České vysoké učení technické v Praze
Fakulta elektrotechnická

BAKALÁŘSKÁ PRÁCE

Porovnávání prostředí pro řízení mobilních robotů

Praha, 2009 **Ladislav Kopecký**

Poděkování

Děkuji všem, kteří se podíleli na vzniku této bakalářské práce. Především vedoucímu práce RNDr. Miroslavu Kulichovi, Ph.D. za odbornou pomoc při psaní práce. Dále rodičům a přátelům za podporu.

Prohlášení

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

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 20.6.2009

podpis

Abstrakt

Práce srovnává vlastnosti dvou prostředí pro řízení mobilních robotů. Je to vývojové prostředí Player/Stage, pro platformu Linux a vývojové prostředí Microsoft Robotics Developer Studio pro platformu Windows. Práce především porovnává obě prostředí po stránce architektury, nebo způsobu implementace nových projektů. Porovnání probíhá i v rovině vlastnosti prostředí jako průběh instalace prostředí a aktualizací, pořizovací cena, podpora pro uživatele, srozumitelnost dokumentace, přenositelnost vytvořených aplikací, nebo aplikace pro jejichž vývoj se prostředí používala.

Abstract

The aim of the thesis is to compare characteristics of two environments for mobile robots control. The first development environment is Player for Linux platform, while the second one is Microsoft Robotics Developer Studio for Windows platform. The thesis compares the environments with respect to used architecture as well as new project implementation. Installation, updating of the software, prize, user support, documentation intelligibility as well as portability of created applications are also studied.

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Ladislav Kopecký

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný

Obor: Kybernetika a měření

Název tématu: Porovnání prostředí pro řízení mobilních robotů

Pokyny pro vypracování:

1. Seznamte se s prostředím pro řízení mobilních robotů Microsoft Robotic Studio a podejte základní přehled jeho vlastností.
2. Seznamte se s prostředím pro řízení mobilních robotů Player/Stage/Gazebo a podejte základní přehled jeho vlastností.
3. Ve spolupráci s vedoucím práce vyberte úlohu, kterou posléze naimplementujete v obou prostředích.
4. Na základě vlastních zkušeností s implementací úlohy proveďte porovnání obou prostředí a případně navrhněte způsob, jakým lze obě prostředí kombinovat.

Seznam odborné literatury:

[1] http://playerstage.sourceforge.net/ [online][10.1.2008]
[2] http://msdn2.microsoft.com/en-us/robotics/default.aspx [online][10.1.2008]

Vedoucí bakalářské práce: RNDr. Miroslav Kulich, Ph.D.

Platnost zadání: do konce zimního semestru 2008/2009

L.S.

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 18.02.2008

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Mobilní robotika je velmi dynamicky se rozvíjející odvětví. Roboty se dostávají do všech sfér lidského života. Pomocí robotů se operují citlivé části těla, kterými jsou oči nebo mozek. Roboty jsou odolné a proto se používají pro podmořský výzkum nebo záchranné práce v zamořených prostorech. Robotické aplikace pomáhají kosmonautům při pracích v otevřeném vesmíru, vojákům při odstraňování výbušnin. V poslední době se roboty dostávají i do domácností v podobě automatických vysavačů nebo dětských hraček.

Většina samostatných akcí robota je jednoduchá, otočení motoru o určitý úhel, přečtení dat z čidla, nebo zobrazení zprávy na display. Pokud je robot složitější a je určen pro náročnější úlohy, jako je třeba chůze humanoida, musí se zkoordinovat několik akcí současně. Pro řízení takových složitých akcí je potřeba použít náročnějších programovacích technik, kterými jsou například vlákna nebo semafory. Prostředí Player/Stage a MRDS přinášejí systém programování, kde se o souběh vláken postará vývojové prostředí. Prostředí obsahují knihovny pro komunikaci s robotickým hardwarem, implementují algoritmy pro výpočty a rozpoznávání prostředí, obsahují simulační prostředí a další nástroje pro zjednodušení vývoje robotické aplikace.

1.1 Simulační prostředí

Při vytváření robotických aplikací programátor naráží na problém testování napsaného kódu. Pokud je kód psaný pro reálný hardware, je testování na tomto hardwaru nebezpečné, protože při skryté chybě může nastat stav, kdy se robot může poškodit. V tomto případě je vhodné použít pro testování napsaného kódu simulační prostředí, ve kterém se

simuluje robot a jeho chování v prostředí. Další výhodou použití simulačního prostředí je, že se programátor nemusí zabývat správnou funkcí robota, zda má nabité baterie, nebo zda správně pracuje datové spojení mezi robotem a výpočetním strojem. Nevýhodou simulace robotické aplikace je, že se nikdy nedají přesně simulovat reálné podmínky prostředí a reálné chování hardware. Vždy se mohou v reálném prostředí vyskytnout nepředpokládané stavy v okolí robota, jako je vlhkost, magnetická pole, nebo prašné prostředí, na které je třeba při programování myslet.

1.2 Obsah práce

Tato práce se věnuje srovnání dvou prostředí pro vývoj robotických aplikací. První je Microsoft Robotics Developer Studio, které vyvíjí firma Microsoft a je určeno pro platformu Windows. Druhým prostředím je vývojové prostředí Player/Stage s GNU licencí, které vyvíjí komunita *The Player Project* a je určeno pro platformy Linux, Solaris, Mac OS X, BSD a další POSIXové platformy. Porovnání vlastností prostředí je na konci každé kapitoly. Porovnání probíhá v rovině architektury prostředí, vlastností, podpory uživatelů, kvality dokumentace a projektech, které se v prostředích vyvíjejí.

V kapitole 2, *Hardware a software*, je popsán seznam používaných programů pro Windows a Linux, které byli při porovnávání vývojových prostředí používány. Dále je tu popsána základní konfigurace používaného vývojového počítače.

Kapitola 3, *Architektura*, je zaměřena na popis vývojového prostředí MRDS a Player/Stage, jejich architekturu, použité systémy komunikace robota a řídící části, struktury vytvářených programů a dalších jejich vlastností.

Ve 4. kapitole *Instalace vývojového prostředí* jsou přiblíženy vlastnosti, průběh instalace a potřebné součásti operačního systému, které musí v systému být, aby se vývojové prostředí nainstalovalo správně. Jsou zde popsány možnosti instalace, části instalačního balíku a jejich popis. V další části kapitoly je seznam knihoven, které jsou dostupné po nainstalování vývojových prostředí a vlastnosti těchto knihoven.

Kapitola 5, *Instalace aktualizací*, přináší přehled vydaných verzí, datum vydání a zásadní změny, které proběhly při vydání nové verze. Je zde popis toho, v jakých formátech aktualizace vychází, jaké jsou výhody a nevýhody tohoto formátu a jak často vycházejí nové aktualizace.

V kapitole 6, *Podpora pro uživatele*, jsou sepsány všechny možné zdroje informací, ze

kterých mohou uživatelé čerpat, pokud se dostanou k problému který nemohou vyřešit. Jsou zde uvedeny blogy vývojářů na kterých informují o novinkách, které vyvinuli, nebo které se připravují pro vývoj.

Kapitola 7, *Srozumitelnost dokumentace*, navazuje na kapitolu 6 a popisuje, které dokumentace jsou vydané a jaké mají formáty. Dále se popisuje jak jsou tyto dokumentace zpracované a na jaké části jsou rozdělené.

Další je kapitola 8, *Podporované robotické platformy*, kde je seznam podporovaného robotického hardware. Jsou zde uvedeny fotografie tohoto hardware a popis komunikace s hardwarem, kterou podporují vývojová prostředí.

Kapitola 9, *Přenositelnost vytvořené aplikace*, přináší přehled systémů na, kterých je možné spustit vytvořené aplikace. Porovnání výhod a nevýhod aplikací vytvořených v prostředí Player/Stage a prostředí MRDS. Je zde popsána vlastnost obou prostředí, že server a klient mohou být umístěni na odlišných operačních systémech.

Kapitola 10, *Praktické využití vývojových prostředí*, představuje několik projektů, které byli vyvíjeny v prostředích MRDS nebo Player/Stage.

V kapitole 11, *Cena prostředí*, je popsáno, jak moc finančně náročné je pořídit si vývojové prostředí a zda je výhodné tuto cenu za vývojové prostředí investovat.

Kapitola 12, *Implementace aplikace*, je vlastně návod na to jak v obou prostředích implementovat aplikaci na řízení robotů. Tato aplikace řeší problematiku toho, aby se robot vyhýbal překážkám (obstacle aviod).

Kapitola 2

Hardware a software

2.1 Použitý robot

Použil se reálný robot Lego NXT Mindstorms. Prostředí MRDS má pro NXT zpracované nejrozsáhlejší návody a příklady zdrojových kódů ze všech podporovaných robotů. NXT je vlastně stavebnice Lego, do které se dají zabudovat robotické součásti. Předně to jsou senzory zvuku, světla a barev, ultrazvukový senzor a dotykový senzor. Pohybové části jsou 3 motory s rozlišením pohybu na 1 stupeň. Všechny výpočty provádí „kostka“ ve které je 32-bitový mikroprocesor Atmel AT91SAM7S256, který pracuje na frekvenci 48MHz a obsahuje 256KB FLASH a 64KB RAM paměti. Součástí je i koprocesor 8MHz, 4KB FLASH a 512Byte RAM. Jako komunikační rozhraní je dostupné USB konektor a Bluetooth rozhraní. Lego NXT má vlastní firmware, který poskytuje služby softwarového API. Toto API je možné modifikovat pro různé programovací jazyky. Na internetu lze nalézt API pro Visual Basic, platformu .NET, Java nebo Perl.

Obrázek 2.1: Robot NXT v konfiguraci nazvané Tribot

2.2 Software pro Windows

Pro testování prostředí Microsoft Robotics Developer Studio byl použit Windows Vista Bussines SP1. Při testování vlastností a chování prostředí bylo použito několik programů. V tabulce 2.1 je uveden název programu, verze a popis programu.

