má ve všech svých bodech stejný směr a stejnou velikost intenzity. Typickým příkladem homogenního pole je elektrické pole mezi dvěma nekonečně velkými nabitými deskami. V případě použití homogenního pole pro robotické aplikace se musí jít ještě o něco dále. Nestačí totiž uvažovat homogenní pole mezi dvěma plochami, ale pole musí mít rovnoměrnou intenzitu paprskovitého tvaru, což je fyzikálně nemožné. Nyní nezbývá, než se oprostit od fyziky a modifikovat některé klíčové vztahy. Pro přiblížení se k tomuto požadavku ve skutečném světě je možné uvažovat dostatečně dlouhé nabité vlákno s rovnoměrně rozloženým nábojem, kdy je dán předpoklad, že intenzita elektrického pole neklesá se vzdáleností od osy vlákna. Dostáváme paprskovité elektrické pole, které má ve všech bodech stejnou velikost elektrické intenzity

$$E = \frac{q}{\epsilon_0} \approx q \,. \tag{14}$$

Ze vztahu (14) vyplývá, že intenzita elektrického pole je v tomto uvažovaném případě konstantní. Potom pro elektrický potenciál takového elektrického pole platí vztah:

$$\phi = \int E \, dr = q\, r \tag{15}$$

Dopad cílového náboje bude pak neměnný v poměru k ostatním nábojům nezávisle na vzdálenosti od něj. Takže vliv působení náboje, indikujícího pozici cíle, bude konstantní bez ohledu na vzdálenost r . V některých aplikacích je dokonce možné naopak vliv cílového náboje s rostoucí vzdáleností zvyšovat, čímž lze dosáhnout daleko „agresivnějšího“ postupu, pokud se robot nachází dále od požadovaného bodu. Pokud je tedy úkolem navést robot na cíl, stačí vhodně umístit záporný pól (cíl) a kladné póly (překážky). Robot se bude pohybovat směrem k nižšímu potenciálu tak, že se překážkám vyhne a dorazí do cíle.

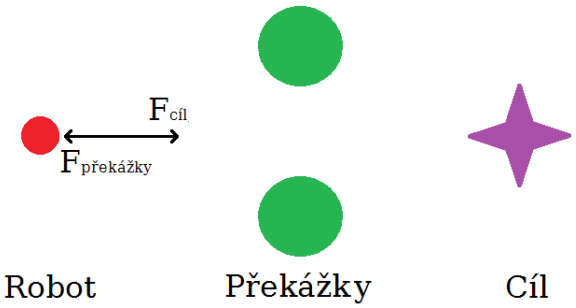
Problematika lokálních minim

Jednou z nejdůležitějších nevýhod vektorových polí je to, že tato metoda nemůže být bez ztráty elegance a rychlosti použita pro plánování trasy v globálním měřítku. Blízce se tohoto postihu týká takzvaná problematika lokálních minim. Lokální minima jsou místa, kde vlivem překážek může robot uvíznout, a ze kterých již nevede žádná cesta směrem k nižší hodnotě potenciálu.

Nejjednodušším druhem lokálního minima je případ, kdy se překážka nachází přímo mezi robotem a cílem. Lokální minimum tedy vznikne, pokud jsou tyto tři prvky na jedné ose a neexistuje žádná jiná překážka v dosahu, která by narušila tento rovnovážný stav. Odpudivá síla překážky se v tomto případě odečte s přitažlivou silou cíle a výsledná síla je nulová, což vede k zastavení robotu. Tento druh lokálního minima je patrný z Obr 6.3. Jedná se ale o druh lokálního minima s velice malou pravděpodobností výskytu ve vysoce dynaimckém prostředí pohybujícího se robotu. Tento druh lokálního minima se tedy nebude nadále uvažovat.

Obr 6.3: Lokální minimum s jednou překážkou

Na Obr 6.4 je častěji se vyskytující druh lokálního minima. Totiž případ, kdy se odpudivé síly více překážek sčítají a vyváží tak přitažlivou sílu cíle. Robot se tedy zastaví v sedle, tvořeném kladným elektrickým polem dvou překážek. Z tohoto důvodu je třeba vhodně volit intenzitu překážek v poměru s intenzitou cíle tak, aby pro vznik tohoto druhu lokálního minima musela být mezera mezi dvěma překážkami menší než šířka robotu. Tím se efektivně zmenší šířka a „prudkost“ sedla, a tudíž je menší pravděpodobnost, že v něm robot uvízne.

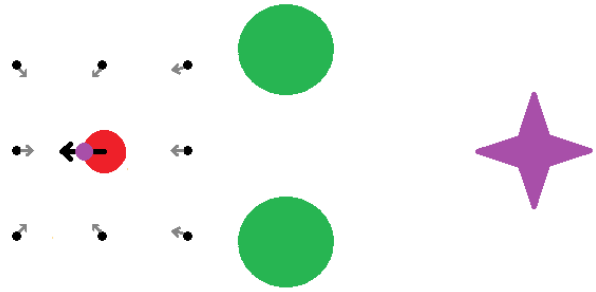


Obr 6.4: Lokální minimum s více překážkami

Výpočet vektoru pohybu

Vektor pohybu se vypočítá jako hodnota požadovaného směru pohybu ve dvou osách souřadné soustavy. Při započetí výpočtu se nejprve inicializuje vektor pohybu na hodnotu působení samotného cíle pomocí rovnice (14). Následně se provede výpočet vektoru působení každé překážky na robot. Pokud je překážka v dosahu (dosah je parametr překážky stejně jako její poloha nebo intenzita), vypočte se vliv intenzity překážky na vektor pohybu pomocí rovnice (13). Zvláštní případ je překážka typu stěna. U stěny se nejprve určí její nejbližší bod k robotu (tj. kraj stěny nebo kolmice na stěnu) a následně se s tímto bodem počítá stejně jako s překážkou typu bod. V rohu místnosti se takto budou sčítat intenzity dvou stěn a robot je vyhodnocuje jako dvě samostatné překážky. To vede k obezřetnému vyhýbání se rohům.

I za předpokladu, že je mapa domu navržena správně, se může v okamžiku nalezení dočasné překážky stát, že robot uvízne v lokálním minimu. Proto při každé iteraci výpočtu požadovaného směru pohybu probíhá také detekce lokálního minima. Tato detekce je částečně založena na detekci použité v [38]. Vzhledem k dynamice pohybujícího se robotu lze očekávat, že se robot nikdy nezastaví přesně v bodu lokálního minima, ale že jím nejprve projede. Kvůli snížení výpočetních nároků neprobíhá detekce lokálního minima až do okamžiku, kdy vektor požadovaného pohybu robotu směřuje od cíle. Následně proběhne detekce lokálního minima, a to tak, že robot vypočítá směry požadovaného pohybu v osmi bodech ve své bezprostřední blízkosti. Pokud všechny tyto body směřují „dovnitř“ k robotu, je možné usoudit, že se nachází v lokálním minimu. Ukázka takovéto detekce je znázorněna na Obr 6.5.



Obr 6.5: Detekce lokálního minima

Na předešlé ilustraci je robot znázorněn červeně. Přesná poloha lokálního minima je vyznačená malým fialovým kruhem v blízkosti robotu. Překážky a cíl jsou znázorněny obdobně jako na předchozích obrázcích. V tomto případě robot fakticky z důvodu dynamiky svého pohybu již překročil hranici lokálního minima. Jeho nově navržená trasa směřuje tedy od cíle. V tomto okamžiku proběhne detekce minima v osmi bodech sousedících s robotem. Tyto kontrolní body jsou vyznačeny černě. Vypočítaná požadovaná trajektorie z těchto bodů je znázorněna šedými šipkami. Vzhledem k faktu,

že všechny tyto šipky směřují k lokálnímu minimu (tj. směrem dovnitř vyhraničeného čtverce), robot vyhodnotí, že se nachází v lokálním minimu.

V okamžiku, kdy robot vyhodnotí, že se ocitl v lokálním minimu, provede pokus o nápravu. Nejprve zvýší intenzitu cíle o 20%. Vlivem této úpravy se pokusí mezi překážkami projet, protože ve skutečnosti nemusí lokální minimum vznikat překážkami umístěnými bezprostředně před robotem. Tato metoda ale s vysokou pravděpodobností selže, čímž je aktualizována poloha dané překážky a tím i poloha robotu. Intenzita překážky je navrácena na původní hodnotu. Následně je upravena trajektorie pohybu tak, aby směřovala na stranu nejmenší výchylky kontrolního bodu. Pokud tento pokus o nápravu opakovaně nepomůže robotu uniknout z lokálního minima, umístí na svoji polohu dočasnou překážku, která jej z lokálního minima bezpečně „odmrští“. Takto vložená dočasná překážka navíc po dobu své existence brání robotu do lokálního minima opětovně zajet.

6.5.4 Příklad návrhu trasy

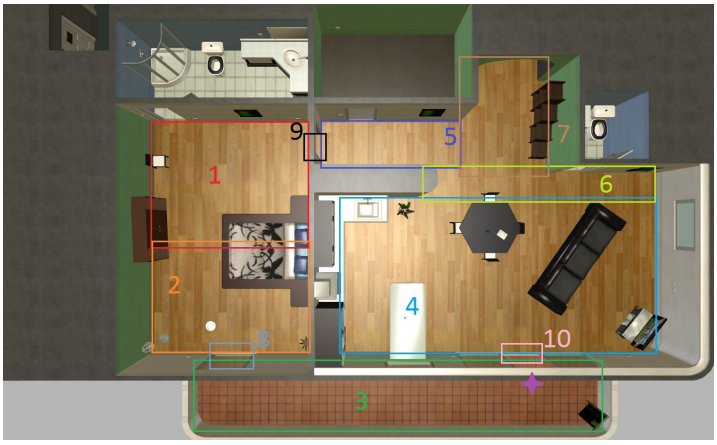
Mějme inteligentní byt, který může vypadat jako apartmán na Obr 6.6. V levém dolním rohu je vidět čekající robot. Tento byt může být v **mapovém** bloku znázorněn pomocí překrývajících se polygonů obdobně jako je tomu na Obr 6.7. Na obrázku zbývají některé nezmapované pokoje, ty jsou ale v tomto příkladu redundantní a není třeba s nimi počítat.

Obr 6.6: Inteligentní byt s robotem v levém dolním pokoji

Na Obr 6.7 jsou jednotlivé polygony očíslovány. Těmto číslům jsou přiřazeny názvy místností. Více polygonů (místností) může mít stejný název, a všechny místnosti nemusí být pojmenovány. Kromě názvů jednotlivých místností je také možné vytvořit

pojmenované body, identifikované přesnými souřadnicemi. Názvy těchto pojmenovaných bodů ale musí být unikátní.

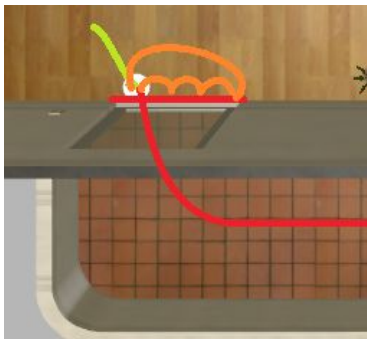
1. Ložnice
2. Ložnice
3. Balkon
4. Kuchyně
5. Chodba
6. Kuchyně
7. Chodba
8. Balkon - ložnice
9. Chodba - ložnice
10. Kuchyně – balkon



Obr 6.7: Rozložení místností inteligentního apartmánu

V našem příkladu inteligentní dům vyhodnotil, že na balkonu došlo k požáru, otevřenými dveřmi se dovnitř dostal kouř. Robot dostal za úkol prozkoumat místo pravděpodobného ohně. Toto místo je znázorněno fialovou hvězdicí u otevřených balkonových dveří. Dům si je vědom toho, že se robot nachází v polygonu 2 a že požár je v polygonu 3. Nahlédnutím do databáze **přechodů** vyhodnotí ovládací algoritmus robotu, že nejkratší cesta do polygonu 3 je přes polygon 8 (viz Obr 6.8).

Obr 6.8: Vhodná trasa k místu určení



Obr 6.9: Snaha robotu projít zavřenými dveřmi

Bohužel jsou automatické balkonové dveře v ložnici právě porouchané, což inteligentní dům neví. Robot vyrazí na cestu do polygonu 8, ale naráží na neočekávanou překážku, jak je patrné z Obr 6.9. Po několika neuspěšných pokusech robot vzdává snahu projít sklem a informuje globální plánovač o neúspěchu. Globální plánovač již má nachystanou alternativní trasu, kterou robotu předá.

Nová trasa vede postupně přes polygony 2, 1, 9, 5, 6, 4 a 10. Tato trasa je zvýrazněna na Obr 6.10. Na obrázku jsou modře vyznačena potenciálová pole jednotlivých stěn a překážek, působících na robot. Červeně vyznačené hvězdy na prahu

balkonových dveří (polygon 8) jsou nově přidané překážky. Ty vznikly neúspěšnými pokusy projít na balkon.

Obr 6.10: Alternativní trasa robotu

Uprostřed cesty (přechod mezi polygony 6 a 4) objevil robot neznámou překážku. Nově nalezená překážka jsou židle a stůl uprostřed místnosti. Po jejím úspěšném překonání se robot opět vydal nejkratší cestou k polygonu 10. Jeho trasa je znázorněna na Obr 6.11. Pokud se robotu podaří na židli a stůl narazit několikrát za sebou aniž by jedenkrát chyběly, inkrementuje jejich validitu v databázi **dočasných překážek**.

Obrázky 6.6 až 6.11 jsou graficky upravené výstupy ze skutečné simulace robotu Create. Použitý simulátor je probrán v kapitole 7.3.7 - Simulace.

Obr 6.11: Deviace z optimální trajektorie vlivem překážek

7 IMPLEMENTACE

V této kapitole je vybrána vhodná vývojová platforma pro robot Create. Je zde vysvětleno použité schéma programu a jeho implementace. Při implementaci je kód tvořen pro daný robot, ale je kladen důraz na to, aby tuto aplikaci bylo možno použít pro jakýkoliv diferenciálně řízený robot. Největší důraz je kladen na robustnost, modularitu a bezpečnost kódu. Při správně konfiguraci a spuštění nesmí po dobu jeho běhu vzniknout žádný nebezpečný (neopravitelný) stav a přístup k robotu musí být limitován pouze pro oprávněné uživatele. Z toho důvodu jsou v inicializační fázi programu neurčité stavy považovány za špatné a je generována výjimka (program selže s popisem problému).

7.1 Vývojová platforma

Základním požadavkem na vybranou vývojovou platformu je snadná interakce s inteligentním domem. Vývojová platforma musí podporovat iRobot Create a širokou škálu dalších senzorů. Je vhodné, aby zvolené prostředí obsahovalo simulátor. Na základě těcto požadavků bylo vybráno robotické vývojové prostřed MRDS 4 (*Microsoft Robotics Developer Studio 4*).

7.2 Microsoft Robotics Developer Studio 4

Pozn.: Tato podkapitola je částečně převzata z mé bakalářské práce [34].
Microsoft Robotics Developer Studio je dnes objektivně nejlepší robotický simulátor k volnému stažení pro domácí a školní použití. Využívá *CCR* a *DSS* (*.NET CLR*), k běhu aplikací je použito řízení tokem dat.

CCR (Concurency and Coordination Runtime) umožňuje aplikacím provádět asynchronní operace, aniž by programátor musel psát složitá vlákna. RDS jej používá jako reakci na zvláštní potřeby robotických aplikací. Vzhledem k rozsáhlosti a složitosti problematiky *CCR* je v této práci vysvětlena pouze zevrubně. Pro jeho dobré porozumění je vhodné nastudovat kapitolu 2. v [40].

DSS (Decentralized Sotware Services) je nástroj, který vývojáři umožňuje sledovat stav jednotlivých vláken (služeb neboli servisů) za běhu programu. *DssNode* je aplikace nadřazená všem službám. Shrnuje o nich informace a je nezbytná k jejich inicializaci i ukončení. Je v ní spouštěn soubor manifest, který inicializuje celou aplikaci. Pro správné pochopení významu *DSS* a jeho jednotlivých funkcí v rámci celé robotické aplikace je doporučeno přečíst kapitolu 3 v [40].

Základním stavebním blokem v RDS je takzvaná služba (servis). Tu je možné považovat za vlákno programu, které obsahuje kód potřebný k provádění jedné nebo

více funkcí (každá služba zpravidla obsahuje několik vlastních vláken). Tím se rozumí například čtení dat z jednoho senzoru, odesílání dat na výstup nebo komunikace s jiným vláknem nebo vnějším softwarem. Robotické aplikace se zpravidla skládají z více služeb, které spolupracují při ovládání robotu. Více v kapitole 1. [40].

Každá služba deklaruje vlastní operace (porty), pomocí kterých jsou s ní jiné servisy schopny komunikovat. Komunikace mezi servisy probíhá zasíláním zpráv na jednotlivé porty. Tyto zprávy (podle druhu jejich implementace) mohou měnit nebo získávat stav přijímajícího servisu. Třetím druhem komunikace mezi službami je takzvaný *subscription*. Servis A může pomocí *subscription* zažádat servis B o pravidelné zasílání zpráv, které služba B generuje například při změně jejího vnitřního stavu.

V MRDS je možné deklarovat takzvané partnerské vztahy mezi servisy. Těmito vztahy se vyjadřuje závislost jedné služby na druhé. Služba zpravidla nemůže být inicializována, dokud nejsou spuštěni její partneři. Případné kruhové partnerské vztahy je DSS schopno řešit interně. Partnerství s jiným servisem dává najevo, že bude mezi oběma službami probíhat důležitý datový a kontrolní tok.

Základním blokem simulace je takzvaná entita. Entitou může být jakýkoliv prvek, obsažený v simulovaném prostředí, například obloha, terén nebo robot, ale třeba i kamera, která zprostředkovává grafický výstup simulace. Parametry entit je možné nastavovat (ovládat) pomocí manifestů. Každá entita je dceřinou třídou základní entity nebo entity z ní odvozené. Dědí tedy vždy vlastnosti všech nadřazených entit.

MRDS je omezeno na programovací jazyk C# a grafický programovací jazyk VPL (*Visual Programing Language*). Platforma, na které je možné MRDS (*DssHost*) provozovat, je pouze operační systém *Windows*.

Simulační engine využívá AGEIA PhysX pro simulaci reálných fyzikálních podmínek. Pomocí přesměrování manifestů a událostí z reálných vstupů a výstupů robotu tvoří obraz simulovaného prostředí. VSE (*Visual Simulation Environment*) je nadstavba nad simulačním prostředím a tvoří nástroj, sloužící ke grafickému zobrazení simulace.

Možnost přenosu aplikace ze simulace na skutečné zařízení je velice jednoduchá, zpravidla jde o změnu pár řádků kódu. Také přenos služby z jedné aplikace do druhé je díky systému řízení tokem dat velmi snadné. RDS navíc obsahuje knihovny pro obsluhu téměř všech druhů externích robotických a jiných zařízení, alespoň v generické podobě. Pro mnoho komerčních robotů navíc RDS obsahuje služby psané speciálně k jejich obsluze, což dává uživateli možnost využít všech jejich výhod oproti generickým zařízením.