Název	**Verze**	**Popis**
Windows Vista Bussines SP1	Longhorn	Operační systém Windows
Robotics Studio Academic Edition	2008	Programování robotických aplikací
Visual Studio Professional 2008	9.0.21022.8	Prostředí pro vývoj aplikací
.NET Framework	3.5 SP1	Podpora přístupu k jádru Windows
Firefox	3.0.10	Internetový prohlížeč

Tabulka 2.1: Programy používané pro testování MRDS

2.3 Software pro Linux Ubuntu

Při testování prostředí Player/Stage byl zvolen operační systém Linux Ubuntu 8.10, protože Ubuntu má velmi rozsáhlou komunitu uživatelů. Z toho vyplývá snadný přístup k informacím o nastavení systému. Další velká výhoda Ubuntu je snadná instalace. Při psaní práce byla vydána nová verze Ubuntu 9.01 s několika změnami. Mohl jsem tedy vyzkoušet, zda se změní chování prostředí Player/Stage. Změny Ubuntu 9.01 se nedotkly fungování prostředí Player/Stage, navíc nová verze přinesla odstranění jedné chyby při instalaci. Odpadla potřeba do systému kopírovat soubor s definicí barev viz. kapitola 4.2.4.

2.4 Použitý vývojový počítač

Pro porovnávání vlastností a chování obou prostředí pro řízení mobilních robotů byl použit notebook Fujitsu Siemens Esprimo V5545 (dále jen FSE). Notebook má konfiguraci: Intel Core2Duo CPU 2GHz, 2GB RAM, 160GB harddisk.

Název	**Verze**	**Popis**
Ubuntu Intrepid Ibex	8.10	Operační systém Linux
Player	2.1.2	Programování robotických aplikací
Stage	2.1.1	Plugin simulační 2D prostředí pro Player
Gazebo	0.8-pre3	Plugin simulační 3D prostředí pro Player
Firefox	3.0.10	Internetový prohlížeč

Tabulka 2.2: Programy používané pro Ubuntu

Kapitola 3

Architektura

3.1 Architektura MRDS

3.1.1 Microsoft Robotics Developer Studio (MRDS)

Microsoft Robotics Developer Studio (MRDS) [1] je platforma vyvinutá společností Microsoft pro tvorbu algoritmů nebo programů pro řízení robotických aplikací a jejich realtimovou simulaci v trojrozměrném simulačním prostředí. Vývoj algoritmů je možný pro širokou škálu různých platforem a hardware. Důležitá vlastnost je, že se dají přidávat vlastní části rozhraní pro vývoj, řízení a simulaci specifického hardware.

Prostředí MRDS je distribuováno ve třech verzích. Academic verze pro akademické použití šířená přes MSDNAA a DreamSpark. Placená verze Standart pro komerční použití v ceně 10 000 Kč. Verze Express je dostupná ke stažení z webu MRDS pro použití zdarma, neobsahuje však všechny nástroje jako verze Academic/Standart.

Pro snadné programování a velkou variabilitu vývoje v prostředí MRDS je možné vyvíjet robotické algoritmy v několika programovacích jazycích. Stěžejním programovacím jazykem je C# (Csharp) nebo Visual Basic.Net (VB.NET), podpora je ale i pro další jazyky, jako je JScript, Microsoft IronPython a C++. Prostředí MRDS poskytuje i možnost implementovat rozšiřující rozhraní pro další jazyky. Microsoft žádné další rozhraní nenabízí a proto by si uživatel musel rozhraní sám naprogramovat.

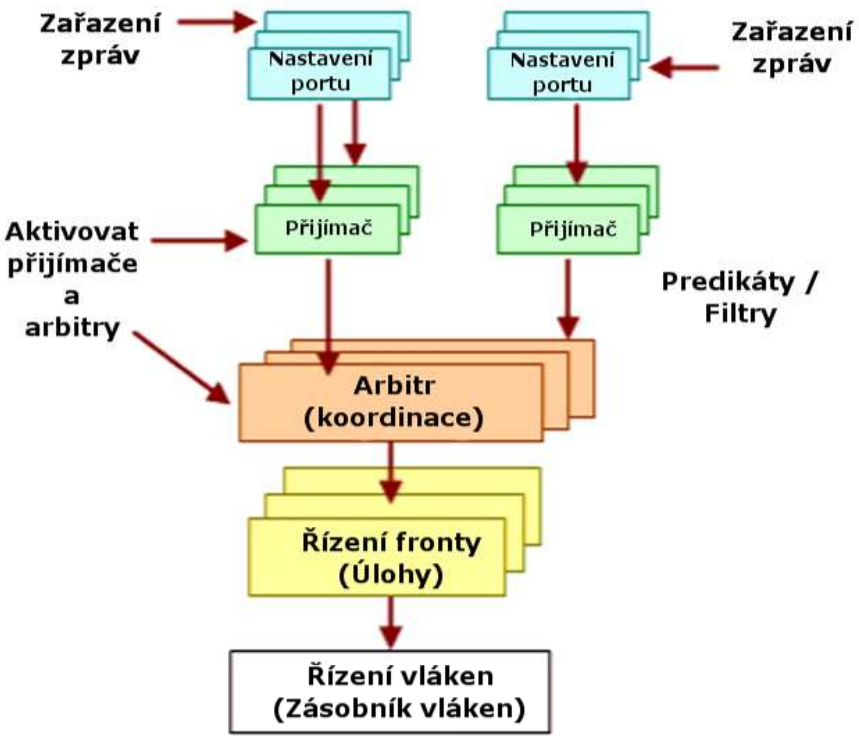
Pro vývojáře, kteří neovládají žádný programovací jazyk, je v prostředí MRDS implementována podpora Microsoft Visual Programming Language (VPL). Je to vizuální programovací jazyk, ve kterém se programuje pomocí systému chyť a táhni (drag and drop). Pro programátory, kteří programují v pokročilém programovacím jazyce, je v prostředí MRDS podpora pro programování v Microsoft Visual Studiu (MVS), nebo v jiném pro-

gramovacím prostředí, které podporuje jazyky MRDS. Například to může být NetBeans nebo Eclipse. Výhoda používání MVS je v tom, že jsou pro MVS připraveny šablony pro nové projekty a návody jak programovat.

MRDS nabízí integrované prostředí založené na platformě .Net (dotNet). Prostředí MRDS obsahuje množství knihoven, které obstarávají funkce pro robotické aplikace.

3.1.2 Concurrency and Coordination Runtime (CCR)

Hlavní knihovna je Concurrency and Coordination Runtime (CCR), která obstarává koordinaci mezi činnostmi robotů, jako jsou například pohyb, informace ze snímačů, reakce na informace a další činnosti podle vlastností robota. Je to vlastně multi-taskingový (více úlohový) správce zodpovědný za koordinaci více-úlohových aplikací pro MRDS. Zjednodušeně řečeno, spravuje vlákna a zajišťuje jejich správné sladění a součinnost. Architektura CCR je uvedena na obrázku 3.1, který je převzat z webové stránky [2].



Obrázek 3.1: Architektura CCR

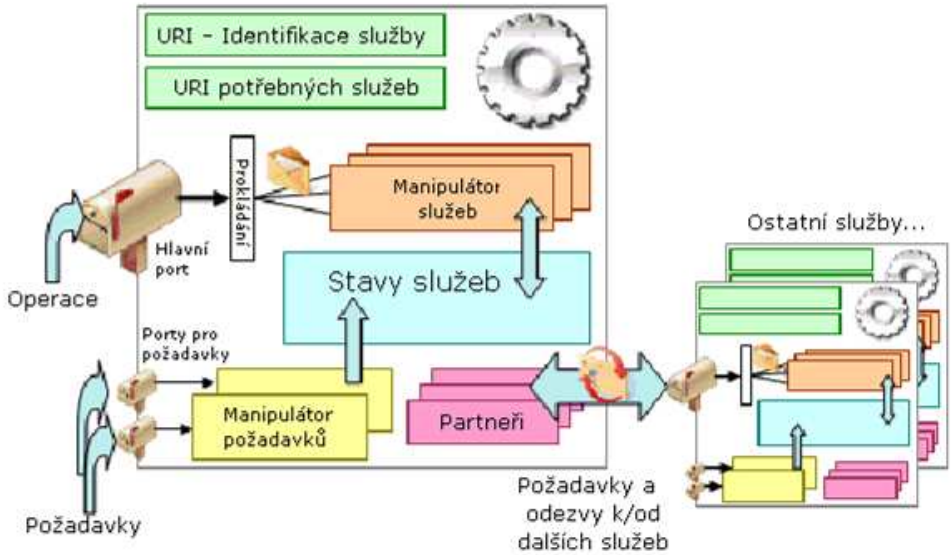
Když přijde příkaz (zpráva) od řídící aplikace přes nastavený port aktivuje se přijímač a arbitr. Přijímač přijme zprávu, zjistí její druh a pošle zprávu arbitrovi, který ji rozřadí

do fronty zpráv podle druhu a priority zprávy. Zprávy spustí úlohu ve vytvořeném vlákně, které je řízeno a spravováno v zásobníku vláken.

Služby jsou obsažené v knihovnách a poskytují přístup k naprogramovaným funkcím. Tyto funkce se používají pro komunikaci s hardwarem, provádí výpočty pokročilých algoritmů, vytváření simulačního prostředí, řízení motorů, čtení dat ze senzorů a další robotické funkce.