Výhodou RDS je především jeho naprostá přizpůsobivost a obrovská, stále se rozšiřující, základna knihoven a ovladačů k novým robotickým zařízením. Řízení tokem dat a systém služeb dává uživateli možnost naprogramovat v zásadě jakoukoliv robotickou aplikaci. Například vytvoření aplikace s použitím joysticku je pro zkušeného

uživatele ve VPL otázkou několika minut. Navíc je možné vytvářet nové služby (servisy) v prostředí *Microsoft Visual Studio*. Je tedy například možné vytvořit službu pro obsluhu joysticku, která vyhovuje všem kladeným potřebám. Také pro implementaci algoritmu vektorových polí poskytuje RDS přivětivé mechanismy, které je možné využít. Samotná implementace je součástí této práce. Nemalou výhodou je i to, že *Microsoft* v roce 2007 uvolnil RDS k bezplatnému domácímu a školnímu použití, aniž by přestal s jeho dalším vývojem.

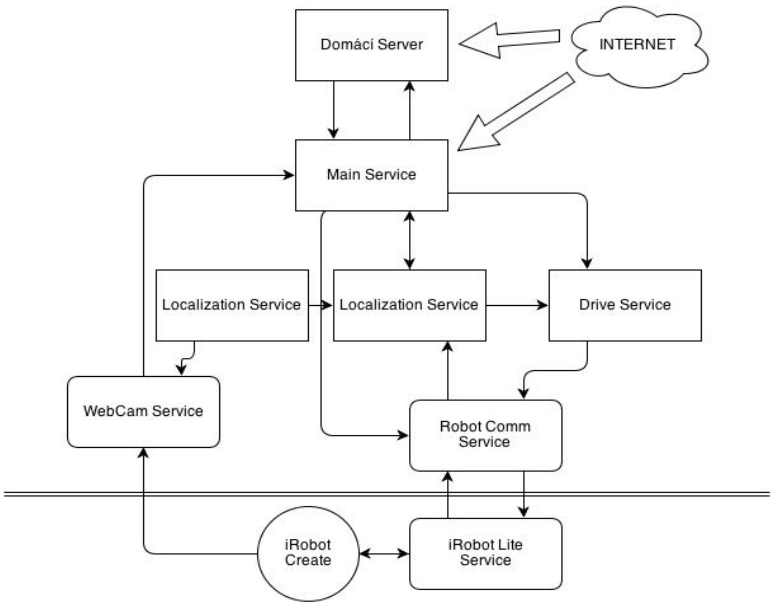
Ve zbytku této práce je při popisech různých funkcí, konstrukcí a operací používaných v MRDS užito slovních a významových zjednodušení, která usnadní pochopení i čtenářům, jež nejsou v MRDS a jeho principech zběhlí. Termíny používané v tomto prostředí, které nemají v českém jazyce nebo procedurálním programování ekvivalent, jsou vysvětleny, nebo je použit programátorský termín lépe popisující výsledek jejich funkce, přestože nepostihuje přesný postup, jakým k tomuto výsledku dochází. Text v této práci je tvořen tak, aby zkušený programátor v MRDS přesně věděl co je autorem zamýšleno, i když nebude použit přesný termín z prostředí MRDS. Autorem této práce jsou doporučeny reference [40] a [41] jako učební pomůcky pro začátečníky v MRDS.

7.3 Řídicí program

Aplikace vyvíjené v MRDS sestávají z více paralelních servisů. Před implementací robotické aplikace je třeba důkladně promyslet jejich schéma. Aplikaci je třeba navrhnout tak, aby přechod ze simulace na skutečný robot byla co nejsnadnější. V ideálním případě by přechod na skutečný robot sestával pouze ze změny konfiguračních souborů. Dále je třeba při návrhu myslet na to, že některé (nebo všechny) servisy mohou běžet přímo na hardwaru připojeném k robotu, a jiné na serveru inteligenního domu. Z tohoto důvodu je třeba navrhnout řídicí program tak, aby mohlo být dosaženo minimalizace datového přenosu mezi domem a robotem, a zároveň kladeny co nejmenší výpočetní nároky na robot samotný (očekává se, že bude k robotu připojeno zařízení s malou výpočetní kapacitou).

Na Obr 7.1 je znázorněn diagram implementovaného rozložení servisů a v několika následujících podkapitolách popsána jejich základní funkčnost. Tyto servisy jsou zobrazeny jako bloky v diagramu na Obr 7.1, přičemž šipky zde označují důležité toky informací. Při implementaci programu na reálný robot je, oproti implementaci do simulátoru, nutné vytvořit další servis, který tvoří rozhraní mezi senzory a akčními členy skutečného robotu a aplikací samotnou. Tento pomocný servis je znázorněn na Obr 7.1 pod dvojitou čarou. Za předpokladu, že jako kontrolní obvod skutečného robotu bude zvoleno zařízení podporující operační systém Windows, poběží některé servisy na samotném robotu, zatímco jiné na domácím počítači. DSS se poté postará o jejich

zabezpečenou komunikaci. Tímto řešením je možné dosáhnout snížení množství přenášených dat.



Obr 7.1: Schéma řídicího programu

Blok *„Domácí Server“* na Obr 7.1 ilustruje vzájemnou komunikaci domu a aplikace robotu. Ve skutečném provozu se může jednat o jakékoliv zařízení v domácnosti, s libovolným operačním systémem, které by komunikovalo se servisem **Main**. Blok *„INTERNET“* zobrazuje vstupní bod internetového připojení k robotické aplikaci. Připojení k servisu **Main** může být přímé nebo zprostředkované (routované) skrze domácí server. Blok *„iRobot Create“* představuje základní hardware, což je robot samotný a webová kamera připojená k jeho tělu. Na Obr 7.1 se nachází pod dvojitou čarou; to symbolizuje, že není využit v simulaci.

V dalších podkapitolách je popsána funkčnost jednotlivých servisů. Vzhledem k faktu, že servisy běží paralelně a mohou být spuštěny na jedné nebo více stanicích, nemusí uvedené pořadí odpovídat tomu, jak jsou ve skutečnosti spouštěny. Ke každému ze servisů je v podkapitolách vysvětlena jeho funkcionalita. Jsou zde popsány již implementované funkce a navrženy další, které je možné implementovat v navazující práci. Ke každému ze servisů jsou vyjmenováni jeho partneři. Je důležité si uvědomit, že v DSS servis A může být partnerem servisu B, ale nemusí tomu tak být obráceně. Více o partnerských vztazích služeb se nachází v podkapitole 7.2.

7.3.1 Servis Main

Centrem robotu je servis **Main**. Servis se chová jako prostředník mezi robotem a inteligentním domem. Navíc spouští webový server, pomocí kterého je možné robot

ovládat na dálku pomocí internetového prohlížeče. Na základě příkazů servis buď zprostředkovává domácímu serveru různé informace o robotu nebo řídí jiné servisy robotu.

Partnerské vztahy

Tato služba (servis) se po vytvoření připojí k následujícím „partnerům“:

1. **RobotComm** je prostředník mezi skutečným hardware robotu a touto aplikací. Servis **Main** zprostředkovává uživateli/serveru informace o stavu senzorů robotu pomocí této služby.
2. Servis **Localization** má za úkol udržovat informaci o poloze robotu v inteligentním domě a provádí navádění robotu na globální úrovni. Servis **Main** z něho získává informaci o aktuální poloze robotu. Pokud robot dostane za úkol dojet na určité místo, je tento požadavek zaslán právě službě **Localization**.
3. **Drive** servis je službou **Main** využíván pro přímý přístup k motorům robotu a tudíž k jeho přímému ovládání na dálku. Stejným způsobem by bylo možno použít partnera RobotComm, ale servis **Drive** má ošetřeny konfliktní stavy a protichůdné příkazy a je tedy pro použití bezpečnější (robot se například nenechá poslat ze schodů).
4. Poslední z partnerů servisu **Main** je služba **Webcam**. Z této služby je získáván a streamován obraz pro webové uživatelské rozhraní.

Rozhraní se serverem

Rozhraní mezi mozkem robotu a domácím serverem je v současnosti implementováno pomocí příkazové řádky. Při implementaci ve skutečném inteligentním domě by bylo vhodné komunikaci přemostit za použití některé verze SOAP (*Simple Object Access Protocol*) protokolu. V takovém případě je možné použit stávající kód pro komunikaci přes příkazovou řádku (popsané v následující podkapitole) a serializovat jej do XML příkazů. Pro tento účel je servis vybaven portem pro přijímání a deserializaci takových zpráv. Přesná forma i protokol vzájemné komunikace závisí na software a struktuře inteligentního domu. Z tohoto důvodu bylo navrženo jednoduché konzolové rozhraní. Inteligentní dům může navíc pro své potřeby získávat aktuální obraz z videokamery robotu pomocí servisu **Webcam**.

Příkazová řádka

Konzole servisu **Main** slouží k získávání a zprostředkování informací o robotu a k zasílání příkazů robotu. V současnosti jsou implementovány následující funkce:

- `help` – vypíše možné příkazy a jejich krátký popis včetně možných parametrů a zkratek.

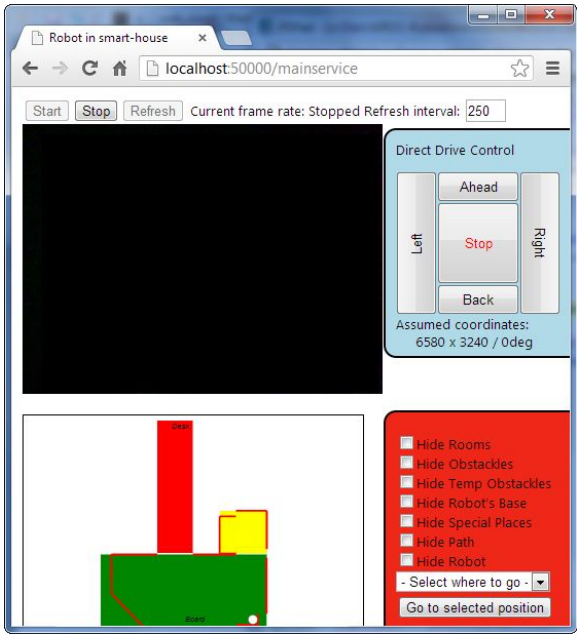
- `fetch` – získá informace o stavu senzorů robotu (ze servisu **RobotComm**). Z důvodu uvolnění komunikační sběrnice po dobu „*pollingu*“ dat probíhá získávání těchto údajů asynchronně. Asynchronně proto, že je očekáván servis **Main** a servis **RobotComm** na různých fyzických stanicích. Nadále je vypsána aktuální poloha robotu.
- `position` – vypíše pouze polohu robotu a jeho natočení, pokud jsou tyto informace známy. Poloha je vypsána formou souřadnic v kartézské soustavě. Jednotka je relativní vůči nastaveným parametrům domu a rozměrům robotu, rozlišení souřadnic je ve formátu *double*. Nulová hodnota natočení robotu ukazuje v kladném směru osy *y*. Kladná hodnota natočení narůstá proti směru hodinových ručiček. Například hodnota 270° odpovídá natočení v kladném směru osy *x*. Hodnota natočení je vždy v rozmezí 0° - 360°, její rozlišení je typu *double*.
  Pozn.: Pojmem „souřadnice“ se v této práci označuje buď poloha v kartézské soustavě souřadnic nebo poloha v kartézské soustavě souřadnic ve spojení s údajem o natočení robotu.
- `goto` `` `název` `` – přikáže robotu nalézt trasu a docestovat do určené místnosti nebo na daný „významný“ bod (v takovém případě robot dojede na souřadnice tohoto bodu). Název všech takových bodů i místností musí být jednoslovný a v případě „významných“ bodů unikátní. Více místností může mít stejný název - v takovém případě je robot navigován do nejbližší z nich. Více o navigaci robotu a nastavení mapy domu naleznete v kapitolách 6.5.1 a 8.2.1. V případě, kdy robot nenalezne možnou trasu nebo při pokusu na dané místo docestovat selže, je o jeho neúspěchu uživatel informován. Obdobně je uživateli sděleno úspěšné dokončení. Navíc jsou ze servisu **Localization** průběžně přijímány a zobrazovány informace o tom, kterými přechody mezi místnostmi robot plánuje projet. Na základě těchto dat může inteligentní dům postupně robotu uvolňovat cestu (například otevřením automatických dveří).
- `goto` `` `x` `` `` `y` `` – slouží pro vyžádání přesunu robotu na konkrétní souřadnice. Funkčnost tohoto příkazu je obdobná jako u příkazu `goto` `` `název` ``.
- `reset` – v případě, kdy se robot ztratí a není schopen nalézt cestu do své dokovací stanice, musí na ni být přemístěn manuálně uživatelem. V okamžiku, kdy se nachází ve své základně (výchozí poloze) jej použití tohoto příkazu informuje o tomto faktu. Vnitřní údaj o poloze robotu se nastaví na výchozí a některé dočasné překážky se vymažou z paměti robotu.
- `stop` – okamžitě zastaví robot při právě prováděné úloze. Uživatel nebude informován o selhání dříve zadaného úkonu.

- `ahead`, `back`, `left`, `right` – tyto příkazy jsou ze skupiny operací *direct drive* a slouží k přímému navádění robotu. Při jejich zavolání se případně ukončí prováděný příkaz `goto`. Bezpečnostní mechanismy robotu zůstavají aktivní, čímž se snižuje riziko jeho poškození nevhodným uživatelským příkazem. Je na zvážení, zda by nebylo vhodné vytvořit obdobnou sadu příkazů, obcházející tyto bezpečnostní mechanismy. Uživatel nebo domácí server by tak mohl například osvobodit uvíznutý robot. Stává se totiž, že se roboty podobného druhu například navinou na záclonu, koberec nebo elektrické kabely a nejsou schopny se samostatně uvolnit.

Webové rozhraní

Důležitým aspektem autonomního robotu v inteligentním domě je schopnost vzdálené kontroly. Uživatel se tak může například z druhého konce světa přesvědčit, že je doma vše v pořádku. V této podkapitole je nejprve vysvětleno ovládání internetového rozhraní a následně popsány principy jeho funkčnosti a způsob implementace.

Na Obr 7.2 je uživatelské rozhraní zobrazeno. Z důvodů popsaných níže v současné době funguje spolehlivě v internetovém prohlížeči Google Chrome. Plné funkcionality je možné dosáhnout také v některých verzích prohlížečů Mozilla Firefox a starších verzích prohlížeče Internet Explorer. Prohlížeč Opera nebyl testován. Doporučeno je použití zmiňovaného Google Chrome.



Obr 7.2: Webové rozhraní domácího robotu

K tomuto rozhraní je možné se připojit zadáním adresy webového serveru a přidáním */mainservice*. Cestu URL, na které se tato stránka nachází, je možné změnit ve zdrojovém kódu aplikace. Uživatel se musí autentizovat jménem a heslem a musí být

v konfiguraci serveru oprávněn k tomuto úkonu. Zvolená autentizace pracuje na principu NTLM (*NT LAN Manager*) a případně SID (*remote Security Identifier*).

V levé horní části rozhraní se nachází výstup streamovaného videa z kamery umístěné na robotu. Obraz je možné spustit pomocí tlačítka *Start* (při načtení stránky je video ve výchozím stavu spuštěno) a zastavit pomocí tlačítka *Stop* nad oknem kamery. Použitím tlačítka *Refresh* se načte pouze jeden aktuální záběr z kamery. Dále se zde nachází selektor obnovovací frekvence obrazu a ukazatel aktuální rychlosti ve snímcích za sekundu. Maximální dosažená rychlost streamovaného videa, s použitím hardware zmíněného v kapitole 8.1, byla zhruba 40 fps.

V pravé horní části prohlížeče se nachází ovládání pro přímý pohyb robotu. Jedná se o tlačítka pro pohyb dopředu, dozadu, rotaci vlevo, rotaci vpravo a okamžité zastavení diferenciálně řízeného robotu. Pomocí tlačítek se aktivuje funkce *Direct Drive*, která je nastíněna v předchozí podkapitole Rozhraní se serverem. Pod ovladačem přímého pohybu se nachází výpis aktuálních souřadnic robotu. V tomto okně se zobrazí *„unknown“* v situaci, kdy se robot ztratí. Tento údaj bývá obnovován společně s informacemi o mapě.

Mapa prostředí, ve kterém se robot pohybuje, je zobrazena v levém dolním rohu webového rozhraní. Nachází se zde obrázek jednotlivých místností. Ty jsou ve výchozím nastavení průhledné, s tenkým šedým okrajem, ale může jim být přiřazena specifická barva pomocí konfiguračního souboru servisu **Localization**. Jednotlivé statické překážky typu bod a stěna jsou zobrazeny červeně. Dočasné překážky, na které robot v průběhu své cesty narazí, jsou v této mapě zobrazeny oranžovou barvou. Dokovací stanice (nebo výchozí poloha robotu) je v mapě zobrazena zeleným písmenem *H* a to tak, že je dokující robot plně překrývá. V průběhu dokování (nebo ve výchozím stavu) robot „kouká“ směrem k horní hraně písmene *H*. Dolní okraj písmene je pro přesnost označen puntíkem stejné barvy. Značka základny robotu má stejnou velikost jako robot samotný. Údaj o velikosti robotu je získán z nastavení rozměru robotu v servisu **RobotComm**.

Velikost každého „významného“ bodu v mapě je možno nakonfigurovat samostatně. Přesná poloha, na kterou se robot pokusí najet, je vyznačena křížkem. Název místnosti (za předpokladu, že je místnost pojmenována) lze zobrazit najetím kurzoru myši na tuto místnost v mapě.

V pravé spodní části webové stránky se nachází selektor zobrazených prvků mapy. Pomocí jednotlivých zaškrtávacích rámečků je možné je podle potřeby skrývat. Pro navigaci a příkaz robotu k dojetí na danou pojmenovanou pozici v mapě slouží zobrazené roletkové menu, které je automaticky naplněno názvy všech místností a „významných“ bodů v mapě. Příkaz pro dojetí na danou polohu se potvrdí tlačítkem *Go to selected position*. Pokud je robot schopen na toto místo docestovat, zobrazí se v mapě dráha z jeho současné polohy na pozici cílovou.

Zbývající odstavce této podkapitoly se zabývají vnitřní funkčností klientské stránky webového prohlížeče.

Servis **Main** po svém startu spustí na nakonfigurovaném portu HTTP server, který zprostředkovává webové uživatelské rozhraní. Vzhledem k tomu, že stav (veřejná data) služby **Main** je v MRDS nativně udržován a přenášen v podobě XML, je pro jeho převod a pro zobrazení webového uživatelského rozhraní použito XSLT (*Extensible Stylesheet Language Transformations*). Při zobrazení webové stránky jsou pak data transformována do podoby HTML. Přesto, že je XSLT otevřený a relativně starý formát pro transformaci XML, není plně podporován prohlížečem Firefox (produkt společnosti Mozilla nepodporuje transformed Javascript).