3.1.3 Decentralized Software Services (DSS)

Další důležitou knihovnou je Decentralized Software Services (DSS), která je grafickou nadstavbou pro CCR. Vytváří grafické prostředí pomocí Web Forms (webových formulářů), nebo Windows Forms (formuláře pro Windows). Přes tyto formuláře se přistupuje k systémovým událostem a službám knihovny CCR. DSS představuje běhové prostředí, které dovoluje službám vyměňovat si zprávy bez ohledu na to, kde v síti se nalézají. Je to tedy prostředí pro komunikaci mezi službami. Na jedné straně jsou služby, které zpracovávají data a provádí výpočty. Na druhé straně jsou služby, které sbírají data ze senzorů a vykonávají příkazy pro ovládaní robotických aplikací. Služby jsou funkce, které jsou poskytované jednotlivými knihovnami. Zjednodušená architektura služeb je uvedena na obrázku 3.2, který je převzat z webové stránky [2].

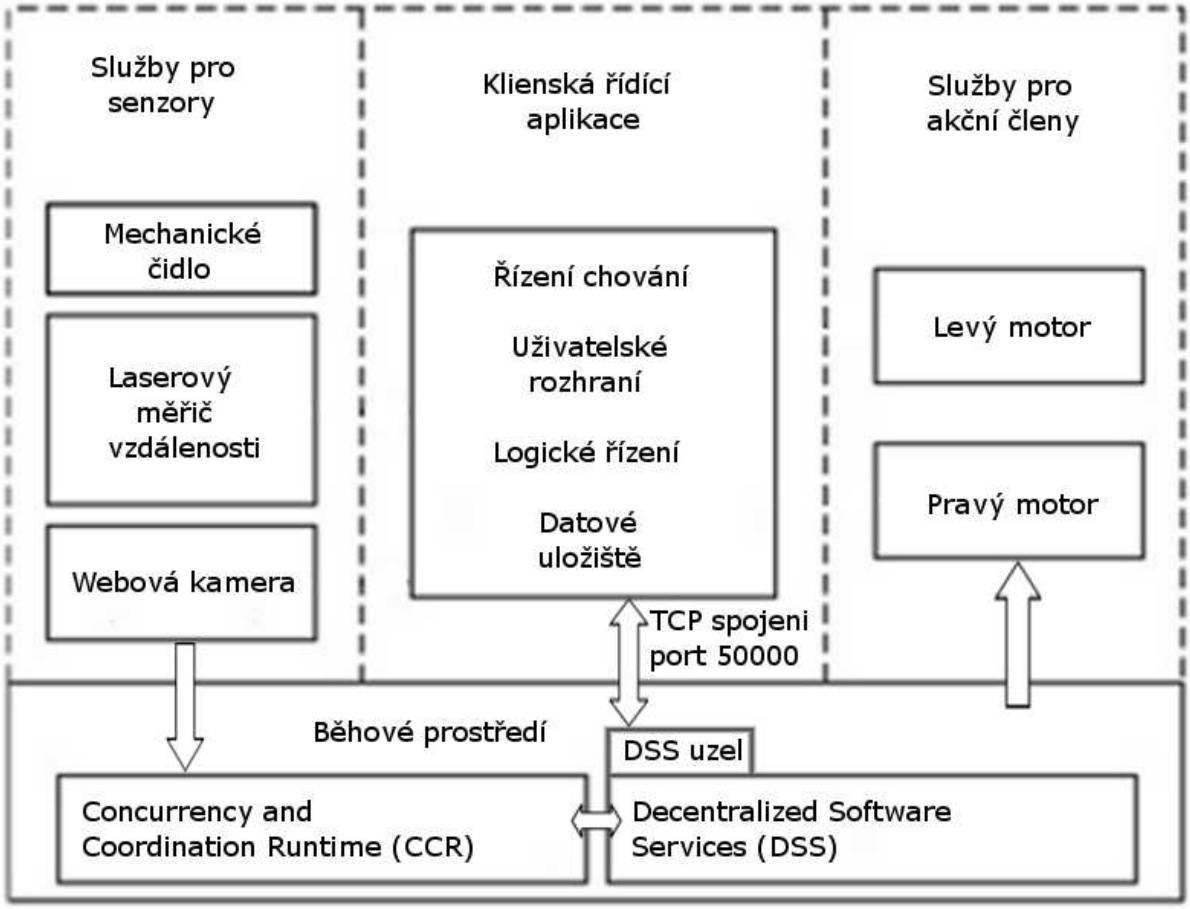


Obrázek 3.2: Architektura služeb

Každá vytvořená služba má unikátní *identifikátor URI*, který reprezentuje službu

v celém prostředí MRDS. Složitější služby mají v hlavičce *reference* na další služby, které jsou ve službě používané. Když vytváříme novou službu nástrojem pro MRDS jsou v hlavičce importované služby pro čtení manifestů a pro práci s protokolem DSSP. Na *hlavním portu* se připojuje aplikace a získává funkce od služeb. Data jsou *prokládána* aby se mohla získávat z více služeb zároveň. Na *porty pro požadavky*, přicházejí požadavky na funkce služeb. V *manipulátoru požadavků* se požadavky třídí a žádají partnerské služby o potřebné funkce. Koordinace mezi požadavky na služby a daty odcházejících od služeb probíhá ve stavech služeb.

Schéma komunikace CCR, DSS, služeb a klientské aplikace je na obrázku. Služby čtou data ze senzorů a předávají je přes CCR a DSS do logického řízení. V logickém řízení se data zpracovávají a po zpracování dat se posílají řídící příkazy přes DSS a CCR na služby akčních členů. Veškeré koordinace služeb řídí běhové prostředí CCR s přispěním DSS, které navíc komunikuje s klientskou řídící aplikací.



Obrázek 3.3: Architektura běhového prostředí MRDS

3.1.4 DSS Command Prompt (DSSCP)

DSSCP(Decentralized Software Services Command Prompt - DSS příkazová řádka) vzhledem vypadá jako klasická příkazová řádka systému Windows viz. obrázek 3.4, ale je napojena na služby a nástroje pro prostředí MRDS. Klasická příkazová řádka Windows nepodporuje příkazy prostředí MRDS. Při spuštění DSSCP se cesta adresáře automaticky směřuje do adresáře MRDS. Seznam nástrojů pro DSSCP:

- *DssInfo.exe* - nástroj, který prozkoumá knihovnu uvedenou v parametru a zobrazí všechny její závislosti na ostatní knihovny i informace, kde je knihovna použita. Informace se uloží do uvedené složky. Je možné si vybrat uložení dat do souboru HTML nebo XML. Nástroj je možné používat i pro více knihoven najednou a vytvořit tak seznam a popis používaných knihoven.

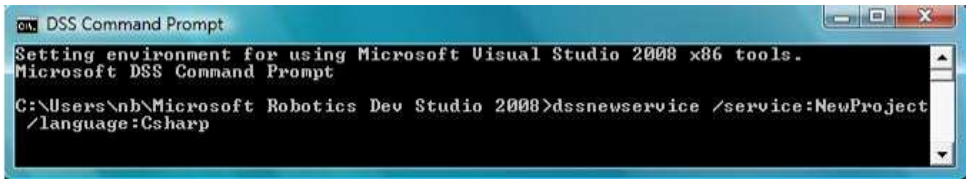
- *DssDeploy.exe* - vytváří a rozbaluje archivy s příponou .exe. Archiv je balíček, který obsahuje soubory MRDS potřebné pro fungování vytvořené robotické aplikace bez nainstalovaného prostředí MRDS. Archivy jsou samo spustitelné nebo se dají rozbalit pomocí nástroje *dssdeploy.exe*.

- *DssHost.exe* - vytvoří DSS uzel na definovaném TCP portu. DSS uzel je rozhraní, ke kterému se připojí klientská aplikace. DssHost spouští jednu nebo více služeb a zobrazuje jejich výpis v příkazové řádce DSSCP. Pokud požadujeme sestavení aplikace pro vzdálené připojení, použijeme argument pro vytvoření IP adresy pro identifikaci aplikace v síti. Když chceme spustit projekt MRDS, lze to provést příkazem z DSSCP *dsshost /m:manifest.Manifest.xml*. Nástroj *DssHost.exe* načte konfigurační manifest, který je zadán v parametru a spustí příslušné služby, které jsou v manifestu nadefinovány.

- *DssNewService.exe* - vytvoří nový projekt, který se použije jako výchozí bod pro vývoj nové robotické aplikace. Tento projekt je vytvořen pro MVS, ale může být editován i v jiném vývojovém prostředí. Je možnost projekt vytvořit v podporovaných jazycích (C#, Visual Basic.Net (VB.NET), JScript, Microsoft IronPython, C++)

- *DssProjectMigration.exe* - dovoluje přenesení projektu z jedné verze MRDS do jiné verze, především se používá pro přenos projektu z MVS 2005 na projekt v MVS 2008 a přenos mezi aplikacemi vytvořenými v MRDS 1.5 do aplikací pro MRDS 2008

- *DssProxy.exe* - nástroj, který vytváří zastoupení, přesměrování a transformaci služeb. Automaticky se spouští při použití nástroje *DssNewService.exe*
- *HttpReserve.exe* - rezervuje port pro uživatele nebo doménu. Tento nástroj se používá, pokud potřebujeme vyhradit port pro aplikaci.



Obrázek 3.4: Vytváření nového projektu v příkazové řádce MRDS

Pokud spouštíme aplikaci v prostředí MRDS jinak než z příkazové řádky DSSCP a v aplikaci se vyskytne chyba, DSSCP sice tuto chybu zobrazí, ale jen na několik málo okamžiků, takže ani není možné tuto chybu přečíst. Toto chování se vyskytuje i u klasické příkazové řádky systému Windows a ztěžuje odhalení chyby.

3.1.5 Soubory Manifest

Manifest je soubor založený na otevřeném standartu XML s příponou .xml nebo .Manifest.xml, který obsahuje tagy - značky, které popisují jednotlivé objekty a jejich vlastnosti. Manifesty jsou uloženy především v adresáři *MRDS\samples\Config*. Soubory manifest definují vlastnosti objektů používaných v prostředí MRDS. Manifesty se používají především jako konfigurační soubory a rozdělují se na tři druhy.

3.1.5.1 Konfigurační Manifest

První druh manifestu se pozná podle párového tagu <*Manifest*>. Tyto manifesty popisují vlastnosti robotického hardware (motor, řídící deska, čidlo a další). Především definují komunikační vlastnosti hardwaru jako jsou porty, rychlost přenosu, druh rozhraní, závislosti na služby a na konfigurační soubory. Tento manifest je možné upravovat a vytvářet pomocí editoru DSSME nebo textovým editorem. Tento druh manifestu se vytváří automaticky při exportu projektu z prostředí VPL do projektu MVS.

3.1.5.2 Manifest pro dokumentaci

Druhý druh manifestu je označen párovým tagem <*doc*>. Tento manifest popisuje služby definované v knihovnách, které byli vytvořeny při kompilaci nové aplikace. Je to soubor dokumentace, který se používá při generování dokumentačních souborů nástrojem *DssInfo*. Tento manifest se nedá upravovat editorem DSSME, ale lze jej upravovat pomocí textového editoru nebo editoru pro dokumentaci. Soubor se automaticky vytváří při kompilaci aplikace v prostředí MVS nebo při exportu aplikace z prostředí VPL.

3.1.5.3 Manifest s popisem simulační scény

Třetí druh manifestu je označen párovým tagem <*SimulationState*>. Tento manifest popisuje stav simulačního prostředí, vložené entity a jejich vlastnosti, chování simulačního prostředí, gravitace a další vlastnosti simulace. Manifest nelze upravovat pomocí DSSME, ale lze jej upravovat pomocí textového editoru nebo načtením do prostředí VSE upravením entit a opětovným uložením *Save Scene As*. Pokud scénu uložíme tímto způsobem, vytvoří se dva manifesty. Jeden obsahuje simulační popis s příponou .xml, který může opět otevřít v prostředí VSE a upravovat. Druhý manifest obsahuje popis referencí na služby. Pomocí něj lze spustit simulaci z příkazové řádky DSSCP. Tento manifest má příponou .Manifest.xml. Pokud chceme spustit simulaci z příkazové řádky DSSCP, musíme použít manifest s příponou .Manifest.xml a příkazem pro DSSCP *dsshost /port:50000 /tcp-port:50001 /m:Apl.manifest.xml* simulaci spustíme. Tento příkaz zobrazí pouze simulační scénu v prostředí VSE. Tento postup je vhodný pro upravování vzhledu simulační scény. Pokud máme ve scéně nadefinovaného robota a chceme jej ovládat, musíme se k simulačnímu prostředí připojit řídícím programem na portech uvedených při spouštění simulace z DSSCP.

3.1.5.4 Příklad konfiguračního Manifestu

Na příkladu 3.1.5.4 je uveden Manifest, který se použije při připojení hlavní „kostky“ NXT k robotické aplikaci. Adresy HTTP uvedené v textu odkazují na služby, které jsou definované v knihovnách. Nejsou to klasické HTTP adresy odkazující na server. Popis jednotlivých služeb lze generovat nástrojem *dssinfo* z knihovny robotické aplikace, která se vygeneruje při kompilaci aplikace v prostředí MVS.

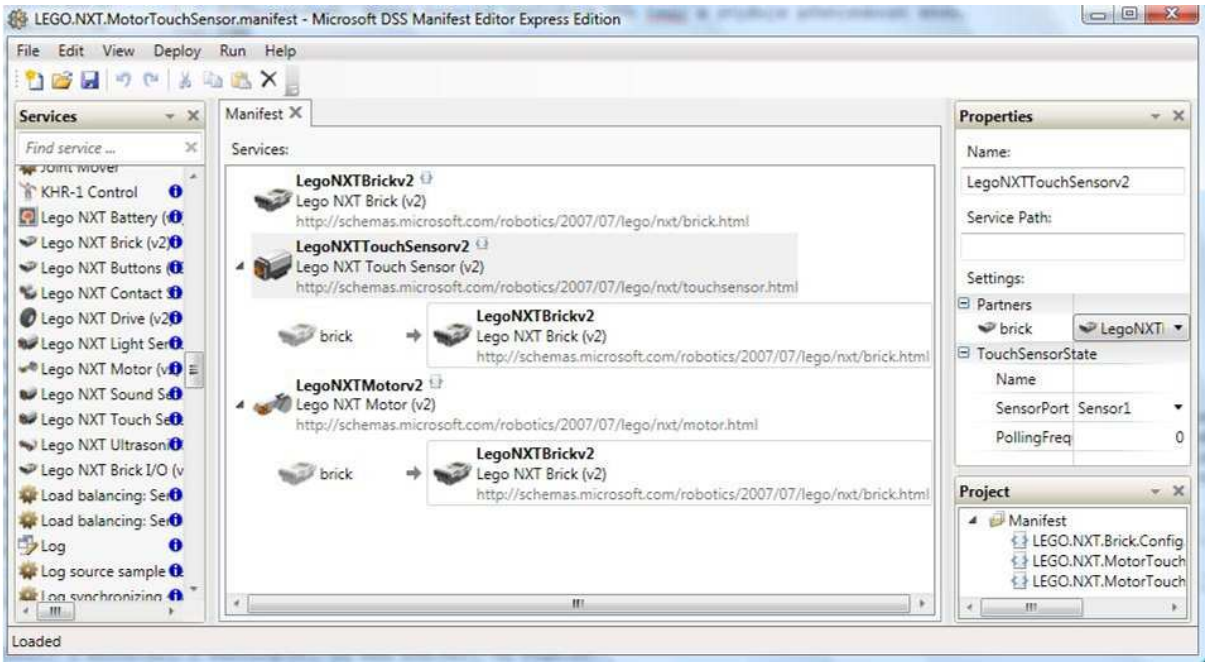
```
Příklad manifestu pro komunikaci s řídící "kostkou" NXT

<Manifest
  #reference na službu, která poskytuje funkce pro komunikaci s NXT
   xmlns:brick="http://schemas.microsoft.com/robotics/2007/07/lego/nxt/brick.html"
  #identifikátor manifestu
   xmlns:this="urn:uuid:2843e826-f443-4338-910c-14169734f723"
  #reference na služby knihovny DSS
   xmlns:dssp="http://schemas.microsoft.com/xw/2004/10/dssp.html"
  #reference na služby pro čtení manifestů
   xmlns="http://schemas.microsoft.com/xw/2004/10/manifest.html">

  <CreateServiceList>   #vytvoření seznamu služeb
    <ServiceRecordType> #fuknce načítání služeb
      <dssp:Contract>   #služba pro komunikaci přes DSS protokol
        #služba, která se načte
         http://schemas.microsoft.com/robotics/2007/07/lego/nxt/brick.html
      </dssp:Contract>
      <dssp:PartnerList />
      <Name>this:LegoNXTBrickv2</Name>  #název vytvořené služby
    </ServiceRecordType>
  </CreateServiceList>
</Manifest>
```

Příklad 3.1.5.4: Manifest pro komunikaci s řídící „kostkou“ NXT

3.1.5.5 Vytváření a úprava Manifestů

Editování a vytváření manifest souborů je možné v libovolném textovém editoru, ale tato cesta je velmi složitá. Programátor musí znát všechny syntaxe všech tagů, které se používají v prostředí MRDS manifestech. Druhá cesta je pomocí nástroje DSS Manifest Editor (DSSME). DSSME na obrázku 3.5 je grafický editor pro vytváření a úpravu manifest souborů.

Nové manifest soubory se vytváří skládáním již existujících služeb, které se načítají z knihoven ze složky MRDS. DSSME při každém startu prochází složky MRDS, aby nalezl existující manifesty a služby. Start editoru se pak může protáhnout na celou jednu minutu. Výhoda tohoto startu je, že při každém startu DSSME jsou načteny opravdu jen existující služby a manifesty. Když DSSME spouštíme častěji je již start rychlejší. V levé části je seznam načtených služeb, které je možné do manifestu vkládat. Služby se vkládají přetažením vybrané služby na pracovní plochu. V pravé části DSSME lze jednotlivým službám měnit nastavení.