Data pro webové rozhraní (HTML Resources) je možné přikompilovat napevno k binárnímu souboru servisu **Main**. Vzhledem k tomu, že výstup této práce není konečné a vizuálně optimální řešení, je transformační XSLT soubor přiložen v textové podobě k souborům webového serveru. Více o jeho použití a úpravě se nachází v kapiole 8.2.5. Soubor samotný je pojmenován *WebService.xslt* a nachází se v příloze této práce.

Dynamické funkce stránky jsou implementovány v jazyce Javascript. Příkazy *Direct Drive* jsou považovány za nebezpečné, pokud pochází ze zdroje náchylného na výpadek spojení, a servis **Main** je po třech sekundách automaticky přerušuje. Proto při stisknutí některého z tlačítek *Direct Drive* ve webovém prohlížeči je příslušný příkaz zasílán opakovaně v intervalu dvou sekund.

Pohyb se ukončí uvolněním daného tlačítka nebo sjetím kurzoru myši z jeho povrchu. Tuto funkcionalitu by bylo možné rozšířit čtením klávesnice pomocí Javascriptu v okně prohlížeče.

Komunikace mezi internetovou stránkou a serverem robotu je realizována pomocí metody POST a zasíláním XML zpráv. Aby nebylo nutné při každé změně znovu načítat celou stránku je pro implementaci komunikace použit AJAX (*Asynchronous JavaScript and XML*). Servis **Main** reaguje na různé zprávy POST a případně navrací vyžádaná data. Těmito daty mohou být informace o změně v dočasných překážkách, poloze robotu nebo například souřadnice plánované trasy robotu.

Pro snížení zátěže serveru byl pro grafické renderování mapy zvolen formát SVG (*Scalable Vector Graphics*) a renderování tedy probíhá přímo v prohlížeči. Server tak zasílá veškeré infomace o mapě ve formátu XML. Ty jsou následně transformovány do formátu SVG a zobrazeny v několika vrstvách v prohlížeči. Tyto průhledné vrstvy odpovídají jednotlivým zaškrtávacím rámečkům. Informace týkající se mapy jsou v pravidelných intervalech (každých 5 sekund) stahovány ze serveru. Je nutné podotknout, že novější verze prohlížeče Internet Explorer (od implementace nového grafického enginu), zatím nepodporují tento formát plně. Prohlížeč Google Chrome (WebKit) není schopen s formátem SVG pracovat dynamicky a proto musí být pro jeho bezchybnou funkci použito několik složitých programátorských úkonů (workaround).

Pro stahování a přenos obrazu z kamery umístěné na robotu je použita technologie HTTPQuery. Požadavky na příslušný záběr jsou tak ze servisu **Main** přeposílány přímo na HTTP rozhraní servisu **WebCam**, který odpovídá žadateli příslušným snímkem z kamery. Pokud se tedy oba servisy nacházejí na rozdílných stanicích, je zvolena optimální cesta streamovaného obrazu a dochází tak k minimálnímu vytěžování služby **Main**. Je nutné podotknout, že pokud jedna ze zmíněných stanic nepodporuje certifikaci SID, může být uživatel vyzván k autentizaci dvakrát. Jednou HTTP serverem servisu **Main,** podruhé HTTP serverem servisu **WebCam**. Podrobnější popis funkčnosti přenosu obrazu se nachází v podkapitole 7.3.4.

7.3.2 Servis Localization

Tento servis obstarává lokalizaci robotu, to jest udržování jeho aktuální polohy a jeho navádění na globální úrovni (kapitola 6.5.1). V této podkapitole jsou nejprve zmíněny jeho partnerské služby a následně popsána jeho funkcionalita.

Partnerské vztahy

1. Jedním ze dvou „partnerů“ servisu **Localization** je servis **RobotComm**. Z této služby je za pomoci operace *Subscribe* online získávána každá aktuální změna polohy (výstupy funkcí *Angle* a *Distance*) a každá změna stavu senzorů dotyku. Změnou stavu je myšlena aktivace nebo deaktivace dotekových senzorů na předním nárazníku robotu, detekce virtuální zdi nebo aktivace jednoho ze senzorů převisu (čtyři přední *cliff* senzory na nárazníku robotu a senzory prověšení krajních nebo středového kola). V poslední řadě je od služby **RobotComm** získávána informace o přijatém IR příkazu z dálkového ovládání. Z důvodu relativně vysokých komunikačních toků je vhodné, aby se obě služby (**Localization** a **RobotComm**) nacházely na stejném fyzickém zařízení.
2. Druhým „partnerem“ je servis **Drive**. Tato služba je použita pro navádění na lokální úrovni. V okamžiku, kdy služba **Localization** dostane příkaz k přemístění robotu na danou polohu, vypočte vhodnou trasu a přikáže servisu **Drive** dojet k nejbližšímu (a poté každému následujícímu) přechodu mezi místnostmi na této trase.

Počáteční stav

Při startu jsou servisu **Localization** předány informace o rozložení místností domu, známých statických překážkách a obrázkových kódech. Tento seznam sestává z jednotlivých dat serializovaných do formátu XML. Formát konfiguračního souboru je popsán v podkapitole 8.2.1. Při startu této služby je tento seznam načten a převeden postupně do jiných stavových proměnných tohoto servisu. Jsou generovány jednotlivé

přechody místností a vypočteny sousedící dvojice přechodů a jejich vzájemné vzdálenosti.

Pro generování přechodů je využita open source knihovna *Clipper*, jejímž původním autorem je pan Angus Johnson [39] (licenční smlouva je přiložena u zdrojových kódů). Tato knihovna slouží pro efektivní a rychlou práci s polygony a jedna z jejích distribucí je v jazyku C#. V této práci je využita především pro nalezení překrývajících se místností. Pokud se místnosti v některém bodě překrývají, symbolizuje to pro robot možný přechod mezi odpovídajícími polygony. Funkcionalita knihovny *Clipper* byla pro potřeby této práce v souladu s jejím licenčním ujednáním rozšířena o schopnost výpočtu středového bodu konvexního polygonu a schopnost rozhodnout, zda je určitý bod součástí daného polygonu.

Z konfiguračního souboru jsou mimo informací o místnostech načteny také:

1. Výchozí souřadnice robotu. Ty nejsou použity, pokud se v některé z místností nachází dokovací stanice robotu.
2. Údaje o výchozích hodnotách intenzit cíle a dočasných překážek. Jejich konfigurace závisí na rozměrech robotu a jednotkách veličiny rozměru mapy ve spojení s intenzitou ostatních statických překážek (více v podkapitole 8.2.1).
3. Údaje o aproximaci informací o změně polohy robotu (více v podkapitole 8.3).

Příkazy Subscribe

Servis **Localization** při své inicializaci zažádá servis **RobotComm** o periodické zasílání změn následujících skupin senzorů:

1. Senzory překážek – těmi jsou přední dotekové senzory umístěné na nárazníku robotu, detekce virtuální stěny, detektor stěny v pravé přední části robotu a detektory převisu.
2. Detekce pohybu – služba **RobotComm** zasílá informace o změně natočení a délce pohybu robotu od poslední aktualizace.
3. Dálkové ovládání – služba **RobotComm** zasílá informace o přijatých příkazech z dálkového ovládání robotu Create.

Sebelokalizace

Služba **Localization** je především zodpovědná za udržování souřadnic robotu a jejich chyby. Při jakémkoliv pohybu robotu se chyba informace o poloze lineárně zvyšuje s koeficientem závislým na velikosti změny polohy od poslední notifikace. Nebo snižuje v případě absolutní lokalizace robotu. Při zvednutí robotu ze země nebo jeho pádu či převrácení je informace o poloze ztracena úplně. Chyba určené polohy udává poloměr kruhu v okolí současných souřadnic, ve kterém se robot nachází. Jedná se o nesystematickou chybu. K jejímu výpočtu je použita rovnice (5). Hodnotu ϵ_e je třeba určit na základě použitého robotu (kapitola 8.3), programu aplikace je předána

jako proměnná *MovementErrorRatio*. Chyba polohy robotu po absolutní lokalizaci za pomoci rozpoznání obrázkového kódu je nastavena na hodnotu přesnosti abslutní lokalizace, uložené v proměnné *VisionAccuracy*.

Veškeré informace o změně polohy ze servisu **RobotComm** prochází statickou transformací pro omezení vlivu systematických chyb pohybu. Experimentálně bylo zjištěno, že se vlivem nedokonalosti hardware robotu výstupní údaje o jeho pohybu liší od reality o konstantní velikost. Bylo prokázáno, že v rozmezí rychlostí motorů robotu od 20% do 80% maximální rychlosti je koeficient této chyby neměnný (to koresponduje s teorií popsanou v 6.1.1). Tato transformace se nastavuje v konfiguračním souboru služby **Localization** (více v podkapitole 8.2.1). Proměnné pro výpočet této transformace jsou čtyři:

1. *RotationRightApproximation* udává procentní odchylku od správného směru natočení robotu při jedné plné otáčce kolem své osy směrem doprava (ve směru hodinových ručiček). Povolené hodnoty jsou v intervalu -1 až 1. Například hodnota +0,5 opravuje přetáčivý pohyb robotu o polovinu otočení: z 360° (výstup z enkodérů kol) na 180° (skutečné pootočení). Záporné hodnoty tedy odpovídají nedotáčivému robotu a kladné hodnoty robotu přetáčivému.
2. *RotationLeftApproximation* analogicky stejná hodnota jako v předchozím bodu, ale pro rotaci robotu směrem doleva.
3. *StraightAngleApproximation* udává hodnotu poloměru kruhu, který robot opisuje při „přímém“ pohybu. Kladná hodnota tohoto koeficientu značí, že robot při pokusu jet rovně zatáčí směrem doleva. Záporná hodnota značí, že robot zatáčí doprava.
4. *StraightApproximation* udává chybu ujeté vzdálenosti po přímce v desetinách procenta. Pro vzdálenost 1 metr je to počet milimetrů, které robot nedojede (záporné hodnoty) nebo přejede (kladné hodnoty).

Je důležité poznamenat, že hodnoty v bodech 3 a 4 by měly být evidovány jednou pro pohyb dopředu a podruhé dozadu. Vzhledem k faktu, že se robot při autonomní jízdě nepohybuje dozadu (kromě krátkého popojetí od nově nalezené překážky), je evidence těchto hodnot zbytečná.

Servis **Localization** obsahuje port pro přijímání příkazů dálkového ovládání robotů Roomba. Myšlenka lokalizace tímto způsobem spočívá v tom, že některé nebo všechny místnosti domu jsou vybaveny vysílači s různým kódováním. Těchto odlišně kódovaných vysílačů může být až 248 (celkem je 254 možných příkazů dálkového ovládání, ale šest příkazů není možné z uživatelského programu zachytit). Robot po přijetí daného signálu může ověřit, zda se nachází ve správné místnosti. V případě, že po udanou dobu není schopen zachytit očekávaný signál, robot vyhodnotí, že se ztratil. Implementace této funkcionality je v kódu předpřipravena, ale pro potřeby této práce se nepodařilo zajistit dálkové ovládání robotu Roomba kompatibilní s roboty Create.

Dobíjecí základna robotů Create vysílá naváděcí paprsky pro úspěšné zadokování robotu k nabíjení. Těmto paprskům program robotu za normálních okolností nevěnuje pozornost. V případě, že hodnota nabití baterií robotu klesne pod 150 mAh a robot neprovádí žádný příkaz, pokusí se k této základně automaticky připojit. V případě, kdy ztratí informaci o poloze a má kontakt se zemí, zažádá uživatele o přemístění do této stanice. Pokud během několika minut nedojde k přemístění, pokusí se nalézt dokovací stanici a připojit se k ní autonomně. Nalezení základny probíhá tak, že nejprve se robot pokusí sám dostat do místnosti, ve které se dobíjecí stanice nachází, a poté je spuštěna interní funkce *„force dock“*, která nalezne dokovací stanici a připojí se k ní automaticky. Při zadokování jsou souřadnice robotu automaticky nastaveny na výchozí hodnotu. Implementace této funkcionality je opět v kódu předpřipravena, ale pro potřeby této práce se nepodařilo zajistit dokovací stanici kompatibilní s roboty Create (ty je možné zakoupit pouze v Americe).

Protože servis **Localization** může přijímat údaje o své poloze z několika zdrojů, bylo by vhodné navrhnout algritmus jejich fúze. V současnosti jsou údaje snímání absolutní hodnoty brány za správné a aktuální poloha robotu je jimi přepsána. To nezpůsobuje znatelnou chybu lokalizace, dokud robot stojí. Z důvodu použití nekvalitní *CMOS* kamery, robot občas ztratí povědomí o své poloze. To je způsobeno tím, že u ní při pohybu robotu dochází, vlivem postupné expozice řádků, ke zkosení obrazu a tím i ke špatné estimaci polohy.

Možným způsobem provedení datové fúze by bylo použití Kalmanova filtru. V [51] je použit v obdobné situaci. Alternativou ke Kalmanovu filtru je metoda dvojitého exponenciálního vyhlazování [52].

Příkazy Goto a Stop

Pro požadavek přijetí robotu na dané místo obsahuje servis **Localization** speciální port. Na tomto portu přijímá zprávy *Goto* s přesně specifikovanými požadovanými souřadnicemi nebo jménem vybraného místa, na které má robot dojet. Pojmenovaným místem může být buď název některé místnosti nebo název „významného“ bodu v některé z místností. Při přijetí takovéhoto požadavku je automaticky upraven stav robotu na stav „provádění příkazu“. Pokud již některý příkaz probíhá, je nejprve zrušen pomocí příkazu *Stop*. Samotný příkaz *Stop* může být také zavolán jiným servisem.

Při přijetí příkazu *Goto* je spuštěno nové vlákno, ve kterém se provádí výpočet optimální trasy na globální úrovni metodou *A**. Při tomto výpočtu se hledá nejkratší sekvence přechodů (nejkratší na jednotky vzdálenosti, nikoliv na počet přechodů), kterými musí robot procestovat pro dosažení cíle. Jak již bylo zmíněno, více místností může být označeno stejným jménem. V takovém případě je nalezena cesta k nejbližší místnosti s daným identifikátorem, respektive cesta na přechod s danou místností. Pokud není nalezena žádná cesta na požadovanou pozici, je jako odpověď na příkaz

Goto odeslána zpráva typu *Fault*. Po úspěšném dokončení navádění je na volající port odeslána zpráva typu *Success*.

Pro každou dvojici přechodů ve vygenerované cestě je postupně zaslán některý příkaz servisu **Drive**. V první fázy pohybu je zaslán inicializační příkaz. V tom jsou nastaveny statické a dočasné překážky, které se nachází v současné a následující místnosti. Dále je s inicializačním příkazem zaslána aktuální pozice robotu a pozice cíle. Služba **Drive** sama provádí nastavování rychlostí motorů robotu tak, aby se robot dopravil na požadovanou pozici. Samotný servis **Drive** ale neobsahuje zpětnou vazbu o změně polohy robotu. Tuto zpětnou vazbu obstarává servis **Localization** zasíláním aktualizačních příkazů. Pomocí nich se upravují souřadnice robotu tak, jak jsou vnímány servisem **Drive**, a informace o nových dočasných překážkách. V případě objevení nové překážky zašle servis **Localization** servisu **Drive** ihned informace o této překážce a příkaz *BackUp* (robot popojede kousek zpět). Navádění je po přerušení možné spustit buď opětovnou inicializací nebo pomocí příkazu *ContinuePathfind*.

V případě, že robot uvízne v lokálním minimu, je ze služby **Drive** navrácena hodnota *Fault*. Servis **Localization** provede pokusy o nápravu této situace metodami popsanými v podkapitole 6.5.3 - Výpočet vektoru pohybu.

Přidání překážek

Robot při jízdě může za pomoci některého ze senzorů nalézt překážku. V takovém případě je poloha nalezené překážky porovnána se seznamem statických i dočasných překážek. Pokud se v okolí o velikosti chyby polohy robotu nachází některá ze známých překážek, robot dojde k závěru, že nová překážka je jednou ze stávajících. Poloha robotu je upravena na dosah známé překážky a perzistence překážky je inkrementována. Pokud je překážka vyhodnocena jako nová, je přidána do seznamu dočasných překážek s hodnotou perzistence 1.

Pokud k detekci překážky dojde jedním z dotekových senzorů, je servisu **Drive** vyslán příkaz *BackUp*. Tím se robot od dané překážky vzdálí natolik, že ji již nedetekuje. Pokud dochází k navádění robotu na lokální úrovni, je o existenci této nové překážky informován servis **Drive** a navádění je opět spuštěno příkazem *ContinuePathfind*.

7.3.3 Servis Drive

Navigaci na lokální úrovni popsanou v kapitole 6.5.2 obstarává servis **Drive**. Mimo to je možné využít funkci *Direct Drive* pro přímé ovládání robotu. Funkce přímého ovládání robotu jsou: pohyb rovně dopředu nebo dozadu a rotace kolem osy robotu vpravo nebo vlevo.

Partnerské vztahy

Služba **Drive** využívá partnerských vztahů s následujícími dvěma servisy:

1. **RobotComm** slouží jako prostředník mezi službou **Drive** a skutečným hardware robotu (konkrétně motory robotu) nebo robotem v simulátoru. Tato koncepce umožňuje užití servisu **Drive** v simulátoru i v robotu bez změny kódu.
2. Servis **Localization** je pomocí funkce *Subscribe* zažádán o zaslání aktualizací polohy robotu. Tyto aktualizace jsou zasílány pouze pokud je robot v režimu lokálního navádění.

Příkazy *Direct Drive* jsou přijímány především od služby **Main**. Ta ale není partnerem servisu **Drive**.

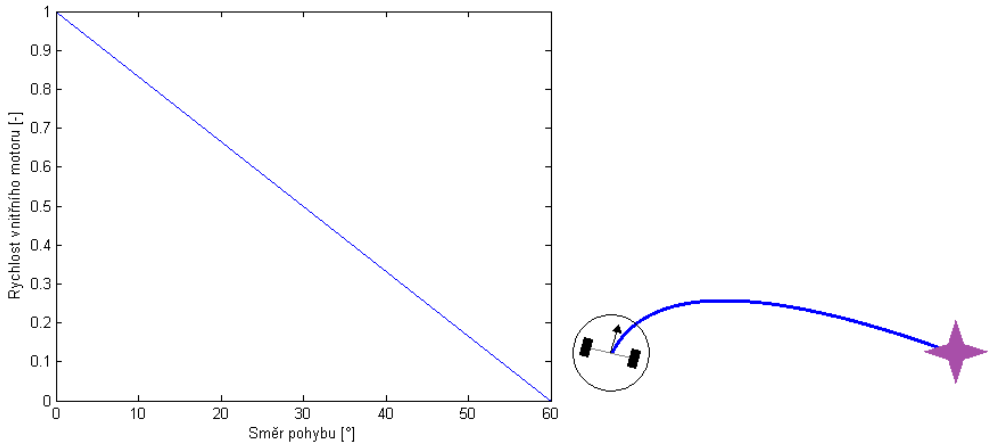
Pohyb robotu

Výpočet trajektorie pomocí metody vektorových polí je popsán v podkapitole 6.5.3 - Výpočet vektoru pohybu. V případě přijetí příkazu k lokálnímu navádění je nejprve vypočten směr požadovaného pohybu ve stupních pomocí funkce *CalculateDirectionDegrees*. V případě, že požadovaný vektor pohybu směřuje od cíle, je ověřeno, zda se robot nenachází v lokálním minimu (tato problematika je podrobně rozvedena v následující podkapitole). Vektor pohybu směřuje od cíle například, když se cíl nachází v bodě (1, 0), robot v počátku kartézské soustavy souřadnic a požadovaný vektor pohybu směřuje do 2. nebo 3. kvadrantu.