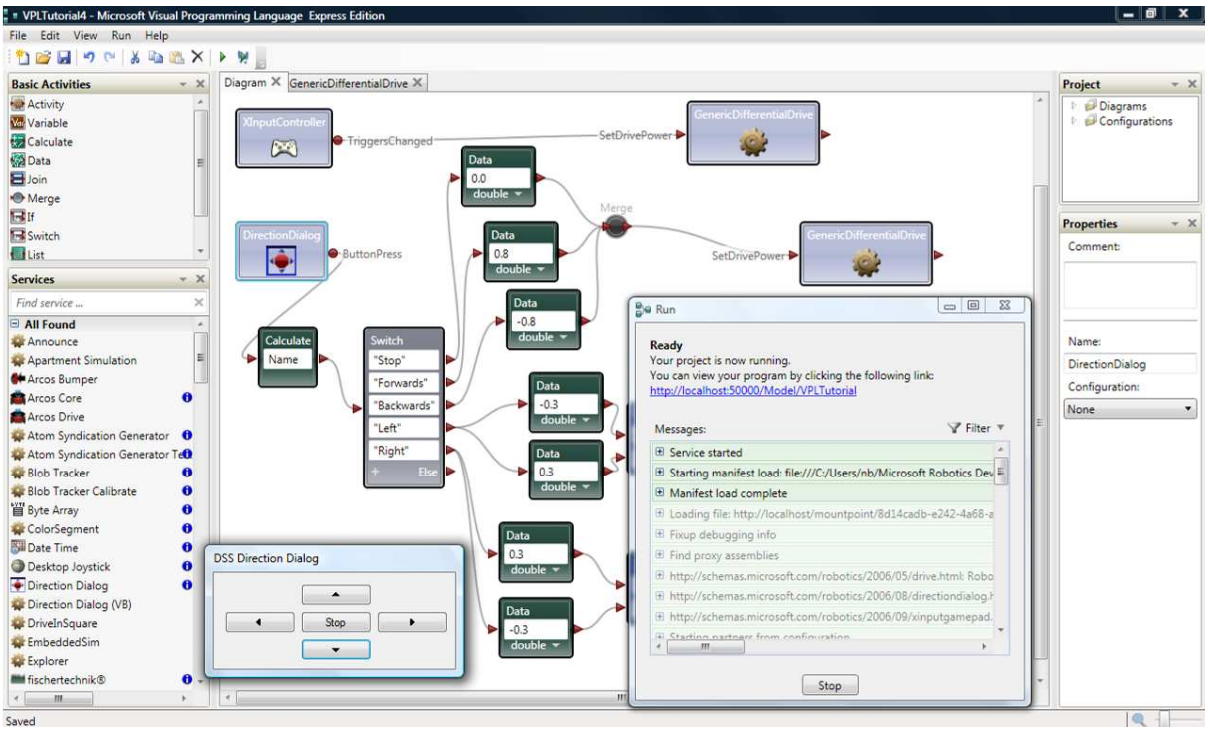


Obrázek 3.5: DSSME s načteným manifestem pro aplikaci s NXT

3.1.6 Microsoft Visual Programming Language (VPL)

Vzhled prostředí VPL je na obrázku 3.6. Je to visuální programovací prostředí vyvinuté přímo pro použití služeb z knihoven DSS a CCR. Programování se provádí stylem drag and drop (chyť a táhni). To znamená, že v programovacím prostředí máme v levém sloupci seznam dostupných služeb a ty vkládáme přetažením na pracovní plochu. Tyto služby lze mezi sebou spojovat a vytvářet tak inteligentní aplikace. V tomto prostředí je možné naprogramovat celou řídící aplikaci. Je určené pro vývojáře, kteří neovládají žádný programovací jazyk, ale používají jej i zkušení programátoři pro rychlé sestavení jednodušších služeb, jejich propojení a export do projektu pro MVS. Projekt je exportován v jazyce C# a do složky projektu se přidají i všechny potřebné knihovny a manifesty. Je to velmi jednoduchá a rychlá cesta, jak vytvořit nový projekt. Touto cestou si vytvoříme vlastní manifest a projekt se službami pro danou robotickou aplikaci. Pomocí souborů manifest připojíme ke službám součásti robota nebo jejich simulace v prostředí VSE.

Problém může nastat, když chceme v prostředí VPL programovat složitější struktury. Především se ve velkém množství služeb můžeme snadno ztratit. Na obrázku 3.6 je vidět prostředí VPL se spuštěným projektem, který obsahuje služby rozhodovací struktury a naprogramované dialogové okno s řídícím křížem pro ovládání pohybu robota.



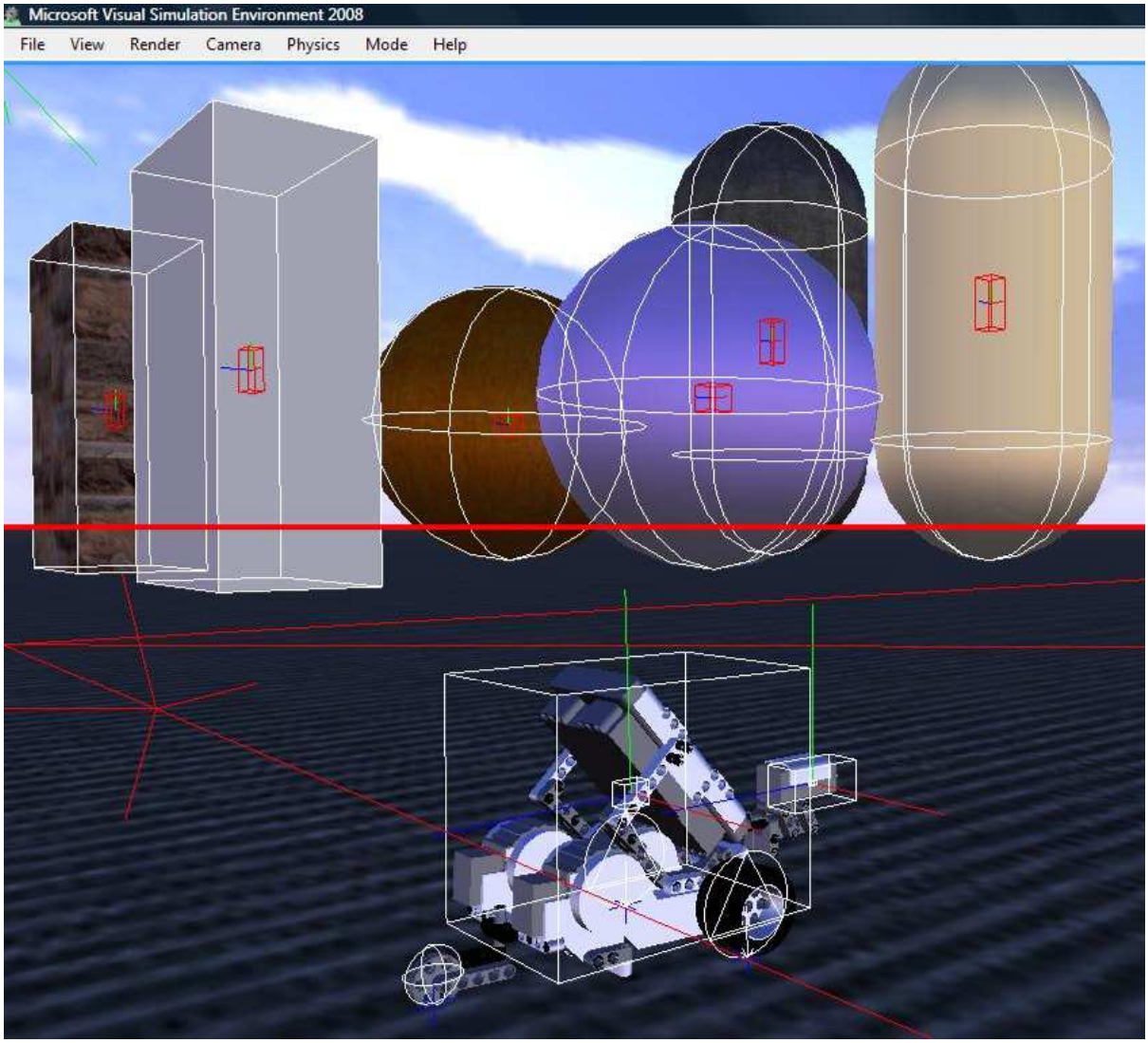
Obrázek 3.6: Prostředí VPL s řídícím křížem

3.1.7 Microsoft Visual Simulation Environment (VSE)

MRDS nabízí pro simulace robotických aplikací simulační prostředí VSE (Visuální simulační prostředí). V tomto prostředí se simuluje reálné chování entit (objektů), se zachováním fyzikálních zákonů. Struktura VSE je založena na *Simulation Engine Service*, která vykresluje entity a výpočet simulačního času. Dále sleduje stav simulačního prostředí a simuluje entity. Další významnou složkou je *Managed Physics Engine Rapper*, která přináší zjednodušené příkazy pro komunikaci s API (Application Programming Interface - rozhraní pro programování aplikace). Jednou z dalších aplikací rozšiřující vlastnosti simulačního prostředí je *AGEIA PhysX Technology*. Tato technologie dokáže používat hardware zvyšující akceleraci a kvalitu simulace chování robotů v simulačním prostředí pomocí *AGEIA PhysX* procesorů. Nejdůležitější částí simulačního prostředí je aplikace *Entities*. Je to aplikace, která v simulačním prostředí reprezentuje roboty a fyzické objekty jako například stěny, překážky a další simulované předměty. Simulační prostředí je napsané v prostředí XNA, což je prostředí pro programování her pod Windows.

Simulační prostředí VSE pro MRDS nabízí třídimenzionální (3D) pohled na simulovanou scénu s možností přepínání mezi vloženými kamerami viz. obrázek 3.7. S kamerami je

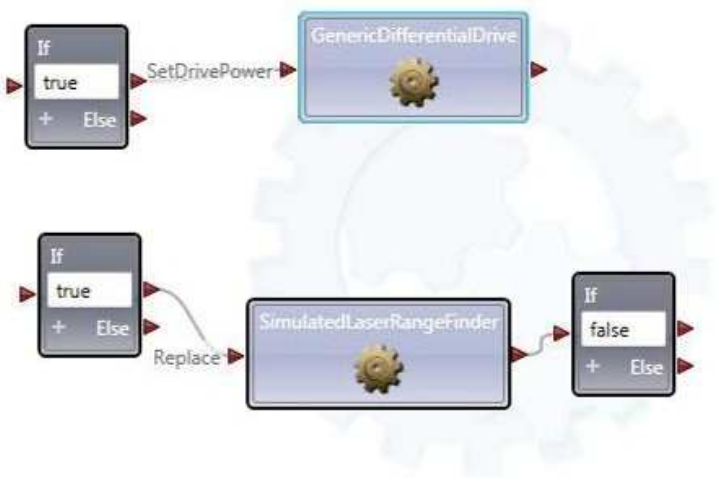
možné pohled plynule přibližovat, oddalovat, plynule měnit úhel pohledu ze všech stran, shora i zdola. Je možné přidávat, odebírat a editovat entity při přepnutí simulačního prostředí do editačního módu. Vkládané entity mohou být ve formátu Collada (soubor pro ukládání dat z animačních programů například z programu SolidWorks). Simulační prostředí v editačním módu je zobrazeno na obrázku 12.1 v kapitole 12.1.2. Lze měnit zobrazení z klasické animace na zobrazení modelů s vykreslením hran, nebo zobrazení os x, y, z, každé z entity. Zobrazení os entit je na obrázku 3.7.



Obrázek 3.7: VSE v zobrazení os x, y, z entit

Pro vyzkoušení funkčnosti prostředí VSE si můžeme v prostředí VPL sestavit jednoduchou aplikaci. Například si vytvoříme aplikaci pro NXT, aby robot jezdil stále dokola. Nejjednodušší je vložit v prostředí VPL blok *IF* s podmínkou „true“, aby simulovaný

robot jezdil stále. Poté vložíme blok *GenericDifferentialDrive*, který vytváří dva motory s jedním řízením. Propojíme bloky *IF* a *GenericDifferentialDrive* a vybereme typ spojení *From: TrueChoice* a *To: SetDrivePower* a hodnoty pro motory nadefinujeme pro *LeftWheelPower 0.8* a pro *RightWheelPower 0.7*. Tyto hodnoty zajistí, že NXT bude jezdit stále dokola. Maximální hodnota *WheelPower* pro motory NXT je 1. Do služby *GenericDifferentialDrive* importujeme manifest *LEGO.NXT.Tribot.SimulationEngineState.xml*. Tento manifest definuje simulační prostředí a robota NXT. Sestavená služba je na obrázku 3.8

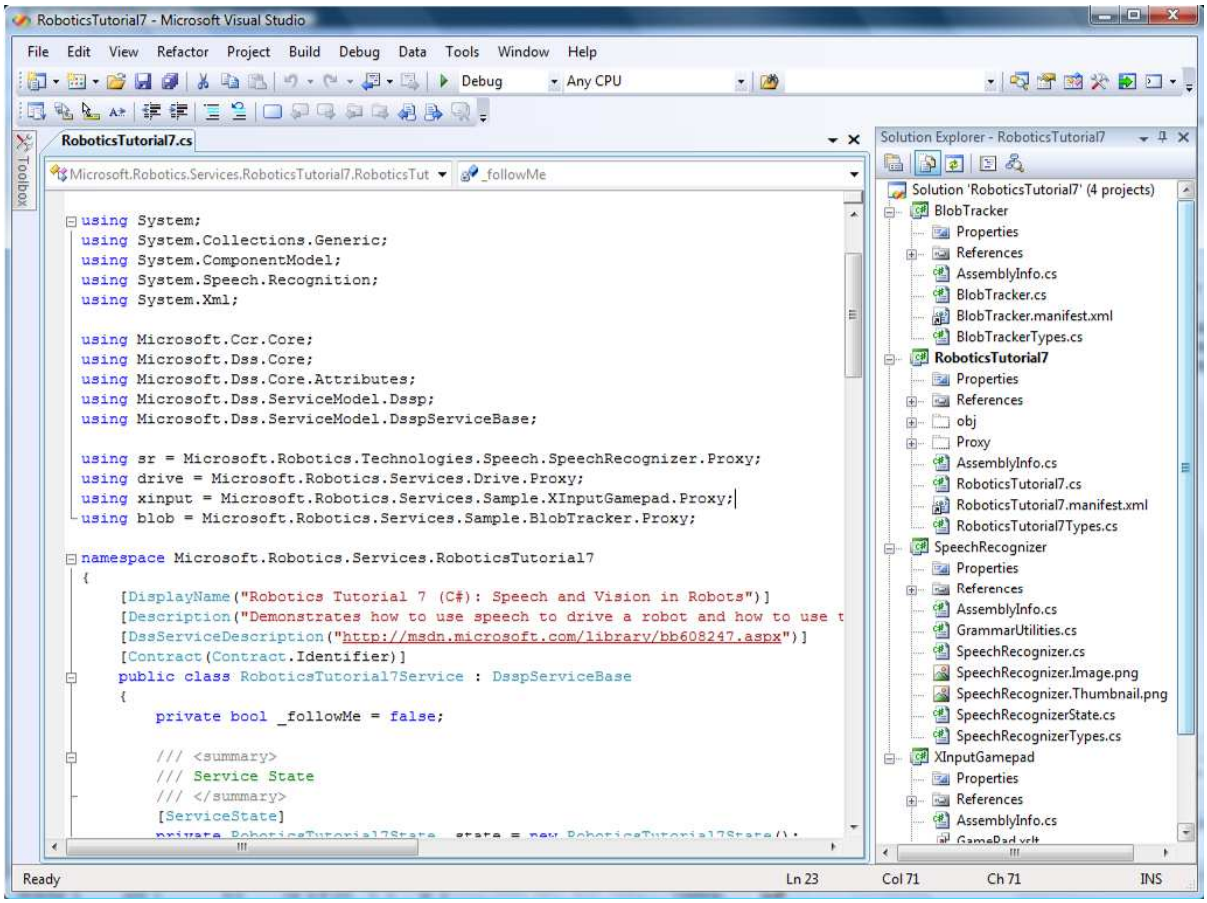


Obrázek 3.8: Sestavené služby pro vyzkoušení simulace

3.1.8 Microsoft Visual Studio (MVS)

Pokud chceme programovat robotické aplikace opravdu efektivně, je třeba použít jen jedno vývojové prostředí pro všechny části projektu, čímž se sníží problémy s kompletací projektu. Vytvoření manifest souborů, naprogramování řídícího programu, sestavení simulační scény, následné propojení všech částí a spuštění projektu. Všechny tyto kroky se dají efektivně programovat v prostředí MVS s použitím nástrojů MRDS. Na obrázku 3.9 je vidět MVS s načteným C# projektem. Na začátku kódu jsou vidět importované knihovny robotických funkcí.

Nejvhodnější jazyk pro programování robotických aplikací pomocí MVS je C#, pro který jsou zpracovány všechny návody a tutoriály pro prostředí MRDS. Pro ostatní jazyky jsou zpracovány pouze některé návody a tutoriály. Velkou výhodou při psaní v jazyce C# v prostředí MVS je IntelliSense. Je to nástroj na inteligentní doplňování kódu integrované do MVS. Pro vývoj robotických aplikací MRDS není potřeba placená verze prostředí



Obrázek 3.9: Projekt MRDS načtený ve Visual Studiu

MVS, ale lze použít verzi Express, která je k dostání zdarma.

Pro zjednodušení práce je vhodné používat nástroje prostředí MRDS. Následující nástroje generují kód v C#, který se může použít v prostředí MVS pro doplňování částí projektu. V seznamu jsou uvedené některé nástroje, které usnadňují vývoj robotické aplikace.

- Vytvoření nového projektu nástrojem *DssNewService*
- Vytvoření entity pomocí programů Autocad, GoogleSketchUp, nebo jiných animačních programů
- Sestavení simulační scény v prostředí VSE a uložení do manifestu
- Sestavení manifestu nástrojem DSSME

3.2 Architektura Player/Stage

3.2.1 Player/Stage

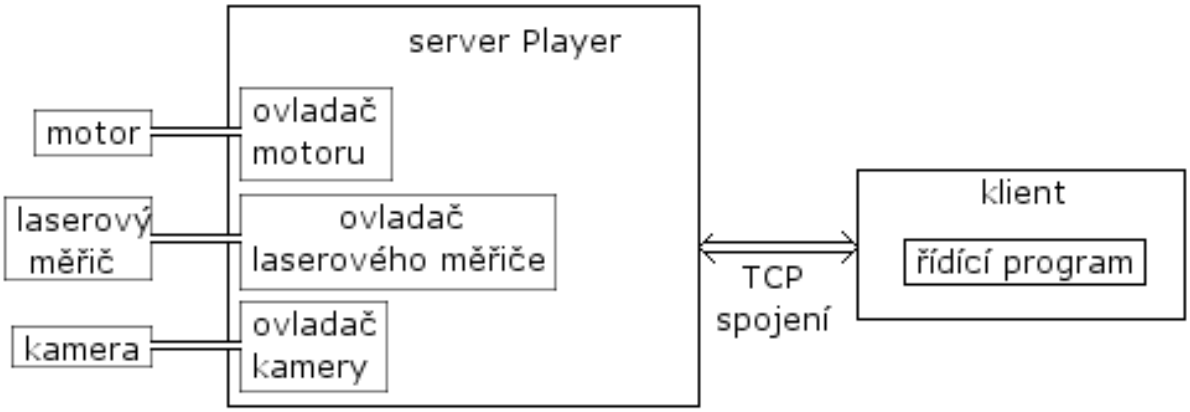
Prostředí pro řízení mobilních robotů Player/Stage je projekt s GNU licencí s volně dostupnými zdrojovými kódy vyvíjenými v komunitě kolem projektu. Player/Stage podporuje platformu Linux (PC a embedded), Solaris, Mac OS X, BSD a další platformy splňující standard Posix. V prostředí Player/Stage je implementovaná podpora pro programovací jazyky C++, Tcl, Java a Python. Architektura prostředí Player/Stage je navržena tak, aby byla nezávislá na použitém jazyce a platformě. Tato vlastnost je největší předností prostředí Player/Stage.

Programy v prostředí Player/Stage mohou být napsány tak, aby byli co nejmenší. To znamená, že se používají pouze potřebné knihovny a funkce. Tak vytvoříme program přesně podle požadavků, bez zbytečných knihoven a funkcí. V některých případech tento programovací přístup může projektu uškodit, protože se zdrojový kód může stát nepřehledným.

Při práci s prostředím Player/Stage jsou základními součástmi, server Player vytvořený na robotickém hardware a klient, který se připojuje k serveru Player. Server poskytuje data ze senzorů a ovládá akční robotické členy. Klient zpracovává přijatá data a vydává pokyny k ovládání robotických členů. Data ze serveru Player se nemusí používat jen pro výpočty v řídícím programu. Data lze ukládat na klientovi a slouží pro monitorování chování robota v prostředí. Klientem je libovolný počítačový systém a může být umístěn i na jiném operačním systému. Používané jsou klasické PC, integrované počítačové desky nebo integrovaná zařízení.

Spojení serveru Player a klienta probíhá přes komunikační rozhraní TCP. Přes TCP připojení server Player podporuje libovolný počet připojených robotů nebo klientů. Spojení je umožněno také vzájemně mezi robotickými aplikacemi. V prostředí Player/Stage je každé zařízení možné konfigurovat takzvaně za „letu“, to znamená během provádění operací.