Relativní směr požadovaného pohybu robotu je vypočten jako rozdíl mezi aktuálním úhlem natočení robotu a požadovaným úhlem pohybu robotu k cíli vypočteným funkcí *CalculateDirectionDegrees*. V závislosti na relativním požadovaném směru pohybu robotu jsou ve funkci *CalculateMotorSpeeds* vypočteny rychlosti obou motorů. V případě, kdy není hodnota relativního směru požadovaného pohybu v rozmezí <-60, 60> stupňů (tj. robot se musí pohybovat o více než 60° doleva nebo doprava), přejde servis **Drive** do stavu *TurnBeforePathfinding*, ve kterém se rotací na místě robot otočí do požadovaného směru. Tím se předchází pohybu ve velkých obloucích v případě, kdy je cíl za robotem. Rotace robotu na místě se vždy provádí poloviční požadovanou rychlostí z důvodu zvýšení přesnosti lokalizace a robot vždy před zahájením rotace vyčká půl sekundy na úplné zastavení.

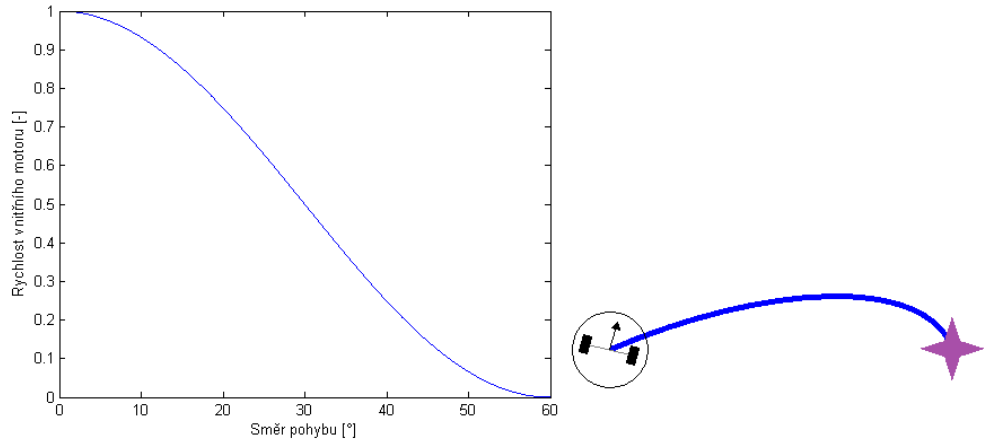
Při pohybu směrem k cíli se robot, naváděný pomocí vektorových polí, pohybuje po parabolických křivkách. Vnější motor diferenciálně řízeného podvozku má při takovém pohybu vždy nastavenu maximální povolenou rychlost a vnitřní motor rychlost úměrně menší (při pohybu po křivce směrem doprava se vnějším motorem myslí levý a vnitřním pravý motor). Pro výpočet poměru rychlostí obou motorů bylo vyzkoušeno několik metod, popsaných v následujících odstavcích. Jednotlivé metody lze otestovat odkomentováním příslušného příkazu ve funkci *CalculateMotorSpeeds*. Jako nejlepší

kandidát pro použití u robotu v inteligentním domě byla z důvodů popsaných níže zvolena funkce *CoefLin*. Následující zmíněné funkce by pro dosažení optimální funkcionality bylo možné kombinovat, je ale třeba dbát na to, aby byl průběh transformační funkce hladký, protože prudké změny rychlostí poškozují motory robotu.



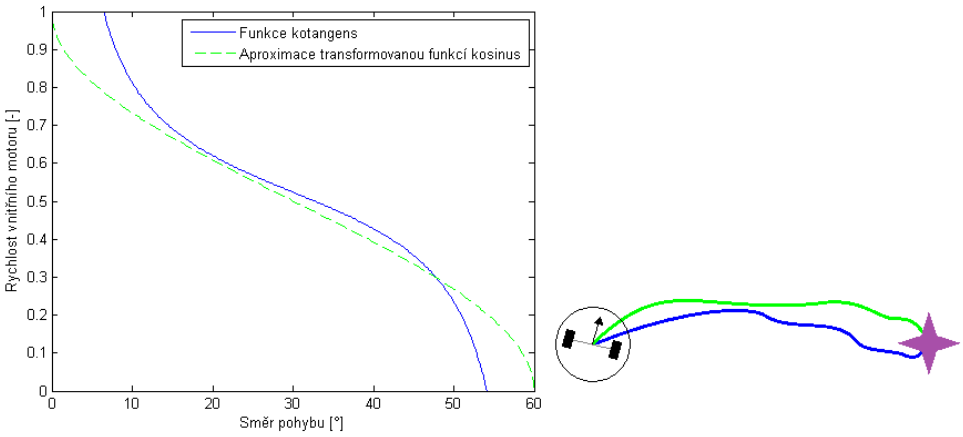
Obr 7.3: Lineární metoda výpočtu rychlosti vnitřního kola podvozku

Nejjednodušším způsobem výpočtu poměru rychlostí jednotlivých motorů je lineární závislost mezi požadovaným relativním úhlem pohybu a rychlostí vnitřního kola. Tato funkcionalita je implementována funkcí *CoefLin* a je znázorněna na Obr 7.3. Na tomto obrázku jsou na ose *y* hodnoty rychlosti vnitřního kola robotu v rozmezí od 0 do 1, kde hodnota 1 odpovídá rychlosti kola vnějšího. Na ose *x* je ve stupních znázorněn úhel relativního směru pohybu. Při použití této metody robot zprvu pomalu zatáčí a velice brzy nabere stabilní kurz přímo k cíli. Pomyslná trajektorie robotu je znázorněna v pravé části Obr 7.3. Použitím této metody nedochází k překmitu a v porovnání s ostatními metodami je výpočetně nejsnažší. Tato metoda je tedy vhodná pro lokální navádění autonomního robotu v inteligentním domě.



Obr 7.4: Metoda výpočtu rychlosti vnitřního kola využívající funkci sinus

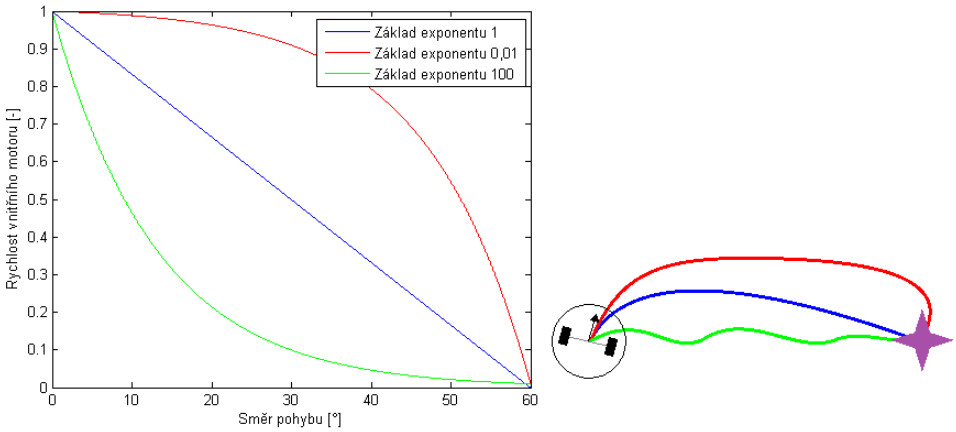
K výpočtu druhé testované metody je možné použít funkci *CoefSin* ve zdrojových souborech servisu **Drive**. Tato metoda využívá goniometrické funkce *Sinus*, jejíž průběh je znázorněn na Obr 7.4. Při použití této metody robot zatáčí rychle v případě, kdy je požadovaný relativní úhel pohybu velký, a naopak pomalu, když je cíl přímo před robotem. To může v některých případech vést k minutí cíle a k nutnosti se po jeho přejetí otočit a pokusit se o přístup znovu.



Obr 7.5: Metoda výpočtu rychlosti vnitřního kola využívající funkci tangens

Třetí testovaná metoda vychází z použití goniometrické funkce *tangens* (*CoefTan*). Nevýhodou této funkce je její nedefinovanost v oblastech asymptot. Pro řešení tohoto problému by ji bylo možné matematicky transponovat do požadované výseče. K velice podobnému výsledku ale vede její uplná náhrada transformovanou funkcí *cosinus*. Na Obr 7.5 jsou zobrazeny obě tyto závislosti společně s názornou trajektorií pohybu robotu. Alternativa této metody byla úspěšně použita v [34], pro potřeby inteligentního domu ale vhodná není, protože se na rozdíl od použití v robotickém fotbalu nejedná o příliš dynamické prostředí s neustále se měnící polohou cíle.

Poslední testovanou metodou je využití exponenciální funkce dělené požadovaným relativním úhlem pohybu. Ve zdrojovém kódu servisu **Drive** je funkce pojmenována *CoefExp*. Základem exponentu je konstantní hodnota a exponent je požadovaný relativní úhel pohybu. Při položení základu exponentu rovno 1, vede tato metoda ke stejnému výsledku jako metoda *CoefLin*. Se zvyšujícím se základem exponentu se průběh funkce blíží průběhu logaritmickému, což vede na prudké zatáčení a vysokou oscilaci robotu. Pro snižující se základ exponentu je robot při zatáčení velice laxní a zpravidla dochází k objíždění cílového bodu po spirále. Tato metoda je při použití základu exponentu různého od 1 nevhodná. Různé výstupy funkce *CoefLin* jsou znázorněny na Obr 7.6.



Obr 7.6: Metoda výpočtu rychlosti využívající exponenciální funkci

Lokální minima

V případě, kdy požadovaný směr pohybu směřuje od cílového bodu, je ve funkci *CalculateMotorSpeeds* provedena detekce lokálního minima metodou popsanou v podkapitole 6.5.3 - Výpočet vektoru pohybu. V případě pozitivní identifikace lokálního minima je na port servisu inicializujícího lokální navádění servisu **Drive** zaslána odpověď typu *Fault*. Primárně je volající službou servis **Localization.** Ten při přijetí zprávy typu *Fault* provede pokus o opravu příčin vzniku lokálního minima, popsaných v podkapitole7.3.2

7.3.4 Servis WebCam

Služba **WebCam** obstarává detekci dostupných kamer připojených k počítači, připevněnému k robotu. Pokud není v konfiguračním souboru zvolena jiná volba, provede zapnutí první nalezené webové kamery. Následně spustí program *WebcamPipeServer.exe* a začne provádět stahování aktuálních snímků z aktivované kamery. Kód samotného *pollingu* obrazu je částečně převzat ze 7. tutoriálu MRDS. Tento servis nemá žádné partnery.

Servis **WebCam** podporuje jakýkoliv hardware kamer, pro který je na příslušné stanici instalován ovladač kompatibilní s lokální distribucí operačního systému Windows. Servis není v současnosti možné z důvodu jeho návrhu jednoduše použít v simulátoru MRDS, protože pro jeho běh nelze spustit vhodný *dispatcher*. Pro spuštění webové kamery v simulátoru je nutné ji přidat manuálně za běhu simulace postupem popsaným v tutoriálu MRDS „*Custom Simulated Entities*“. Tato funkcionalita bude pravděpodobně v příštích verzích MRDS přidána.

Servis **WebCam** naslouchá na portu *HTTPQuery* a případně odpovídá zasláním nejnovějšího snímku z kamery. Pro zaslání snímku musí být žadatel autentizován.

7.3.5 Servis ImageRecognition

Servis **ImageRecognition** provádí absolutní zjišťování polohy robotu metodou popsanou v kapitole 6.3. Tento servis definuje důležité třídy pro práci s rozpoznávanými vzory. Konkrétně se jedná o třídu *GlyphCode*, která udržuje informaci o „hodnotě“ daného vzoru a zprostředkovává metody pro práci s ní. Třída *Glyph*, tvoří obal třídy *GlyphCode*. Rozšiřuje ji o informace o poloze a rotaci vzoru a pomocí statické proměnné udržuje informaci o jeho rozměrech. Třída *Camera* slouží k reprezentaci vzájemné polohy robotu a kamery.

Partnerské vztahy

Tato služba není využívána žádným jiným servisem. Navazuje však partnerský vztah se servisy:

1. Obrázky pro detekci vzorů jsou získávány ze servisu **WebCam**. V případě jeho nenalezení, se služba **ImageRecognition** sama ukončí. Snímky jsou získávány ve formátu *BMP*. Servis **ImageRecognition** spoléhá na to, že získané snímky budou předzpracovány.
2. Službě **Localization** jsou zasílány informace o estimované poloze robotu. Ta je nadále zodpovědna za jejich provedení datové fúze.

Inicializace

Při startu je načten inicializační soubor *ImageRecognition.state.xml*, ze kterého jsou získány konstanty pro proces počítačového vidění (kapitola 8.2.2) a případně i databáze vzorů a infomace o parametrech kamery. Poté zažádá servis **Localization** o zaslání seznamu rozpoznávaných vzorů, kterým je stávající seznam doplněn (preferované řešení je ukládat vzory do konfiguračního souboru servisu **Localization**). Ze servisu **Localization** je získána také informace o kameře robotu a rozměrech rozpoznávaných vzorů. Pro vizuální kontrolu jsou všechny rozpoznávané vzory vypsány na standardní výstup.

Rozpoznávání

Po fázi inicializace je spuštěn exkluzivní cyklus *pollingu* snímků z kamery a jejich zpracování. Pro provádění operací počítačového vidění je využita knihovna *AForge* [50] (*Aforge* je distribuována volně i se zdrojovými kódy pod GPL licencí). Po rozpoznání vzoru je využita funkce *CoplanarPosit* ke zjištění relativní pozice mezi jím a kamerou, ze které je vypočtena pozice robotu. V případě, že stěna, na které se vzor nachází,

a odhadovaná poloha robotu svírají úhel menší než 45°, nebere se rozpoznaný vzor v potaz (*CMOS* kamera, použitá v této práci, vykazuje pro takové úhly velice špatné hodnoty). Při cyklu rozpoznávání je pro každý vzor vypočtena hodnota chyby. Ta je získána z poměru počtu bílých a černých pixelů venkovní poloviny okraje vzoru transformovaného obrázku v kroku 7 kapitoly 6.3.2. Chyba je zaslána servisu **Localization** společně s estimovanou pozicí.

7.3.6 Servis iRobotLite

IRobotLite tvoří rozhraní mezi skutečným hardware robotu a klientskou aplikací. Přestože je v této práci shrnut do jedné kapitoly, a ve zdrojových kódech je zahrnut pouze do jednoho projektu, ve skutečnosti sestává z několika proxy servisů, které obhospodařují komunikaci s robotem přes zvolené médium a zprostředkovávají data z jeho jednotlivých senzorů. Servis je vyvinut společností Microsoft a je součástí instalace MRDS. Při jeho implementaci bylo nalezeno a opraveno několik vážných chyb, například špatné návratové hodnoty funkcí *Angle* a *Distance* nebo chyba vyvolávající výjimku při připojovaní k robotu, způsobenou nemožností nastavit atribut *song*. Z tohoto důvodu byly upravené zdrojové kódy servisu **iRobotLite** přidány k této práci. Zdrojové kódy jsou součástí tutoriálu MRDS, zveřejněného společností Microsoft, a jako takové podléhají odstavci 3.b.1 licenčního ujednání, připojeného k instalaci MRDS. Tento odstavec dává svolení s jejich úpravou a distribucí.

Servis **iRobotLite** tvoří nejnižší rozhraní mezi aplikací a robotem. Za svého běhu komunikuje pouze se servisem **RobotComm**, a to buď za pomoci metody *subscribe* nebo pomocí příkazů *iRobotCommand*. Tyto příkazy jsou využívány pro nastavování rychlostí motoru nebo pro spouštění vnitřních funkcí robotu (například funkce *„force dock“*). Při spuštění aplikace v simulátoru není servis **iRobotLite** využit vůbec, protože v takovém případě služba **RobotComm** přistupuje k simulovanému hardware.

7.3.7 Servis RobotComm

Servis **RobotComm** (ve zdrojových kódech pojmenovaný *iRobotComm*) tvoří obal servisu **iRobotLite**. Tento přístup je zvolen z důvodu snadné možnosti přenést celou aplikaci do simulovaného prostředí nebo na úplně jiný druh robotu. V případě takového přechodu musí být pozměněn pouze servis **RobotComm**, což vede na vysokou modularitu a možnost opětného použití kódu v jiné robotické aplikaci.

Servis **RobotComm** obsahuje veřejnou třídu *Dimensions*, která je přikompilována k výstupnímu *dll*. Třída obsahuje několik konstantních proměnných, které udávají rozměry robotu. Při použití jiného hardware robotu je nutné tyto proměnné změnit (společně s většinou ostatních funkcionalit servisu). Tato data jsou využita jinými službami například pro výpočty při lokálním navádění.

Partnerské vztahy

Kromě partnerů se službou **RobotComm** komunikuje především servis **Main** a servis **Drive**. Služba **RobotComm** při svém spuštění inicializuje následující dva partnerské vztahy:

3. Servis **iRobotLite** je využíván pro komunikaci s hardware robotu. Při inicializaci servisu proběhne pokus o připojení k robotu. V případě použití v simulátoru je místo něj inicializován partner **SimulationEngine**.
4. Služba **Localization** je upozorňována na různé změny stavů senzorů robotu. Jako partner je přidána z důvodu možnosti reakce na stisknutí tlačítek na těle robotu klientskou aplikací.

Počáteční stav

Servis **RobotComm** při svém spuštění načte svůj stav z přiloženého konfiguračního souboru. V konfiguračním souboru jsou parametry důležité pro inicializaci spojení s robotem. Více informacím o struktuře inicializačního souboru se nachází v kapitole 8.2.2.

Na základě nastavení atributů robotu v počátečním stavu dojde při inicializaci servisu **RobotComm** k pokusu o připojení k robotu za pomocí služby **iRobotLite**. V průběhu připojování je uživatel informován o aktuálním stavu. K chybě připojení k robotu může dojít ze dvou důvodů. Buď robot není zapnutý, v tom případě je nutné vypnout tuto službu, zapnout robot a opětovně službu spustit, nebo došlo k rozsynchronizování komunikace mezi robotem a službou **iRobotLite**. V takovém případě je nutné robot i aplikaci vypnout, po 10 sekundách robot opět zapnout a následně spustit aplikaci.

Příkazy Subscribe

Služba **RobotComm** při své inicializaci zažádá servis **iRobotLite** o zasílání změn všech senzorů robotu. Tyto změny jsou ukládany do stavu servisu tak, aby jeho stav vždy odpovídal aktuálnímu stavu robotu. Při změně některého z důležitých senzorů robotu jsou zaslány notifikace ostatním servisům robotické aplikace, které si o jejich zasílání zažádaly. Při poklesu nabití baterie pod minimální úroveň je zaslána *LowPower* notifikace.

Příkazy Get a Insert

Servis **RobotComm** podporuje na několika svých portech získávání údajů o robotu. Pomocí těchto příkazů je především možné získat stav jednotlivých senzorů. Příkazy jsou rozděleny do následujících skupin:

1. *Power* navrátí volající službě informace o stavu baterie robotu. Těmito informacemi jsou její aktuální napětí, kapacita (hodnota nemusí být správná při použití neoriginálních baterií), proud tekoucí z/do baterie, teplota baterie a stav nabíjení robotu.
2. *ObstackleAhead* navrací informace o stavu senzorů detekující překážky před robotem. Těmito senzory jsou pravý a levý dotekový senzor, detektor virtuální stěny, pravý, pravý přední, levý přední a levý detektor převisu a tři detektory prověsu kol.
3. *CliffDetail* navrací přesnou hodnotu infračervených detektorů převisu a detektoru stěny jako hodnotu typu *integer*.

Pozn.: Pro potřeby této práce byly testovány hodnoty z detektorů převisu jako nástroj detekce barvy podlahy. Ukázalo se, že detekce přesné barvy pomocí těchto snímačů není možná. Výstupní hodnoty závisí silně na osvětlení místnosti a na natočení robotu vůči zdroji tohoto světla. Senzory je možné použít pouze pro diferenční měření barvy povrchu. Vhodná aplikace by mohla být na robotu sledujícím čáru na zemi.

Simulace

Pro simulaci robotu byl použit simulátor distribuovaný společně s instalací MRDS. Jedná se o simulátor VSE (*Visual Simulation Environment*). Simulátor je vhodný pro potřeby simulace robotu v inteligentním domě, protože poskytuje realistické suchozemské prostředí. Pro renderování prostředí je v simulátoru interně využíváno *Physix*, *XNAFramework* a *DirectX*. Na Obr 7.7 je zobrazen robot v simulátoru. Při simulaci byl zvolen manifest *Apartment.Simulation.Manifest.xml*. V tomto manifestu je vytvořen byt vhodný pro simulaci inteligentního domu.

Obr 7.7: iRobot Create v simulovaném apartmá

Aplikace v této práci je navržena tak, aby pro přesunutí robotu do simulátoru bylo nutné pozměnit pouze servis **RobotComm**. Z důvodů popsaných výše není možné použít výstup simulované kamery v aplikaci robotu.

Služba **RobotComm** je pro použití simulátoru upravena tak, aby se na základě informací ze simulovaných senzorů doteku a enkodérů kol chovala téměř stejně jako při použití servisu **iRobotLite**. Servis **iRobotLite** při simulaci není využíván vůbec.

7.4 Shrnutí možností robotu

Robotická aplikace vyvinutá v této práci je vhodná pro potřeby autonomního robotu v inteligentním domě. Lokalizace robotu probíhá použitím metody *dead-reckoning*, pomocí detekce dokovací stanice robotu a pomocí rozpoznávání grafických vzorů rozmístěných v inteligentním domě. Možnost implementovat lokalizaci na bázi přijímání infračervených signálů zařízení rozmístěných po domě je v kódu nastíněna a připravena, nebyla však implementována, protože se pro potřeby této práce nepodařilo obstarat příslušný kompatibilní hardware.

V současnosti je robot schopen přicestovat na zadané souřadnice nebo pojmenované místo/místnost v domě. Navádění probíhá sofistikovanou metodou na dvou úrovních, přičemž na první je cesta navržena metodou *A**, na druhé je použit algoritmus navádění pomocí vektorových polí. Robot je schopen vyhnout se nově nalezeným překážkám a po dostatečně dlouhou dobu o nich neztratit povědomí. Nevýhodou zvoleného přístupu je, že jsou veškeré informace uloženy do dvourozměrné mapy, takže v domě s více podlažími (propojenými výtahem nebo svislou rampou) není toto navádění schopno fungovat správně.

Pro vizualiaci dění v domě je robot, vybavený kamerou, schopen zprostředkovávat živý přenos přes internet. Autentizovaný uživatel je tak schopen pomocí internetového prohlížeče kontrolovat dění v domě. Nevýhodou je v současnosti kompatibilita pouze s webovým prohlížečem Google Chrome.

Pro interakci mezi robotickou aplikací a domem nebo uživatelem terminálu inteligentního domu bylo vytvořeno příkazové konzolové rozhraní. Toto rozhraní může být samo využito inteligentním domem přesměrováním standardních vstupů a výstupů robotické aplikace nebo může být jednoduše nahrazeno jiným protokolem, pracujícím například na principu zasílání SOAP zpráv.

8 INSTALACE

Tato část práce obsahuje návod na instalaci a konfiguraci robotické aplikace probrané v předchozí kapitole. Instalaci je možné provést rozmístěním jednotlivých servisů na libovolný počet stanic, nicméně v této kapitole bude prezentována pouze možnost instalace aplikace do simulátoru, na samotné PC a na dvojici stanic Server – Robot. V kapitole bude také probrán hardware použitý pro běh aplikace. A nachází se zde návod ke konfiguraci jednotlivých servisů a ke kalibrace robotu.

8.1 Použitý hardware

V této podkapitole je probrán hardware a jeho různé kombinace, které byly použity pro běh této robotické aplikace. Jednotlivé zmiňované součásti jsou všechny zachyceny na Obr 8.1. Složený robot je zachycen na Obr 8.2. Následující podkapitoly jsou chronologicky seřazeny v pořadí, v jakém byl příslušný hardware při vývoji aplikace použit. U jednotlivých PC jsou zmíněny pouze podrobnosti o hardware a software důležitých pro tuto práci.

Obr 8.1: Rozložený robot

Obr 8.2: Kompletovaný robot

8.1.1 Master PC

Hlavní počítač byl použit pro simulaci serveru inteligentního domu. Na stanici je nainstalovaný operační systém *Microsoft Windows 7 32bit*, vývojové prostředí *Visual Studio 2010*, vývojové prostředí *Microsoft Robotics Developer Studio 4* (*.NET 4.0*) a internetový prohlížeč *Google Chrome*. Počítač disponuje procesorem *Intel Core i3 M350*, *4GB RAM*, *Wi-Fi*, *USB* sběrnicí a grafickou kartou *ATI Mobility Radeon HD 5800*. Byl přidán do pracovní skupiny s PC zmíněném v kapitole 8.1.3.

V první fázi vývoje aplikace byl počítač použit pro simulaci robotu v grafickém simulačním prostředí *Visual Simulation Environment*. Následně byl použit pouze v kombinaci s robotem samotným a nakonec zastupoval místo domácího serveru, kdy komunikoval s PC upevněným k tělu robotu pomocí Wi-Fi.

8.1.2 iRobot Create Model 4400

Použitý robot, popsaný v kapitole 4, byl *iRobot Create Model 4400*. V následujících podkapitolách je popsán veškerý hardware, který byl použit se zvoleným robotem. Navíc byly v ČR zakoupeny a testovány *Dobíjecí základna – Home Base Compact* a *Dálkové ovládání iRobot Roomba*. Tato zařízení se ale ukázala být nekompatibilní s použitým robotem.

iRobot Serial Cable + Delock Adapter USB 2.0 to seriál

Pro připojení robotu k PC byl použit originální sériový kabel značky iRobot. Vzhledem k tomu, že žadný z dostupných počítačů není vybaven sériovým portem, byla pro komunikaci použita redukce značky *Delock*.

iRobot Roomba Power Adapter + baterie APS

V robotu byla využita značková baterie *APS* k robotům firmy iRobot. Elektrický adaptér iRobot, zakoupený v ČR, byl použit k jejímu nabíjení.

Bluetooth Adapter Module

BAM model 10542B, připojený ke konektoru DB-25 na robotu Create, byl použit pro testování spojení robotu s PC *ASUS Eee*, který obsahuje modul Bluetooth. Komunikace mezi počítačem a robotem za pomocí Bluetooth proběhla úspěšně, nebyla však použita pro potřeby této práce z důvodů popsaných v kapitole 5.3.1.

Fourth Wheel

Z důvodu nadměrného zatížení nákladové části při nesení ovládacího počítače bylo použito přídavné kolečko pod zadní částí robotu. Toto přídavné kolečko je popsáno v závěru kapitoly 5.1.1.

8.1.3 ASUS Eee PC 1000H

V úloze stanice, ovládající hardware robotu, byl použit netbook *ASUS Eee PC 1000H*. Tento počítač má rozměry 200 mm x 270 mm a hmotnost 1,45 Kg. Na této stanici je nainstalovaný operační systém *Microsoft Windows XP*, *.NET 4.0* a robotické balíčky *CCR*, *DSS* a *XNA Framework* (odlehčená instalace MRDS). Tento počítač je vybaven

procesorem *Intel Atom N270*, *1GB RAM*, webovou kamerou *1,3M Pixel*, *Wi-Fi*, *Bluetooth* a *USB* sběrnicí.

Tento netbook disponuje optimálními rozměry a hmotností pro potřeby této aplikace. Za jejího běhu je otevřený počítač položen na těle robotu. Propojení mezi robotem a stanicí probíhá za pomocí sériového kabelu. Stanice má za běhu aplikace výdrž zhruba 4 hodiny. Vzhledem k tomu, že zařízení robotu i PC využívá napájení na obdobných napětích, bylo by možné použít pro běh stanice konvertovaný proud přímo z baterie robotu. Ten by sice bylo nutné častěji nabíjet, ale odpadla by tím potřeba připojovat obě zařízení do adaptérů. Při použití dokovací základny by dobíjení mohlo probíhat souběžně a automatizovaně.

8.2 Konfigurační soubory

Výhodou použití robotu v inteligetním domě je, že robot se pohybuje v pseudo-známém prostředí. Je tak možné předem navrhnout mapu, ve které jsou zahrnuty všechny trvalé překážky. Nastavení robotu je pro přehlednost rozděleno do několika konfiguračních souborů, které jsou při svém startu načteny odpovídajícími servisy. Konfigurační soubory jsou formátovány pomocí *xml*, takže jsou vysoce strukturované a jejich editace je přehledná.

8.2.1 Soubor Localization.State.xml

V souboru *Localization.State.xml* jsou uloženy informace o jednotlivých místnostech domu, výchozí nastavení intenzity cíle, výchozí nastavení intenzity dočasných překážek, výchozí pozice robotu při startu aplikace a hodnoty aproximací pohybu robotu (více k jejich významu naleznete v podkapitole 7.3.2; způsob jejich určení je popsán v kapitole 8.3). Tento soubor inicializuje servis **Localization**. Rozměrové jednotky, použité v konfiguračních souborech, musí být stejné, jako ty deklarované ve třídě *Dimensions* v servisu **RobotComm**.

Následuje vzorový příklad struktury konfiguračního souboru. Jednotlivé elementy jsou popsány níže v této podkapitole.

```
<?xml version="1.0" encoding="utf-8"?>
<LocalizationState>
  <DefaultPosition/>
  <DefaultGoalIntensity/>
  <DefaultObstackleIntensity/>
  <MovementErrorRatio/>
  <VisionAccuracy/>
  <RotationRightApproximation/>
  <RotationLeftApproximation/>
  <StraightAngleApproximation/>
  <StraightApproximation/>
```

```
  <Rooms> //Pokračování
    <Room/>
  </Rooms>
  </Cam>
  <Glyphs>
    <Glyph/>
  </Glyphs>
  <GlyphCorners/>
  <GlyphSizeInSquares/>
</LocalizationState>
```

Výchozí pozice a intenzity

V tagu *DefaultPosition* v konfiguračním souboru je uložena výchozí pozice robotu. Tyto souřadnice jsou použity v případě, kdy žádná z místností nespecifikuje umístění dokovací stanice. Správné vyplňení tagu je povinné. Pokud není načtena výchozí pozice, aplikace vygeneruje výjimku a nedojde k jejímu úplnému spuštění. Samotný tag *DefaultPosition* obsahuje tři povinné hodnoty typu *double*. Hodnoty *X* a *Y* (velikost písmen je v *xml* důležitá) určují pozici středu robotu při startu aplikace. V tagu *Orientation* je hodnota počátečního natočení robotu ve stupních relativní ke směru zvoleného souřadného systému. Pokud tato hodnota není v rozmezí 0-360, je upravena pomocí funkce *mod* tak, aby do tohoto rozmezí spadala.

```
<DefaultPosition>
  <X>123.4</X>
  <Y>567.8</Y>
  <Orientation>120</Orientation>
</DefaultPosition>
```

Pozn.: Oddělovač desetinných míst je znak '.' (tečka).

Tagy *DefaultGoalIntensity* a *DefaultObstackleIntensity* jsou hodnoty typu *double*. Vyjadřují intenzitu cíle, která bude primárně použita při navádění robotu pomocí vektorvých polí, a intenzitu dočasných překážek, které robot objeví. Nastavení těchto hodnot tak, aby pracovaly optimálně v souhře s ostatními hodnotami překážek, je popsáno níže v této kapitole. Příklad zápisu daných tagů:

```
<DefaultGoalIntensity>1</DefaultGoalIntensity>
<DefaultObstackleIntensity>42831.5</DefaultObstackleIntensity>
```

Aproximace pohybu

V odpovídajících sekcích konfiguračního souboru lze nastavit hodnoty aproximace pohybu. Jednotlivé tagy se jmenují *MovementErrorRatio*, *VisionAccuracy*, *RotationRightApproximation*, *RotationLeftApproximation*, *StraightAngleApproximation* a *StraightApproximation*. Tyto tagy nejsou povinné. Jejich naplnění je obdobné jako u tagu *DefaultGoalIntensity*. Jejich hodnoty je třeba změřit a vypočítat v závislosti na

použitém hardware. Tento postup je vysvětlen v kapitole 8.3. Způsob jejich použití za běhu aplikace je popsán v podkapitole 7.3.2 - Sebelokalizace.

Nastavení místností

V tagu *Rooms* se nachází seznam všech místností, jejich počet není omezen. Tento tag není povinný, ale pro aktivaci funkcí automatického navádění je nezbytné jej využít.

Každá z místností je deklarována v samostatném tagu *Room*. V tomto tagu se nachází informace o poloze, názvu, statických překážkách, významných bodech a infračerveném kódu dané místnosti. Špatným nastavením místností může dojít k nesprávné funkci aplikace. Z technologických důvodů není možné toto nastavení ověřit za běhu aplikace. Z důvodu chyb při výpočtu trajektorie je třeba dbát na to, aby žádná z místností nebyla nedopatřením zdvojena. Uživatel by se měl také vyvarovat deklarování místnosti uvnitř jiné místnosti nebo vzájemného překrytí třech místností (toto nezpůsobí chybu za běhu aplikace, ale zbytečně narostou nároky na výpočet trajektorie robotu).

```
<Room>
  <Name/>
  <RoomColor/>
  <HomeBase/>
  <IRID/>
  <Shape/>
  <Walls/>
  <Obstackles/>
  <SpecialPlaces/>
</Room>
```

V tagu *Name* je uložen řetězec názvu místnosti. Tento údaj není povinný a více místností může mít stejný název. V názvu místnosti jsou povoleny jakékoliv znaky sady *utf-8*, je ale doporučeno používat jednoslovný název bez diakritiky. V případě použití bílých znaků v názvu může dojít k jeho špatnému zobrazování v uživatelském rozhraní a pravděpodobně nebude možné zadat navádění do dané místnosti. Při použití znaků různých od znakové sady *ASCII*, nemusí být tyto znaky zobrazeny správně v internetovém prohlížeči.

Hodnota uložená v tagu *HomeBase* udává souřadnice, na kterých se robot nachází v zadokované poloze. V konfiguračním souboru se tato hodnota smí vyskytovat nanejvýš v jedné místnosti, v případě opakovaného výskytu je při inicializaci vypsáno varování a starší výskyt je přepsán novějším. Struktura tagu je stejná jako u tagu *DefaultPosition*.

V tagu *IRID* je uložen infračervený identifikátor místnosti. Jeho hodnota musí korespondovat s hardware umístěným v dané místnosti, který kód vysílá. Ten může být použit při lokalizaci robotu způsobem popsaným v kapitole 7.3.2. Typ tohoto

identifikátoru je *unsigned byte*, tudíž jsou povoleny pouze celočíselné hodnoty v rozmezí 0-255.

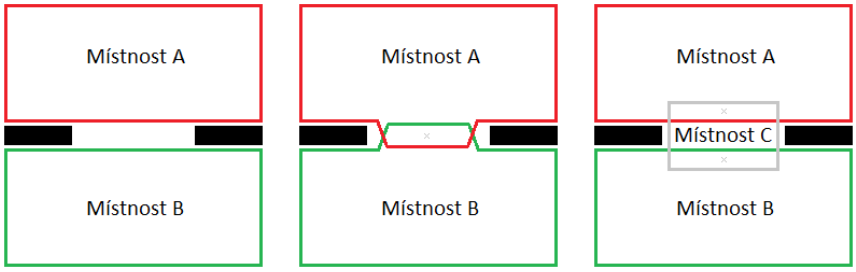
Tvar místnosti

V rámci mapy si lze každou místnost představit jako polygon o minimálně třech rozích. Pro optimální funkcionalitu lokalizace pomocí vektorových polí je vhodné volit konvexní tvar místnosti a to tak, aby okraje místnosti sledovaly statické překážky. Místnosti, které jsou rozděleny významnou statickou překážkou, je vhodné složit z několika oddělených polygonů, které se mohou jmenovat stejně (příkladem mohou být místnosti 1 a 2 na Obr 6.7, str. 56).

Tvar místnosti je specifikován v rámci tagu *Shape* a jednotlivé rohy místnosti jsou vyjádřeny hodnotami *x* a *y* typu *double* v rámci vnořeného elementu *IntPoint*. Příklad deklarace trojúhelníkové místnosti:

```
<Shape>
  <IntPoint>
    <x>1</x>
    <y>0</y>
  </IntPoint>
  <IntPoint>
    <x>2</x>
    <y>0</y>
  </IntPoint>
  <IntPoint>
    <x>1.5</x>
    <y>1</y>
  </IntPoint>
</Shape>
```

Přechody mezi místnostmi, které může robot využít pro cestování, jsou vyjádřeny překrytím polygonů dvou místností. Vzhledem k nenulové tloušťce reálných stěn je pro překrytí dvou místností v oblasti průchodu (dveří) možné učinit výstupek v jedné nebo obou sousedních místnostech tak, aby se polygony přilehlých místností překrývaly. To ovšem vede na konkávnost daných místností. Doporučenou metodou tvorby přechodů mezi místnostmi je vytvoření spojovací místnosti, která zasahuje do obou spojovaných místností alespoň na šířku robotu.