Architektura prostředí Player/Stage je založena na modulárním schématu. Základní bloky jsou server - klient a k nim se připojují ovladače pro hardware viz. obrázek 3.10. Algoritmy použité pro jednotlivé aplikace, jakými jsou zjišťování polohy, řízení motorů, zpracování dat laserových měřičů a ostatní se pro odlišné robotické aplikace používají stejné, pouze se mění ovladače pro daného robota.



Obrázek 3.10: Zjednodušená architektura Playeru

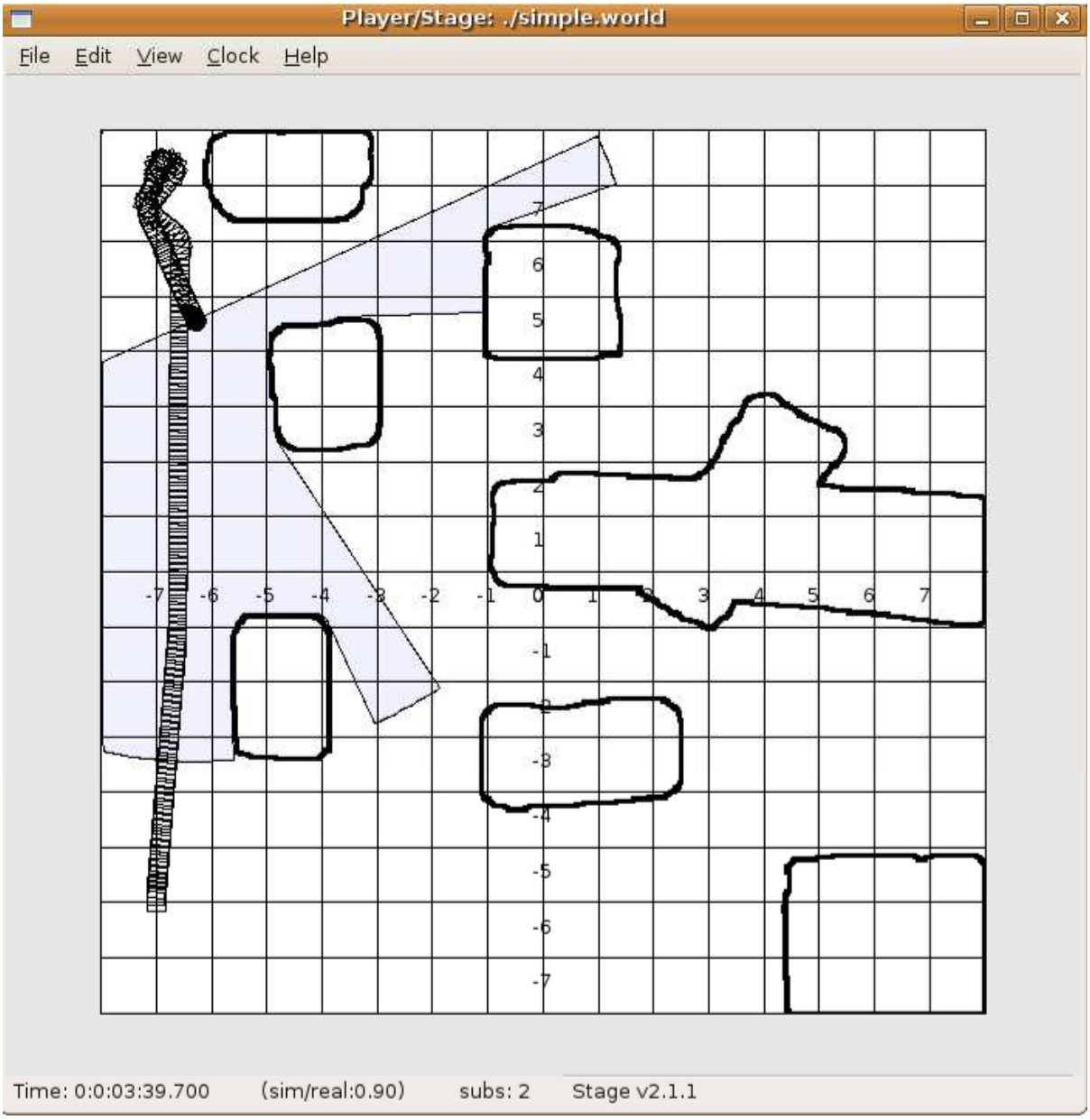
3.2.2 Stage

Stage je 2D simulační prostředí pro server Player. Simulace se spustí na serveru Player a simuluje chování robotického hardware. Pro připojení simulační aplikace k serveru Player se používá ovladač, stejně jako pro připojení reálného hardwaru. Když tedy vyvíjíme aplikaci v prostředí Player/Stage je vhodné si připravit dva konfigurační soubory, které budou téměř stejné, jen se budou se v použití ovladačů. Jeden konfigurační soubor s ovladači pro reálný hardware a druhý s ovladači pro simulační prostředí Stage.

Simulace probíhá v 2D grafice s půdorysovým pohledem. Stage má grafické prostředí uvedené na obrázku 3.11, kde se zobrazuje průběh simulace. V simulačním prostředí je možné sledovat několik simulovaných veličin.

- Dráhu pohybu robota
- Mřížku, která rozdělí prostředí na oblasti
- Mřížka rozpoznání překážek
- Zobrazení prostoru rozsahu detektoru robota (laserový detektor nebo sonar)
- Umístění robota v souřadnicích

Podkladové simulační prostředí je jen bitmapa, kterou Stage prozkoumá a místa v obrázku, kde jsou čáry převede na překážky.



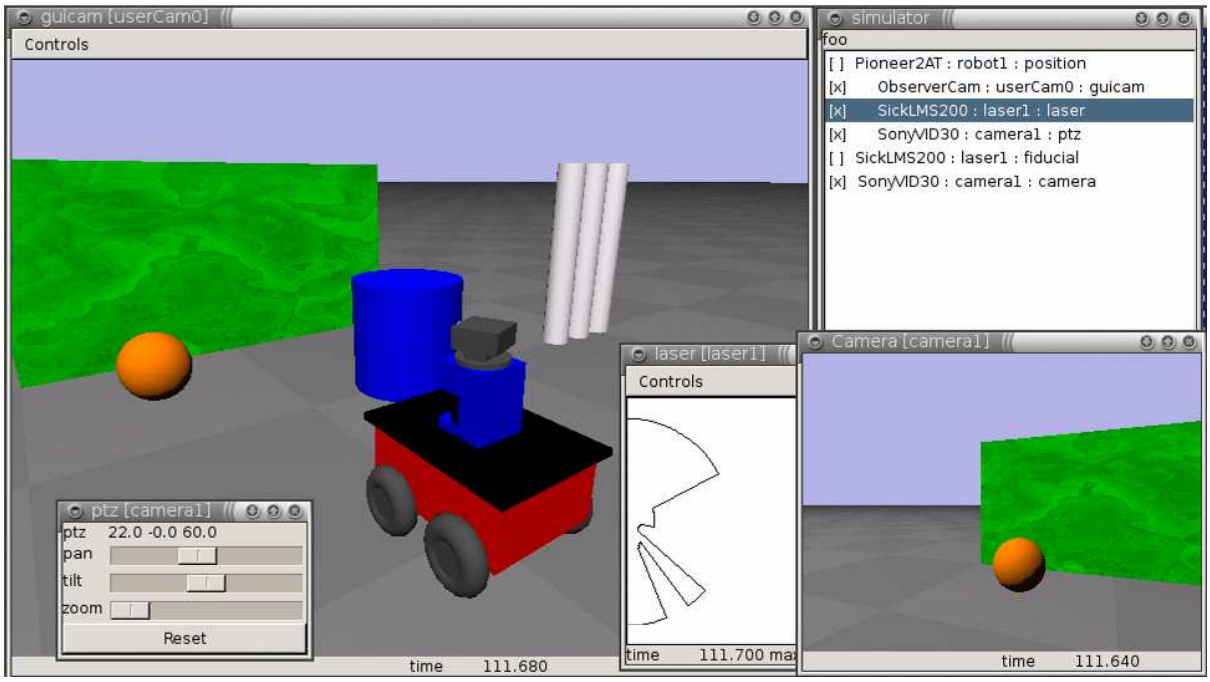
Obrázek 3.11: Grafické okno simulačního prostředí Stage

3.2.3 Gazebo

Vývoj aplikací lze v prostředí Player/Stage simulovat i ve 3D pohledu. Tuto 3D simulaci poskytuje pro prostředí Player/Stage plugin Gazebo. Gazebo je zobrazeno v grafickém okně viz. obrázek 3.12. Systém simulace je stejný jako v prostředí Stage.

V simulačním prostředí Gazebo lze simulovat několik robotů ve venkovním prostředí. Simulovat lze klasické senzory, jako je laserový měřič, sonar, kamery, GPS a další. Pod-

pora je pro simulaci běžných robotů jako jsou například P2DX, P2AT, nebo Segway. Simulování probíhá v prostředí, kde platí fyzikální zákony. V nejnovější verzi Gazebo je možné importovat na roboty vzhled vytvořený v modelovacích programech. Zobrazení je možné přepínat mezi klasickým zobrazením animace a drátovým modelem. Je možné zobrazovat i místa kde snímají například laserové snímače.



Obrázek 3.12: Grafické okno simulačního prostředí Gazebo

Kapitola 4

Instalace vývojového prostředí

4.1 Instalace vývojového prostředí MRDS

4.1.1 Distribuované verze

Microsoft Robotics Developer Studio je distribuováno ve třech verzích. Express Edition, která je pro použití zdarma, ale nemá implementovány všechny funkce. Verze Academic Edition a Standart Edition jsou funkčně shodné. Academic Edition je ve většině případech šířena prostřednictvím MSDNAA (program pro šíření výrobků Microsoft pro akademické použití) a DreamSpark (program pro šíření výrobků Microsoft pro střední školy) a je ke stažení zdarma. Verze Standart Edition je určena především pro komerční sektor a je distribuována jako placená verze.