Obr 8.3: Příklad vytvoření přechodu mezi místnostmi

Demonstrace propojení místností je vyobrazena na Obr 8.3. V levé části jsou zobrazeny nepropojené místnosti A a B. Přestože je mezi těmito dvěma místnostmi volný průchod, robot se jej nikdy nepokusí využít, protože polygony obou místností nejsou překryty. V prostřední části je ukázka implementace propojení obou místností vytvořením spojovacího „zubu“. Šedý znak zobrazuje střed přechodu mezi místnostmi, na který se robot snaží přicestovat. Přestože se jedná o správné a fungující řešení, elegantnější přístup je zobrazen v pravé části Obr 8.3. Zde jsou místnosti A a B propojeny třetí místností C, což vede ke snaze robotu dostat se na střed dveří a průchod samotný projet nejbezpečnější cestou.

Statické překážky

V konfiguračním souboru servisu **Localization** lze deklarovat statické překážky typu *Wall* a *Obstackle*. Tyto překážky jsou považovány za neměnné po dobu celého běhu aplikace. Při jejich výjimečné změně je třeba přenastavit soubor *Localization.State.xml* a aplikaci restartovat.

Tag *Obstackles* obsahuje libovolný počet statických bodových překážek v dané místnosti. Deklarace jedné překážky je uzavřena v tagu *Obstackle*. Hodnota I*ntensity* udává sílu, s jakou se robot snaží dané překážce vyhnout. Hodnota *Radius* udává maximální dosah, ve kterém je překážka robotem vnímána. Všechny elementy vnořené do tagu *Obstackle* jsou typu *double*.

```
<Obstackles>
  <Obstackle>
    <X>6260</X> <Y>3240</Y>
    <Intensity>450000</Intensity>
    <Radius>1000</Radius>
  </Obstackle>
</Obstackles>
```

Tag *Walls* obsahuje seznam statických překážek typu stěna v dané místnosti. Každá instance stěny je uložena uvnitř tagu *Wall*. Stěna je úsečka, specifikovaná pomocí dvojice krajních bodů. Ostatní parametry tagu *Wall* jsou obdobné jako u tagu *Obstackle*.

```
<Walls>
  <Wall>
    <Begin><X>6760</X><Y>4090</Y></Begin>
    <End><X>6760</X><Y>6000</Y></End>
    <Intensity>450000</Intensity>
    <Radius>1000</Radius>
  </Wall>
</Walls>
```

S deklarací statických překážek úzce souvisí hodnoty tagů *DefaultGoalIntensity* a *DefaultObstackleIntensity*. Pro správnou funkci algoritmu vektorových polí (kapitola

6.5.3) je třeba nalézt pro každou statickou i dočasnou překážku vhodné hodnoty atributů *Intensity* a *Radius*.

K nalezení správné rovnováhy mezi atributy jednotlivých překážek je třeba brát v úvahu pozici dané překážky, existenci sousedních překážek v jejím okolí a rozměry robotu. Dále je třeba brát v úvahu, že dvě překážky (nejčastěji navazující rohy místností) sčítají své působení na robot.

Pro získání základní představy závislosti mezi intenzitou překážky a jejím vlivem na trajektorii robotu lze uvažovat případ, kdy se v prostoru nachází pouze jedna překážka typu bod, s nulovými fyzickými rozměry, a jeden cíl. Výslednou intenzitu elektrického pole působícího na robot lze ve vzdálenosti r od překážky vypočítat jako vektorový součet rovnic (13) a (14). Ve vzdálenosti robotu od překážky, rovné poloměru robotu, dochází ke kontaktu mezi robotem a překážkou. Pro zamezení nárazu robotu do překážky musí mezi nábojem překážky a nábojem cíle platit vztah

$$\vec{E}_{min} = \frac{q_o}{r_r^2} > q_g = \vec{E}_g \tag{16}$$

kde $\vec{E}_{min}$ *… je minimální nutná intenzita pole překážky,*
q_o *… je náboj překážky,*
r_r *… je poloměr robotu,*
q_g *… je náboj cíle,*
$\vec{E}_g$ *… je intenzita pole cíle.*

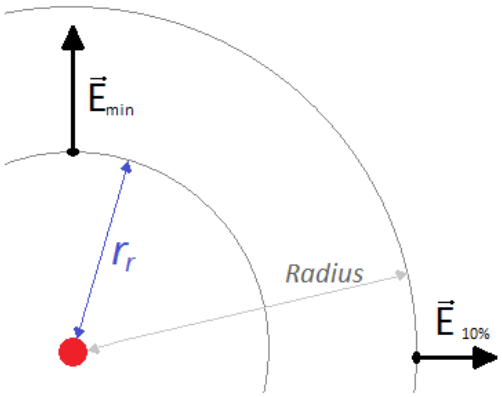
Pro ilustraci je tato překážka znázorněna na Obr 8.4. Hodnota r_r u použitého iRobot Create je 170 mm. Při navádění se robot standardně pohybuje polovinou maximální rychlosti, tj. zhruba 250 mm/s. Úplné zastavení robotu při této rychlosti trvá maximálně 400 ms. Zastavování robotu lze aproximovat rovnoměrně zpomaleným pohybem. Pro brzdnou dráhu robotu platí

$$s = v_0 t + \frac{1}{2}\frac{\Delta v}{\Delta t} t^2 = \frac{1}{2} v_0 t = 50 \ [mm] \tag{17}$$

kde s *… brzdná dráha robotu,*
v_0 *… rychlost při započetí zastavování 250 mm/s,*
t , Δt *… doba brždění 400 ms,*
Δv *… úbytek rychlosti -250 mm/s.*

Při přímém pohybu robotu k cíli je pro zamezení srážky nutné, aby intenzita odpudivého pole překážky $\vec{E}_{min}$ vyrovnala intenzitu přitažlivého pole cíle $\vec{E}_g$ ve

vzdálenosti r_{min} , která je dána součtem poloměru překážky (v tomto příkladě 0 mm), poloměru robotu a brzdné dráhy robotu. Hodnota r_{min} je v takovém případě 220 mm. Jako referenční hodnotu náboje cíle je možné použít jakoukoliv konstantu. Dosazením do rovnice (16), za r_r hodnotu r_{min} a za q_g hodnotu 1 C, lze vyjádřit minimální bezpečnou hodnotu náboje cíle $q_o > 48400$ C. Atribut *Intensity* překážky v tomto příkladu tedy musí být hodnota větší než 48400 za předpokladu, že je atribut *DefaultGoalIntensity* roven 1. Při použití v reálné aplikaci bude ale hodnota *Intensity* překážek menší z důvodu nutnosti kompenzovat situaci, kdy na robot působí více překážek současně. *DefaultObstackleIntensity* je vhodné nastavit v okolí hodnoty 42800. Při této hodnotě se robot většině dočasných překážek vyhne, je ale méně náchylný na uvíznutí v lokálním minimu, a při občasném „ťuknutí“ do překážky ověří její existenci.



Obr 8.4: Rozložení intenzit elektrického pole v okolí překážky

Pro optimalizaci a urychlení výpočtu trajektorie robotu je vhodné pro statické překážky specifikovat také hodnotu *Radius*. Program robotu se tak vyhne zbytečnému počítání s příliš vzdálenými překážkami. Z důvodu omezení opotřebení robotu je vhodné, aby se rychlost motorů neměnila rázově. Za maximální povolenou změnu rychlosti lze zvolit takovou, která je způsobena změnou intenzity pole, působícího na robot, o 10%. Intenzita pole působícího na robot nemá sama na rychlost robotu žádný vliv, její prudká změna má ale vliv na změnu směru pohybu robotu, což bezprostředně ovlivňuje rychlosti motorů robotu (změna rychlostí motorů závisí na vektoru pohybu robotu při vstupu do dosahu překážky). Při uvažování hodnoty q_g rovno 1 C, lze úpravou rovnice (16) získat pro *Radius* překážky vztah

$$Radius > \sqrt{\frac{10\,q_0}{q_g}} = \sqrt{484000}\ \ [mm] = 695\ \ [mm]\,. \tag{18}$$

Z rovnice (18) vyplývá, že hodnota *Radius* překážky by neměla být menší než 695 mm.

Definice atributů jednotlivých statických překážek v konfiguračním souboru *Localization.State.xml* je sofistikovaný proces a závisí především na potřebách a rozloze daného domu. Postupy a rovnice, zmíněné v této podkapitole, mohou být kombinovány při konfiguraci mapy inteligentního domu. Možnost algoritmizace procesu tvorby mapy by bylo zajímavé rozšíření této práce a mohlo by na ni navazovat. Takovýto program by jako vstupní parametry mohl přijímat plán domu a mohl by sám generovat rozložení jednotlivých místností a hodnoty statických překážek.

Významná místa

Pro zlepšení použitelnosti robotu lze deklarovat v rámci domu „významná“ místa. Tato místa jsou pojmenované souřadnice, na které bude robot často posílán. Uvnitř tagu *SpecialPlaces* se může nacházet libovolný počet elementů *PointOfInterest*. V rámci tohoto tagu je deklarován unikátní jednoslovný název *Name* „významného“ bodu (unikátní v rámci názvů místností a jiných „významných“ bodů). Dále jsou deklarovány souřadnice bodu. Vnitřní element *TitleHeight* není povinný a určuje velikost fontu popisu tohoto bodu v mapě webového prohlížeče. Zobrazení názvu „významného“ místa si lze představit tak, jako by vedle tohoto bodu ležel banner s jeho názvem. Výchozí výška písmen banneru je totožná s průměrem robotu. Jinou velikost fontu v milimetrech lze definovat pomocí celočíselného atributu *TitleHeight*.

```
<SpecialPlaces>
  <PointOfInterest>
    <Name>Desk</Name>
    <Position>
      <X>10090</X>
      <Y>4360</Y>
    </Position>
    <TitleHeight>180</TitleHeight>
  </PointOfInterest>
</SpecialPlaces>
```

Proměnné počítačového vidění

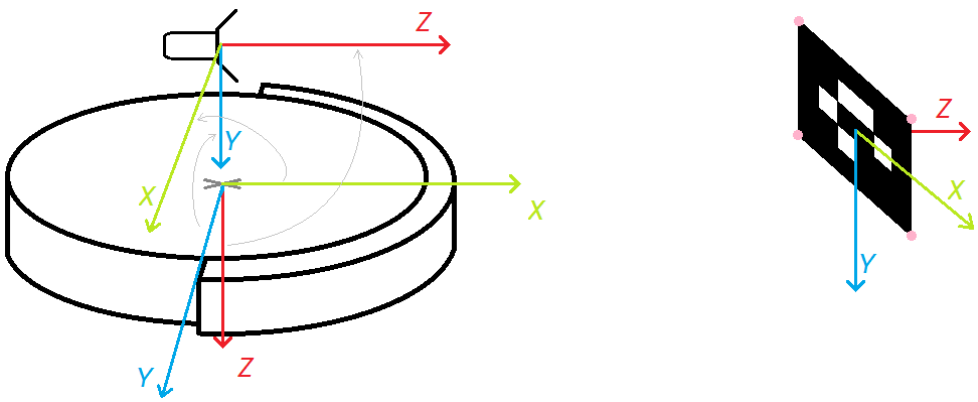
Součástí mapy inteligentního domu jsou pozice detekovatelných grafických vzorů. Jejich seznam je uložen v tagu *Glyphs*. Instance jednoho vzoru se nachází v tagu *Glyph*. Ten obsahuje souřadnice tagu v referenční soustavě domu a údaj o jeho rotaci v trojrozměrném prostředí. Tyto údaje se zjistí tak, že se nejprve provede translace středového bodu vzoru na požadovanou pozici (hodnoty tagu *Position*). Poté se provede jeho rotace kolem os v pořadí Z, X a Y. Toto pořadí je zvoleno v souladu se standardem frameworku *XNA*. Použitá prostorová soustava kartézských souřadnic je levotočivá, takže při pohledu na vzor zepředu směřuje osa Z za něj, osa X doprava a osa Y dolů. Názorná ukázka souřadné soustavy zavěšeného vzoru je na Obr 8.5 vpravo. V tagu *Code* je zaznamenáno rozložení černých a bílých čtverců vnitřní části vzoru, v člověku

čitelné podobě. Znak '1' odpovídá černé barvě a znak '0' bílé. Okraj vzoru je vždy vyplněn černě.

```
<Glyphs>
  <Glyph>
    <Position>
      <X>3360</X> <Y>7512</Y> <Z>-640</Z>
    </Position>
    <Rotation>
      <X>90</X> <Y>180</Y> <Z>0</Z>
    </Rotation>
    <Code>
      <Value>
        <string>1001</string>
        <string>0110</string>
        <string>1011</string>
      </Value>
    </Code>
  </Glyph>
</Glyphs>
```

Relativní pozice rohů vzoru vůči jeho referenčnímu bodu v milimetrech je uložena v tagu *GlyphCorners*. Tento tag musí obsahovat přesně 4 záznamy *Vector3*. Na Obr 8.5 jsou rohy naznačeny růžovými tečkami. Pokud by byl celý vzor široký 240 *mm*, vysoký 200 *mm* a jeho referenční bod byl v jeho geomtrickém středu, měl by první záznam (levý horní roh) tagu *GlyphCorners* hodnoty totožné s ukázkou. Ostatní rohy musí následovat ve směru hodinových ručiček.

```
<GlyphCorners>
  <Vector3>
    <X>-120</X> <Y>-100</Y> <Z>0</Z>
  </Vector3>
</GlyphCorners>
```



Obr 8.5: Ukázka souřadné soustavy robotu, kamery a grafického vzoru

Tag *GlyphSizeInSquares* obsahuje šířku a výšku celého vzoru ve čtverečcích.

```
<GlyphSizeInSquares>
  <X>6</X> <Y>5</Y>
</GlyphSizeInSquares>
```

Parametry použité kamery jsou uloženy v tagu *Cam*. Hodnoty *Tranlation* a *Rotation* značí vztah mezi referenční soustavou robotu a kamery, jak je patrné z Obr 8.5. Hodnota v tagu *FocalLength* je ohnisková vzdálenost kamery. Ta jedůležitá pro absolutní lokalizaci metodou Coplanar POSIT. Tato hodnota bývá blízká šířce obrazu kamery v pixelech. Pro dosažení lepších výsledků lokalizace je ale vhodné ji určit přesně [48].

```
<Cam>
  <Translation>
    <X>-170</X> <Y>0</Y> <Z>33</Z>
  </Translation>
  <Rotation>
    <X>0</X> <Y>-90</Y> <Z>90</Z>
  </Rotation>
  <FocalLength>630</FocalLength>
</Cam>
```

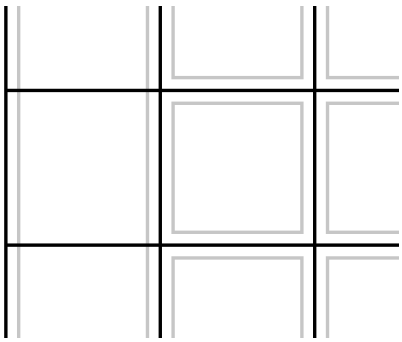
8.2.2 Soubor ImageRecognition.State.xml

V konfiguračním souboru služby **ImageRecognition** lze zadat výchozí hodnoty tagů *Cam* a *Glyphs,* popsaných v kapitole 8.2.1 - Proměnné počítačového vidění.

```
<ImageRecognitionState>
  <Cam />
  <Glyphs />
  <EdgeDetectorTreshold>40</EdgeDetectorTreshold>
  <MinBlobSize>30</MinBlobSize>
  <SquareSize>10</SquareSize>
  <SquareBorder>2</SquareBorder>
</ImageRecognitionState>
```

Tag *EdgeDetectorTreshold* je konstantní hodnota prahu, který je aplikován ve 3. kroce detekce vzoru v kapitole 6.3.2. Tag *MinBlobSize* udává minimální horizontální nebo vertikální rozměr uzavřených křivek, detekovaných v kroku 4. Tag *SquareSize* je násobek rozměru detekovaného vzoru, zmíněný v kroku 6 kapitoly 6.3.2.

V 8. a 9. kroce detekce grafického vzoru v kapitole 6.3.2 jsou počítány průměrné barvy jednotlivých čtverečků a rámce vzoru. Pro redukci chyby tohoto výpočtu, způsobené rozmazáním obrazu, je plocha v pixelech, použitá pro výpočet průměrné barvy, oříznuta o hodnotu *SquareBorder*. Výpočet nepřesnosti detekce vzoru (kapitola 7.3.5) ale tuto hodnotu nebere v potaz. Plocha oříznuté části čtverečků je znázorněna šedou barvou na Obr 8.6.



Obr 8.6: Ořez čtverců vzoru

8.2.3 Soubor iRobotComm.State.xml

Konfigurační soubor *IrobotComm.State.xml* slouží ke konfiguraci servisu **RobotComm**. Tento soubor je použit pouze při startu verze, která komunikuje se servisem **iRobotLite**. Všechny níže popsané atributy konfiguračního souboru jsou povinné.

Komunikace s robotem musí probíhat za pomoci sériového portu. Jeho číslo je nastaveno v tagu *SerialPort*. Při inicializaci spojení musí být přenosová rychlost nastavena na 57600 bps nebo 19200 bps. Po startu aplikace lze přenosovou rychlost zvýšit až na 115200 bps.

Pomocí tagu *IRobotModel* lze nastavit druh robotu z rodiny robotických vysavačů firmy iRobot. Povolené hodnoty jsou „Roomba“ a „Create“. Zvolením první možnosti lze aplikaci spustit i na skutečném vysavači firmy iRobot. Nebude však využita funkce streamování dat senzorů, kterou umí využívat pouze robot Create (maximální frekvence *pollingu* senzorů je nižší) a funkce tlačítka *play*, protože se hardware ovládací lišty robotů Create a Roomba liší. Aplikace nebyla s roboty Roomba testována.

Tag *ConnectionType* nastavuje druh připojení k robotu. V případě použití některého bezdrátového spojení je nutné k robotu připojit příslušný specializovaný hardware a na ovládacím PC zprovoznit připojení za pomoci Bluetooth na vybraném virtuálním portu. Akceptované hodnoty jsou:

1. „RooTooth“ – modul Bluetooth připojený k sériovému portu robotu.
2. „BluetoothAdapterModule“ (BAM) – bezdrátový adaptér připojený ke konektoru DB-25 v oblasti *Cargo Bay* robotu Create.
3. „RoombaSerialPort“ – připojení k robotu Roomba za pomocí kabelu.
4. „CreateSerialPort“ – připojení k robotu Create pomocí sériového kabelu.

V elementu *PollingInterval* lze nastavit periodu stahování dat z robotu. Jedná se o kladnou, celočíselnou hodnotu v milisekundách. U robotu Create je minimální doporučená hodnota 100 milisekund při použití zvýšené rychlosti připojení (*BaudRate*).

```
<IRobotCommState>
  <SerialPort>0</SerialPort>
  <BaudRate>57600</BaudRate>
  <IRobotModel>Create</IRobotModel>
  <ConnectionType>CreateSerialPort</ConnectionType>
  <PollingInterval>200</PollingInterval>
</IRobotCommState>
```

8.2.4 Soubor webcam.user.config.xml

Konfigurační soubor servisu **WebCam** je při jeho absenci generován automaticky během startu aplikace. Dochází k detekci připojených kamer a jejich seznam je do tohoto souboru zapsán. K použití s robotickou aplikací je vybrána první nalezená kamera. Pokud je ke stanici, na které běží servis **WebCam**, připojeno více kamer a není automaticky vybrána uživatelem preferovaná, může být tento konfigurační soubor manuálně pozměněn. Pro použití jiné kamery stačí přepsat tag *ActiveCamera*. Po restartu aplikace se požadovaná kamera automaticky použije.

8.2.5 Soubor WebService.xslt

Soubor *WebService.xslt* je transformační soubor, sloužící k převedení příslušných dat o robotu do HTML formátu. Tento soubor je načten internetovým prohlížečem při použití webového ovládacího rozhraní. Musí být přiložen ve složce „store“ v kořenovém adresáři běžícího servisu **Main**. V tomto souboru je hojně využit Javascript, který vytváří dynamické uživatelské rozhraní, a AJAX pro asynchroní stahování příslušných měnících se dat.

8.3 Kalibrace

Pro omezení chyb při lokalizaci robotu je nutné provést kalibraci daného hardware. Kalibrovat je nutné každý robot zvlášť, protože velikost systematických a nesystematických chyb navádění závisí především na drobných chybách hardware, daných nepřesností výroby. Kalibrace musí být provedena až po připevnění ovládacího PC. Při kalibraci je nutné nejprve provést eliminaci systematických chyb, poté je možné určit koeficient nesystematické chyby robotu při pohybu. Dostatečně uspokojivých přesností bylo dosaženo provedením deseti kalibračních cyklů.

8.3.1 Aproximace systematických chyb

Systematické chyby hardware jsou rozčleněny do několika kategorií. V rámci těchto kategorií je třeba experimentálně určit velikost chyby, kterou způsobují. Při výpočtu hodnot aproximací je použita modifikace metody *UMBmark* [32]. Použité metody

z tohoto postupu koncepčně vychází, liší se však jeho přesnou implementací. Kalibraci je třeba provádět na povrchu, na kterém se bude robot pohybovat nejčastěji, a za teploty vzduch, při které bude operovat (roztažnost kol).

Chyba rotace

Vlivem nepřesností enkodéru kol a vlivem poloměru kol různého od toho, se kterým pracuje vnitřní funkce *Angle*, je výstupní hodnota této funkce nepřesná. To vede na přetáčivost nebo nedotáčivost rotace robotu. V praxi je rotace robotu Create nepřesná až o 5%. Chyba této rotace se mírně liší v závislosti na jejím směru.

Chybu rotace robotu lze vyjádřit jako rozdíl naměřené a požadované hodnoty rotace ku požadované hodnotě rotace. Požadovaná hodnota je ta, kterou vrací funkce *Angle* a skutečná hodnota je ta, o kterou se robot fyzicky otočí. Po úpravě platí vztah

$$\epsilon_{rotace} = \frac{\alpha_{skutečná}}{\alpha_{požadovaná}} - 1[-] \tag{19}$$

kde ϵ_{rotace} … *chyba rotace v rozsahu <-1, 1>,*

$\alpha_{skutečná}$ … *úhel natočení robotu,*

$\alpha_{požadovaná}$ … *požadovaný (a naměřený) úhel otočení robotu*.

Chybu rotace lze experimentálně zjistit například tak, že se robotu přikáže otočit o úhel 360° ($\alpha_{požadovaná}$) a po jeho zastavení se odečte úhel jeho skutečného pohybu ($\alpha_{skutečná}$). Pro chybu rotace platí, že je rovna nule pokud je výstupní hodnota funkce *Angle* správná. Kladná chyba rotace značí, že je robot přetáčivý.

Chybu rotace je nutné změřit pro pohyb doprava i doleva. Výsledné hodnoty by měly být téměř stejné (pokud není změněno těžiště robotu, například přidaným PC). Hodnoty těchto chyb se předají aplikaci formou tagů *RotationRightApproximation* a *RotationLeftApproximation* v konfiguračním souboru *Localization.State.xml.*

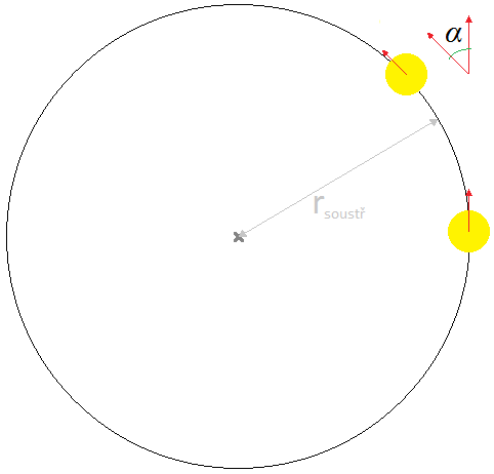
Chyba vzdálenosti

Z důvodů probraných v předchozí podkapitole dochází k nepřesnosti návratové hodnoty interní funkce *Distance*. Tato funkce vrací hodnotu vzdálenosti, kterou robot urazí, v milimetrech. Kladné hodnoty značí pohyb robotu směrem dopředu a záporné hodnoty směrem dozadu.

Tuto chybu lze vyjádřit jako rozdíl skutečné a požadované uražené vzdálenosti robotu za předpokladu, že je požadovaná vzdálenost tisíc milimetrů. Pokud má tedy robot za úkol urazit 2000 mm a ve skutečnosti ujede 2400 mm, je hodnota této chyby 200. Pokud robot urazí pouze 1800 milimetrů, je hodnota této chyby -100. Servisu **Localization** je tato hodnota předána formou elementu *StraightApproximation* v souboru *Localization.State.xml*.

Chyba soustřednosti přímého pohybu

Vlivem nerovnoměrností enkodérů kol, položení obou kol mimo osu nebo jejich různým poloměrem vzniká chyba soustřednosti přímého pohybu robotu. Kalibraci této chyby je třeba provést až po kalibraci chyby rotace a vzdálenosti.



Obr 8.7: Ukázka chyby soustřednosti při přímém pohybu

Při přímém pohybu robotu je patrné, že robot opisuje kružnici s poloměrem $r_{soustř}$. Kontrolní obvody přitom indikují, že robot jede po přímce. Pokud by se robot pohyboval po této trase, navrátí se po určité době do výchozího bodu. Ukázka chyby je zobrazena na Obr 8.7.

Vzhledem k faktu, že je poloměr soustřednosti většinou relativně velký, není v praxi proveditelná kalibrace měřením skutečného poloměru opsaného kruhu. K jeho zjištění je však možné použít rovnici pro výpočet kruhového oblouku. Robot ujede po přímce vzdálenost d = 10m (případně menší vzdálenost, pokud není možné nalézt vhodný prostor pro provedení kalibrace) a odečte se změna jeho natočení ve stupních. Kladné hodnoty stupňů přitom odpovídají otočení směrem doleva kolem osy robotu. Na Obr 8.7 je tento úhel naznačen symbolem α. Pro poloměr kružnice soustřednosti potom platí vztah

$$r_{soustř} = \frac{180\,d}{\alpha\,\pi}[m] \qquad (20)$$

kde $r_{soustř}$ … *poloměr kruhu soustřednosti robotu v metrech,*
d … *délka trajektorie v metrech,*
α … *natočení robotu oproti výchozí hodnotě ve stupních.*

Je patrné, že robot s nulovou chybou soustřednosti má tento poloměr roven nekonečnu. Vypočtená hodnota poloměru soustřednosti (kladná pokud robot zatáčí doleva a záporná pokud robot zatáčí doprava) je předána robotické aplikaci pomocí aproximační proměnné *StraightAngleApproximation* způsobem popsaným v kapitole 8.2.1. Rozměr (fyzikální jednotku) této proměnné je nutno převést z metrů na milimetry. Pro zpřesnění výpočtu proměnné je vhodné kalibraci provést opakovaně a výslednou hodnotu zprůměrovat.

8.3.2 Chyba polohy robotu vlivem nesystematických chyb

K určení nesystematické chyby pohybu robotu ϵ_e lze použít metodu *UMBmark* [32] a rovnici (4). Robot je ponechán projet čtverec o délce strany 4 m nebo kruh o poloměru 2 m. Z rozdílu jeho výsledného a očekávaného natočení je vypočtena hodnota chyby pohybu ϵ_e , popsaná v kapitole 6.1.1. Hodnota této chyby je předána robotické aplikaci formou elementu *MovementErrorRatio* v konfiguračním souboru *Localization.State.xml*.

8.3.3 Chyba počítačového vidění

Vzhledem k tomu, že je zjišťování polohy pomocí počítačového vidění absolutní měření, lze předpokládat normální rozložení vstupních hodnot. Proměnná *VissionAccuracy* potom bude odpovídat hodnotě σ^2 této závislosti. Tu lze získat opakovanou detekcí vzoru a následnou tvorbou normálního rozložení vrácené hodnoty. Fakt, že chyba tohoto měření závisí také na vzdálenosti kamery od rozpoznávaného kódu, a jejich vzájemném úhlu, lze pro potřebu této práce zanedbat.

8.4 Instalace na dvě stanice

V elektronické příloze k této práci jsou ve složce „Instalace na dvě stanice“ spustitelné soubory *Robot.exe* a *Server.exe*. Na obě stanice je nutné nainstalovat základní robotické prostředí za pomoci instalátoru MRDS. Obě stanice je nutné přidat do stejné pracovní skupiny (workgroup) systému Windows a do jedné podsítě. Správná konfigurace obou stanic pro podporu více DSSNode je vysvětlena v [46].

Pro instalaci na dvě stanice bylo nutné vybrat vyvážené rozložení servisů mezi stanicemi. Bylo nutné brát ohled na výpočetní zátěž ovládacího PC robotu a na velikost datového přenosu mezi stanicemi. Zvolená kombinace je taková, že servis **Main** (internetový server aplikace) běží na stanici *Server* a zbývající služby na stanici *Robot*.

Přiložené samorozbalovací balíčky je možné nainstalovat kamkoliv na příslušnou stanici, kam má daný uživatel právo zapisovat. Po instalaci je nutné pozměnit vytvořené

soubory s příponou *manifest*. Namísto všech výskytů výrazu „127.0.0.1“ je nutné zadat adresu IP druhé stanice. Aplikaci lze spustit pomocí souboru s příponou *cmd* ve složce s instalací. Na URL *http://localhost:50000/mainservice* stanice server se nachází internetové uživatelské rozhraní robotu.

Konfigurační soubory, použité v kapitole 9, jsou přiloženy ve složce „Instalace na dvě stanice\store“. Tyto soubory je nutné podle potřeby upravit a přesunout do místa instalace příslušného servisu. Nachází se zde také soubor *WebcamPipeServer.exe*. Ten musí být ve složce „bin“ příslušné instalace, pokud nejsou na dané stanici instalovány kompatibilní ovladače webkamery.

8.5 Instalace na samostatnou stanici

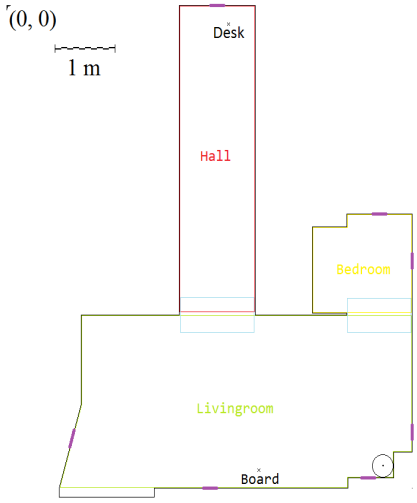
V elektronické příloze k této práci je ve složce „Instalace na jednu stanici“ samorobalovací program *Robot.exe*. Pomocí tohoto programu lze aplikaci instalovat na samostatný počítač, připojený k robotu. Aplikaci lze spustit pomocí vytvořeného souboru *Robot.cmd*. Lze použít konfigurační soubory z kapitoly 8.4.

8.6 Použití simulátoru

V elektronické příloze k této práci se ve složce „*Instalace do simulátoru*“ nachází archiv zip, který obsahuje soubory *dll* pro běh aplikace v simulátoru. Pro správný běh simulátoru je nutné mít nainstalované MRDS na příslušném počítači. Tento balíček je nutné extrahovat do kořenové složky MRDS, čímž se nainstalují všechny potřebné dokumenty. Aplikaci lze spustit souborem *Simulation.cmd* v kořenovém adresáři instalace MRDS. Před použitím simulátoru je nutné nastavit konfigurační soubor *Localization.State.xml* a zkopírovat příslušné **.xml* soubory.

9 UKÁZKA ČINNOSTI

V této kapitole je proveden a zdokumentován jednoduchý experiment s reálným robotem. V tomto experimentu bude nejprve předvedena schopnost navádění robotu na dané místo, a poté bude otestována funkčnost absolutní lokalizace. Experiment byl zachycen na video pomocí externí kamery (soubor *cam-1.mod* v elektronické příloze) a pomocí programu snímajícího obrazovku řídicí stanice (soubory *gui-*.avi* v elektronické příloze). Přehled a popis uživatelského rozhraní se nachází na Obr 7.2 na straně 64. Konfigurační soubory tohoto experimentu se nachází ve složce „Instalace na dvě stanice“ v elektronické příloze k této práci.



Obr 9.1: Mapa domu

Základní prerekvizita robotu, pohybujícího se v inteligentním domě, je apriorní znalost rozvržení místností (kapitola 8.2.1). Mapa místností se nachází na Obr 9.1. V levém horním rohu tohoto obrázku se nachází počátek souřadnic mapy a měřítko mapy. Jednotlivé pojmenované místnosti (barevný text) a „významné“ body (černý text) jsou v mapě popsány. Na Obr 9.1 jsou azurovou barou vyznačeny pomocné místnosti. Ty tvoří přechody. Kruhem je na tomto obrázku znázorněna výchozí pozice robotu.

Příprava robotu

V tomto experimentu byla použita instalace na samostatnou stanici (kapitola 8.5) a hardware popsaný v kapitole 8.1. Na robot byl, za pomoci lepicí pásky, pevně připojen řídicí notebook. Poté byl nastaven tag *Cam* v souboru *localization.state.xml*, přičemž je uvažováno, že monitor notebooku bude svírat pravý úhel se zemí. Ukázka obdobně sestaveného robotu je na Obr 8.2.

S notebookem v provozní poloze byla provedena kalibrace podvozku robotu (kapitola 8.3.1). Použitím funkcí *direct-drive* grafického uživatelského rozhraní byl

robot nejprve rotován o deset otáček doleva. Z uživatelského rozhraní byla odečítána hodnota, o kterou se robot údajně otočil. Touto hodnotou bylo poděleno číslo 3600 (deset otáček) a od výsledku byla odečtena 1 (vzorec(19)). Toto číslo tvoří kalibrační konstantu *RotationLeftApproximation* (kapitola 8.2.1 - Aproximace pohybu). Stejnou metodou byla zjištěna proměnná *RotationRightApproximation*. Data byla uložena do konfiguračního souboru a robot restartován.

Poté byl robot pomocí uživatelského rozhraní donucen dvakrát ucestovat pět metrů rovně. Rozdíl mezi výchozí a koncovou souřadnicí: *d* (je možné že bude nutné použít pythagorovu větu) byl odečten od hodnoty 10000 (deset metrů). Výsledek vytvořil hodnotu kalibrační konstanty *StraightApproximation*. Úhel, o který se robot mezi počáteční a koncovou souřadnicí otočil, lze označit jako α. Pomocí rovnice (20) byla z hodnot $d/1000$ a α odvozena chyba soustřednosti robotu (*StraightAngleApproximation*). Data byla opět uložena do konfiguračního souboru a robot restartován.

Předchozí tři kalibrační metody byly prováděny opakovaně a nový výsledek vždy kumulován se starým. Pro dobrou kalibraci podvozku je vyžadováno minimálně deset takových cyklů.

Nesystematická chyba robotu byla určena metodou *UMBmark*. Robot se nechal čtyřikrát jet rovně a zatočit o pravý úhel. Z rozdílu počáteční a výsledné hodnoty natočení byla za pomoci rovnice (4) určena konstanta *MovementErrorRatio.*

Cesta z výchozí pozice do místnosti Bedroom

První experiment je zkouška funkce *PathFind*. Robotu je přikázáno navést se z výchozí pozice (z hypotetické dokovací stanice) do místnosti s názvem Bedroom. Ve skutečnosti se ale robot v dokovací stanici nenachází. Jak je patrné z přiloženého videa, robot si svoji skutečnou polohu brzy uvědomí díky rozpoznanému vzoru na blízké zdi. Na mapě uživatelského rozhraní je vidět aktualizace odhadované polohy. Robot dále pokračuje po navržené trajektorii. A na rozhraní mezi současnou a cílovou místností zastaví.

Pozn.: na kamerovém záznamu je možné si všimnout některých zakrytých vzorů na stěnách. Experimentálně bylo prokázáno, že se drobnou redukcí jejich počtu schopnost lokalizace příliš nezmění.

Test absolutní lokalizace

Z videozáznamů *gui-2.avi* je patrná reakce robotu na situaci, kdy s ním manipulují lidé. Robot v počátečním stavu nezná svoji polohu. V průběhu ruční manipulace s ním si ji ale přesně odvozuje, v závisloti na detekovaných obrazcích. Pomalá reakce na změnu zobrazované polohy v *gui* je dána nízkou frekvencí *pollingu* těchto dat ze serveru.

10 NÁVÁZÁNÍ NA PRÁCI

V této kapitole jsou zmíněny způsoby, kterými lze na tuto práci navázat.

Ovládání hlasem

Zajímavým rozšířením práce by mohla být implementace ovládání robotu hlasem. MRDS obsahuje služby pro detekci hlasu, které by bylo možno použít a případně rozšířit o český jazyk. Při použití diferenčního mikrofonu by také mohlo ovládání hlasem být použito ke zjištění směru uživatele. Robot by tak byl schopen reagovat na příkaz „pojď sem“ nebo by bylo možné realizovat hru *Marco-Polo*. Autonomní robot by tuto funkcionalitu mohl použít při vyhodnocování krizových situací.

Rozpoznání postav

Pokud by byl vytvořen servis, používající vstup z kamery k vyhodnocování charakteristických znaků lidí, mohl by robot mít povědomí o obyvatelech nebo nezvaných hostech v domě. Mohl by tak být schopen se uživatelům vyhýbat a fungovat jako bezpečností systém.

Program pro automatické generování mapy

Před použitím robotu je nutné ručně vytvořit mapu domu a nastavit parametry statických překážek pro metodu vektorových polí (kapitola 8.2.1). Z toho důvodu by mohl být vytvořen program, který by generoval mapu domu automaticky, například za pomoci grafického uživatelského rozhraní nebo z plánů domu v papírové podobě. Mohl by být také vytvořen program, který je autonomně schopen umisťovat do mapy statické překážky a nastavit jejich hodnoty.

GlyphMaker v 2.0

Na tuto práci by mohlo být navázáno optimalizací a rozšířením programu na tvorbu grafických vzorů pro lokalizaci. Mohlo by být například provedeno rozšíření o detekci plně variantních vzorů. Nebo vytvořit mechanismus implementace *checksumu* a *parity* do obrázkových kódů.

Létající robot

Mechanismy absolutní lokalizace a navádění, vytvořené v této práci, by mohly být rozšířeny a použity při tvorbě autonomního létajícího robotu v inteligentním domě.

11 ZÁVĚR

Zadáním této diplomové práce bylo vytvořit autonomní robot, vybavený pro pohyb v prostorách inteligentního domu. Výstupem práce je software robotu, aplikovatelný na roboty z rodiny iRobot Create, případně (po změně software servisu **RobotComm**) na jiný diferenciálně řízený robot. K ovládání robotu je nutné jej propojit s PC s operačním systémem Windows. Robot může být vybaven kamerou, která přenáší obraz pomocí protokolu *HTTP*. Bylo vytvořeno webové uživatelské rozhraní, pomocí kterého může autorizovaný uživatel robot ovládat. Pokud je k robotu připojena kamera, je pomocí tohoto rozhraní možné zprostředkovat uživateli obraz. To je vhodné pro vizualizaci nestandardních a krizových situací v inteligentním domě.

Prvním bodem této práce bylo provedení rešerše funkcí a senzorických systémů inteligentního domu, spolupracujícího s autonomním robotem. V podkapitolách 2.1 až 2.4 byla probrána funkčnost, mechanismy a úkoly inteligentního domu. V podkapitole 2.5 pak byla nastíněna reprezentativní škála senzorů inteligentního domu, využitelných pro práci s autonomními roboty.

Druhému bodu zadání byla věnována 3. kapitola. Byly probrány různé druhy robotů, jejich funkce a požadavky kladené na jejich výběr. V závěru kapitoly byly nastíněny možnosti využití dat, získaných pomocí senzorů inteligentního domu, autonomními roboty a způsoby využití robotů v jejich prostředí.

V kapitole 4 této práce byl učiněn výběr vhodného autonomního robotu a zdůvodněno jeho využití v inteligentním domě. Vybraným robotem je iRobot Create. Ten koncepčně vychází z robotických vysavačů stejné firmy, je ale upraven pro snažší programovatelnost a není vybaven vysávacím ústrojím. Důvodem volby robotického vysavače jako autonomního robotu je jeho relativně nízká cena a malé omezení při pohybu v domě.

Při řešení 4. bodu zadání byly probrány jednotlivé možnosti použití robotu v inteligentním domě, včetně způsobů vyhodnocení různých situací, které mohou v domě nastat za nepřítomnosti obyvatel (kapitoly 3.2, 6). Dále byly probrány různé typy senzorů, které je možné použít (kapitola 5), a které by byly implementovány buď na samotný robot nebo v interiéru domu.

V kapitole 6 byla probrána široká škála možností lokalizace a navrženy způsoby aproximace chyb při pohybu robotu (kapitola 8.3). Byla navržena a implementována víceúrovňová metoda navádění robotu v dynamicky se měnícím prostředí. A byla implementována lokalizace robotu pomocí metody *Dead-reckoning* a rozpoznání grafických značek v prostorách domu.

Pro implementaci ovládacího software robotu bylo zvoleno vývojové prostředí MRDS (*Microsoft Robotics Developer Studio 4*) a v něm vytvořena robotická aplikace,

naprogramovaná v jazyce C# (kapitola 7; zdrojové kódy se nachází v příloze k této práci). Tuto aplikaci je možné spustit na jedné nebo dvou stanicích s operačním systémem Windows v inteligentním domě, přičemž jedna z těchto stanic komunikuje přímo s hardware robotu Create. Komunikace mezi stanicí a robotem může probíhat pomocí sériového kabelu nebo modulu Bluetooth. Komunikace mezi oběma stanicemi robotické aplikace nebo mezi stanicí robotické aplikace a inteligentním domem probíhá protokolem TCP/IP. Doporučeným způsobem spojení je šifrovaná Wi-Fi. Komunikace mezi jednotlivými stanicemi robotu je vedena pomocí odlehčeného protokolu SOAP.

Pro řešení práce byl zakoupen a zprovozněn robot Create. Vytvořená aplikace byla testována v simulačním prostředí VSE (kromě části týkající se počítačového vidění, z důvodu popsaného v podkapitole 7.3.4) a byla na robot aplikována (kapitola 9). Propojení mezi robotem a počítačem bylo realizováno pomocí sériového kabelu i modulu Bluetooth. Použití, konfigurace a instalace aplikace je probrána v kapitole 8. Byly vytvořeny samorozbalovací programy pro instalaci aplikace na skutečný robot. Tyto balíčky slouží k instalaci na „standalone“ PC nebo na dvojici stanic – robot a server. Pro instalaci aplikace s použitím simulátoru byl vytvořen třetí balíček, který vyžaduje funkční prostředí MRDS na dané stanici. V současné době robot může být ovládán přímo nebo mu může být přikázána cesta na dané souřadnice, „významný“ bod nebo specifikovanou místnost pomocí internetového rozhraní. Přitom si udržuje dynamickou mapu domu a je schopen registrovat a vyhnout se překážkám.

Pro realizaci rozhraní mezi robotem a inteligentním domem byl navržen příkazový protokol, kerý byl implementován na konzoli stanice reprezentující server inteligentního domu a pomocí kterého lze robotu zasílat píkazy. V kapitole 7.3.1 bylo popsáno jeho ovládání a navržena možnost jeho reimplementace pomocí SOAP protokolu.

Výstupem práce je tedy řídicí software, aplikovatelný na diferenciálně řízený robot. Všechny základní mechanismy pro funkci robotu v inteligentním domě jsou implementovány. Nicméně vzhledem k rozsáhlosti této problematiky a široké škále možností není výstup této práce v současné formě vhodný pro nasazení v komerčním sektoru. K tomu by bylo podle mého odhadu zapotřebí minimálně další 2-3 roky práce jednoho člověka. Bylo by také nutné použití jiného hardware (robot s vysavačem), pro který by bylo třeba aplikaci drobně upravit. V současnosti je jako ovládací stanice použit notebook, který je umístěn na robotu, a který je za běhu aplikace otevřený. Tím může dojít k jeho poškození překážkami s nízkým podhledem. Pro aplikaci robotu by tedy bylo vhodné zvolit jiný druh ovládací stanice. Nevýhodou při uvedení do komerčního prostředí by byla nutnost pořídit licenci MRDS. Výhodou stávajícího řešení je možnost instalace programu na skutečný robotický vysavač Roomba.

Vytvořená koncepce řešení problematiky autonomního robotu a zvolené vývojové prostředí se jeví být správným řešením a je na ně možné navázat. Výhody a možnosti vytvořené aplikace jsou shrnuty v kapitole 7.4.

Literatura

[1] Microcontroller. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012 [cit. 2012-04-03]. Dostupné z: http://en.wikipedia.org/wiki/Microcontroller

[2] SUTHERLAND, Jay. Electronic Computer for Home Operation (ECHO):The First Home Computer. *IEEE Annals of the History of Computing*. 1994, roč. 16, č. 3, s. 59-61. Dostupné z: www.computer.org/cms/Computer.org/ComputingNow/computingthen/atty/1994/ATTY-1994-2-Echo.pdf

[3] Home automation. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012 [cit. 2012-04-03]. Dostupné z: http://en.wikipedia.org/wiki/Digital_home

[4] STREISSGUTH, Tom. *HVAC System Explained. EHow* [online]. [cit. 2012-04-03]. Dostupné z: http://www.ehow.com/way_5809808_hvac-system-explained.html

[5] The Ultimate Teen Bedroom. *AB Audio Visual*: *Interactive Sound, Light and Vision* [online]. 2012 [cit. 2012-04-03]. Dostupné z: http://www.abaudiovisual.co.uk/teen%20bedroom.html + obrazky

[6] CHENG, Jim a Thomas KUNZ. A *Survey on Smart Home Networking*. Carleton University, 2009. Dostupné z: http://kunz-pc.sce.carleton.ca/thesis/SmartHomeNetworking.pdf. Technical Report. Carleton University.

[7] Yousuf, M.S., El-Shafei, M., Power Line Communications: An Overview - Part I, 4th International Conference on Innovations in Information Technology, Nov. 2007

[8] Phase-shift keying. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012 [cit. 2012-04-03]. Dostupné z: http://en.wikipedia.org/wiki/Binary_Phase_Shift_Keying

[9] Pulse-position modulation. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012 [cit. 2012-04-03]. Dostupné z: http://en.wikipedia.org/wiki/Pulse_position_modulation

[10] AMERICAN SOCIETY OF HEATING, REFRIGERATING AND AIR-CONDITIONING ENGINEERS (ASHRAE). *BACnet* [online]. 2012 [cit. 2012-04-03]. Dostupné z: http://www.bacnet.org/

[11] GATES, Bill. A Robot in Every Home. *Scientific American* [online]. 2006, roč. 2007, č. 1, s. 6 [cit. 2012-05-03]. Dostupné z: http://www.scientificamerican.com/article.cfm?id=a-robot-in-every-home

[12] BIEN, Z. Zenn, Hyong-Euk LEE, Jun-Hyeong DO, Yong-Hwi KIM, Kwang-Hyun PARK a Seung-Eun YANG. INTELLIGENT INTERACTION FOR HUMAN-FRIENDLY SERVICE ROBOT IN SMART HOUSE ENVIRONMENT. *International Journal of Computational Intelligence Systems. Atlantis Press*. 2008, roč. 1, č. 1, s. 77-93. Dostupné z: www.atlantis-press.com/php/download_paper.php?id=1549

[13] Domestic robot. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012 [cit. 2012-04-03]. Dostupné z: http://en.wikipedia.org/wiki/Domestic_robot

[14] KWANG-HYUN, Park, ZEUNGNAM, Lee JU-JANG, BYUNG, Lim JONG-TAE, Kim JIN-OH, Lee HEYOUNG, Stefanov DIMITAR, Kim DAE-JIN, Jung JIN-WOO, Do JUN-HYEONG, Seo KAP-HO, Kim CHONG, Song WON-GYU a Lee WOO-JUN. Robotic smart house to assist people with movement disabilities. Springer Netherlands, 2007, č. 22, s. 183-198. DOI: 10.1007/s10514-006-9012-9. Dostupné z: http://dx.doi.org/10.1007/s10514-006-9012-9

[15] Ambrogio Robot: Ambrogio L400. ZUCCHETTI. *Ambrogiorobot* [online]. 2011 [cit. 2012-04-03]. Dostupné z: http://www.ambrogiorobot.org/index.php/ambrogio-l400.html

[16] Husqvarna Automower 305. HUSQVARNA. *Husqvarna* [online]. 2011 [cit. 2012-04-03]. Dostupné z: http://www.husqvarna.com/cz/products/robotic-mowers/automower-305/

[17] IROBOT. *IRobot Corporation*: *Robots that Make a Difference* [online]. 2012 [cit. 2012-04-03]. Dostupné z: http://www.irobot.com/

[18] ECHELON CORPORATION. *LonTalk Protocol Specification*. 3. vyd. 4015 Miranda Avenue, Palo Alto, CA, USA, 1994. Dostupné z: http://www.enerlon.com/JobAids/Lontalk%20Protocol%20Spec.pdf

[19] The World's Biggest Companies: Software & Programming. [online]. [cit. 2012-10-18]. Dostupné z: http://www.forbes.com/global2000/#p_1_s_a0_Software%20&%20Programming_All%20countries_All%20states_

[20] Choosing The Right Vacuum Cleaner For Your House. [online]. 2012 [cit. 2013-01-01]. Dostupné z: http://www.squidoo.com/vacuum-cleaners-for-sale

[21] BLUETOOTH SIG. *Bluetooth* [online]. 2013 [cit. 2013-01-01]. Dostupné z: www.buetooth.org

[22] IROBOT CORPORATION. *IRobot Create Owner's guide*. 2006. Dostupné z: http://www.irobot.com/filelibrary/create/Create%20Manual_Final.pdf

[23] IROBOT CORPORATION. *IRobot Create Open Interface_v2*. 2006. Dostupné z: http://ebookbrowse.com/create-open-interface-v2-pdf-d19654406

[24] DB-25. *Db-25 pinout*. West Chester: Aviom, 2013. Dostupné z: http://www.aviom.com/library/technical-resources/75_db25-pinout-information.pdf

[25] ARDUINO. *Arduino* [online]. 2013 [cit. 2013-01-01]. Dostupné z: http://arduino.cc/

[26] Electric Imp. *Devwiki.electricimp.com* [online]. 2013 [cit. 2013-01-01]. Dostupné z: http://devwiki.electricimp.com/doku.php

[27] MCCI. *PC/104 and small form factors*. Arlington, Texas, 2013.

[28] OJEDA, Lauro a Johann BORENSTEIN. Non-GPS Navigation for Emergency Responders. *Sharing Solutions for Emergencies and Hazardous Environments,*. 2005, s. 8

[29] CARLSON, Christopher R., J. Christian GERDES a David POWELL. Error Sources when Land Vehicle Dead Reckoning with Differential Wheelspeeds. *Error Sources when Land Vehicle Dead Reckoning with Differential Wheelspeeds*. 2003, s. 28.

[30] IRobot Create Robot. *Mbed.org* [online]. [cit. 2013-01-01]. Dostupné z: http://mbed.org/cookbook/iRobot-Create-Robot

[31] D. TTITERTON and J. WESTAON, "Strapdown Inertial Navigation Technology, 2nd Edition." *Progress in Astronautics and Aeronautics Series*, Published by AIAA, 2nd Edition, (2004).

[32] BORENSTEIN, Johann a Liqiang FENG. THE UNIVERSITY OF MICHIGAN. UMBmark: A Method for Measuring, Comparing, and Correcting Dead-reckoning

Errors in Mobile Robots [online]. Ann Harbour Michigan, 1995 [cit. 2013-01-23]. Dostupné z: http://www-personal.umich.edu/~johannb/Papers/umbmark.pdf

[33] A star algorithm. In: *Wikipedia*: *the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2012 [cit. 2013-1-20]. Dostupné z: *http://en.wikipedia.org/wiki/A*_search_algorithm*

[34] KUPAROWITZ, T. *Robotický fotbal v RDS.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 50 s. Vedoucí bakalářské práce Ing. Petr Honzík, Ph.D.

[35] HONDA. *Asimo The Honda HUMANOID ROBOT* [pdf]. Honda Motor Co., Ltd. Public Relations Division, 2007, 18 s. [cit. 2013-1-23]. Dostupné z: http://asimo.honda.com/downloads/pdf/asimo-technical-information.pdf

[36] NATIONAL AERONAUTICS AND SPACE ADMINISTRATION. *Robonaut 2 NASA Facts* [pdf]. Lyndon B. Johnson Space Center Houston, Texas 77058, 2007, 4 s. [cit. 2013-5-10]. Dostupné z: http://www.nasa.gov/pdf/464887main_Robonaut2FactSheet.pdf

[37] LUCAS, G.W. A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators. In: [online]. [cit. 2013-05-10]. Dostupné z: http://rossum.sourceforge.net/papers/DiffSteer/

[38] JULIA, Miguel, Arturo GIL, Luis PAYA a Oscar REINOSO. Local Minima Detection in Potential Field Based Cooperative Multi-robot Exploration. [online]. [cit. 2013-05-10]. Dostupné z: http://arvc.umh.es/documentos/articulos/ISAR08Exploration.pdf

[39] JOHNSON, Angus. Clipper: an open source freeware polygon clipping library. In: [online]. [cit. 2013-05-10]. Dostupné z: http://www.angusj.com/delphi/clipper.php

[40] JOHNS, Kyle a Trevor TAYLOR. MICROSOFT. *Professional Microsoft® Robotics Developer Studio*. 1. vyd. 10475 Crosspoint Boulevard, Indianapolis, IN 46256: Wiley Publishing, Inc. ISBN 978-0-470-14107-6.

[41] MORGAN, Sara. *Programming Microsoft Robotics Studio*. Washington : Microsoft Press A Division of Microsoft Corporation, 2008-03-12. 288 s. ISBN 9780735624320.

[42] SIEGWART, R a D SCARAMUZZA. ETH ZURICH - ASL. *Autonomous Mobile Robots*. Zurich, 2011. Dostupné z: http://www.asl.ethz.ch/education/master/mobile_robotics/year2011/Lecture_8.pdf

[43] THRUN, Sebastian, Dieter FOX, Wolfram BURGARD a Frank DELLAERT. Robust Monte Carlo Localization for Mobile Robots. 2000, roč. 2000, č. 125, s. 42.

[44] LG. *LG Eletronics* [online]. 2013 [cit. 2013-05-10]. Dostupné z: http://www.lg.com/

[45] Detailed technical information for PK 465 DN Board Camera. [online]. [cit. 2013-05-10]. Dostupné z: http://www.pacific-board-cameras.co.uk/colour_board_cameras/index/?product_id=29

[46] DSS Node Security. [online]. [cit. 2013-05-10]. Dostupné z: http://msdn.microsoft.com/en-us/library/dd771988.aspx

[47] List of device bit rates. *Wikipedia.org* [online]. [cit. 2013-07-20]. Dostupné z: http://en.wikipedia.org/wiki/List_of_device_bit_rates

[48] KIRILLOV, Andrew. Glyphs' recognition. [online]. 2010 [cit. 2013-07-20]. Dostupné z: http://www.aforgenet.com/articles/glyph_recognition/

[49] OBERKAMPF, Denis, Daniel F. DEMENTHON a Larry S. DAVIS. Iterative Pose Estimation Using Coplanar Feature Points. *Computer Vision and Image Understanding*. 1996, č. 3, s. 495-511.

[50] *AForge.NET* [online]. 2012 [cit. 2013-08-09]. Dostupné z: http://www.aforgenet.com/

[51] KING, Peter Haywood. *A Low Cost Localization Solution Using a Kalman Filter for Data Fusion*. Blacksburg, Virginia, 2008. Dostupné z: http://scholar.lib.vt.edu/theses/available/etd-05082008-095647/unrestricted/PHKThesis.pdf. Magisterská práce. Virginia Polytechnic Institute.

[52] LAVIOLA, Joseph J. Double Exponential Smoothing: An Alternative to Kalman Filter-Based Predictive Tracking. *Brown University Technology*. Dostupné z: http://www.eecs.ucf.edu/isuelab/publications/pubs/kfvsexp_final_laviola.pdf