4.1.2 Rozdíly ve verzích

Rozdíl ve verzích lze porovnat na webové stránce [3]. MRDS Express Edition je dostupná ke stažení z webových stránek MRDS [4]. Tato verze je k použití zdarma, ale chybí v ní několik nástrojů a vlastností, které jsou obsaženy ve verzi Academic/Standart. Především je to .NET Compact Framework Support, což je podpora programování mobilních aplikací s přístupem přímo do jádra operačního systému Windows Mobile. Dále nelze v simulačním prostředí VSE editovat entity a jejich vlastnosti přímo. Do VSE nelze importovat vzhled entit ve formátu Collada a počet entit je omezen na 64. Při používání VPL nelze exportovat vytvořené diagramy do projektu pro MVS v jazyce C#. Další znevýhodnění MRDS Express Edition je, že nepodporuje instalaci aktualizací a je tedy třeba stáhnout a nainstalovat celou novou verzi programu.

4.1.3 Průběh instalace MRDS

Průběh instalace MRDS byl testován na dvou systémech Windows. Na systému Windows XP SP3 Professional bylo testováno pouze chování instalace. Na systému Windows Vista SP1 byla testována instalace a používání MRDS. Instalátor je proveden aplikací InstallShield Wizard ve verzi 14.0.0.162.

Jednou z vlastností Windows je, že s počtem nainstalovaných programů se systém zpomaluje a při běžném používání se do systému instalují různé balíčky a nástroje. Nešlo by tedy poznat, které balíčky potřebuje MRDS při instalaci. Proto se testování chování instalace provedlo i na Linuxovém virtuálním stroji VirtualBox 2.2.0 [5] na čistě nainstalovaném systému Windows Vista. Po spuštění instalace MRDS na virtuálním stroji trvalo naběhnutí instalačního průvodce dvě a půl minuty. Tato časová prodleva se potvrdila i na klasické instalaci Windows Vista a Windows XP. Tato doba spouštění instalace je při instalaci všech verzí MRDS. Při této době instalátor kontroluje přítomnost potřebných instalačních balíčků v počítači. Při první instalaci na nově nainstalovaný systém se po spuštění instalačního programu MRDS zobrazí okno, ve kterém jsou zobrazeny závislosti na potřebné instalační balíčky, bez kterých by MRDS nefungovalo správně. Jsou to závislosti na balíčky uvedené v tabulce 4.1.

Název balíčku	**Popis balíčku**
Microsoft Visual C++ 2008 SP1 Redistributable Package (x86)	Balíček s knihovnami obsahujícími služby potřebné pro programování v C++
Microsoft XNA Framework 2.0	Knihovna potřebná pro simulaci ve fyzikálním prostředí
NVIDIA PhysX System Software	Prostředí pro fyzikální simulování chování robotů
Microsoft .NET Compact Framework 3.5	.Net Framework pro mobilní systémy Microsoft
Microsoft CCR and DSS Runtime 2008	Knihovny pro podporu CCR a DSS

Tabulka 4.1: Přehled balíčků potřebných pro instalaci MRDS

Pokud tyto balíčky nejsou obsažené v počítači, automaticky se nainstalují před samotnou instalací MRDS. Instalování těchto balíčků trvá přibližně minutu a potom instalátor přepne na instalování MRDS. V průběhu instalace je možné vybrat Kompletní nebo Volitelnou instalaci. Ve volitelné instalaci je možné si vybrat místo na disku, kam se aplikace

MRDS nainstaluje a které součásti se nainstalují. Na výběr jsou součásti z tabulky 4.2.

Název balíčku	**Popis balíčku**
Runtime a Tools	Knihovny CCR a DSS, nástroje a příkazová řádka pro MRDS, šablony MVS pro služby DSS a DSS .NET Compact Framework
Robotics Common	Obsahuje knihovny pro robotické funkce MRDS
VPL	Visuální programovací prostředí
Simulation Environment	3D Simulační prostředí
Documentation	Dokumentace a nápověda pro MRDS
Samples	Příklady a průvodci pro programování v MRDS
.NET Compact Framework Support	Podpora MRDS pro .NET Compact Framework (podpora programování pro mobilní zařízení)

Tabulka 4.2: Přehled instalovaných součástí MRDS

Celková doba potřebná pro instalaci se pohybuje kolem čtyř minut. Může se lišit podle verze použitého Windows. Pokud jsou v počítači již nainstalované některé potřebné balíčky, doba instalace se snižuje.

Po instalaci se do nabídky Start systému Windows přidá složka obsahující zástupce pro spuštění nástrojů MRDS. Především jsou to zástupci na spuštění Dokumentace prostředí MRDS, popis DSS a CCR knihoven, příkazový řádek DSSCP, DSS Manifest Editor, zástupci pro spouštění příkladů simulací.

Základní přehled knihoven nabízených po instalaci prostředí MRDS je v tabulce 4.3. Podrobné popisy knihoven jsou v dokumentaci na webu [6].

Při prvním spuštění jakékoliv aplikace vytvořené v prostředí MRDS je potřeba povolit firewall a přidat výjimku pro další spouštění. Důvodem je, že aplikace vytvořené v prostředí MRDS se spouští jako webová služba a data jsou poskytována přes TCP rozhraní.

Pro spuštění simulace ve fyzikálním prostředí je potřeba mít v počítači grafický adaptér s podporou Direct 3D a shaders v2.0. Pro programování v prostředí MRDS není jiný speciální hardware potřeba.

Nevýhodou instalace prostředí MRDS je to, že velikost instalačního souboru i místo zabrané instalací je relativně velké a to v řádech stovek megabajtů (ve verzi MRDS 2008 má instalační soubor 400MB).

Název knihovny	**Funkce**
Microsoft.Ccr.Adapters	Funkce grafického prostředí CCR (WPF a WinForms)
Microsoft.Dss.Core	Základní funkce pro DSS
Microsoft.Dss.Hosting	Poskytuje hostování DSS runtime libovolné .NET aplikaci
Microsoft.Dss.Runtime	Uděluje oprávnění pro služby a uživatele
Microsoft.Dss.Schemas	Sestavuje zprávy pro komunikaci s ostatními službami
Microsoft.Dss.ServiceModel	Sestavuje zprávy pro komunikaci přes DSS protokol
Microsoft.Dss.Services	Služby pro práci s DSS
Microsoft.Robotics.PhysicalModel	Služby pro fyzikální model
Microsoft.Robotics.Services	Služby čidel, baterii, pohonu, kamer, koordinace
Microsoft.Robotics.Simulation	Služby simulace

Tabulka 4.3: Přehled knihoven MRDS

Pokud nejdříve nainstalujeme prostředí MRDS ve verzi Express Edition a potom chceme instalovat verzi Standart/Academic, je potřeba odstranit všechny soubory z harddisku po verzi Express Edition, nebo nainstalovat novou verzi jinam na harddisk, jinak se nová verze nainstaluje znovu ve verzi Express Edition.

Pokud byla v počítači nainstalovaná starší verze MRDS, jsou dostupné i starší verze šablon *DSS Service* pro MVS. I když starší verzi MRDS odinstalujeme, zůstanou starší šablony *DSS Service* v adresáři projektů MVS v Dokumentech. Starší šablony nejsou funkční, proto je potřeba je odebrat ručně.

4.2 Instalace vývojového prostředí Player/Stage

4.2.1 Možnosti instalace

Instalace prostředí Player/Stage je možná z repositáře Ubuntu přímo jako instalační balíček pro Debian/Ubuntu (Ubuntu je klon Debianu) příkazem *robot-player_<verze>_i386.deb*. V repositáři Ubuntu je Player ve verzi 2.0.4. Na oficiálním serveru [7] je Player ve verzi 2.1.2, což je o celou jednu vývojovou verzi novější vydání. V logovacím souboru se změnami provedenými od verze Player 2.0.4 je několik důležitých oprav a z tohoto důvodu je lepší stáhnout a zkompilovat nejnovější oficiální verzi.

4.2.2 Vlastnosti instalace

Instalace Playeru byla provedena na nově nainstalovaný systém Ubuntu8.10 Intrepid. Do nové instalace byly staženy všechny nejnovější aktualizace. Pro zjištění všech závislostí v balíčcích byl systém Ubuntu8.10 nejdříve instalován ve virtuálním stroji VirtualBox 2.2.0 [5]. Po zjištění závislostí potřebných balíčků byl Player provozován již jen na klasické instalaci Ubuntu8.10 na notebooku FSE. Balíčky potřebné pro instalaci Playeru na systém Ubuntu8.10 jsou uvedeny v tabulce 4.4.

4.2.3 Instalace Player

Při konfiguraci (*./configure*) Playeru je možné si vybrat, které ovladače se budou instalovat. Je také důležité, kam se prostředí Player/Stage nainstaluje. Nejlepší je nainstalovat prostředí Player/Stage do samostatného adresáře. Pokud použijeme základní nastavení, je potom obtížné nalézt všechny složky, kam byl Player/Stage nainstalován. Po konfiguraci jsou vypsány ovladače, které nejsou běžně zahrnuty do konfigurace, nebo které se nepodařilo nalézt a zkonfigurovat. Pokud tyto ovladače chceme používat, je potřeba je nainstalovat a znovu konfigurovat instalaci prostředí Player/Stage.

Po úspěšné konfiguraci se Player nainstaluje použitím *make install*. Kompilace a instalace Playeru je celkem časově náročná, přibližně sedm minut. Ušetřit čas a místo na disku při kompilaci a instalaci lze zkrátit vypnutím kompilace všech ovladačů. Když ovladače při konfiguraci vypneme, mohou později chybět při komunikaci s hardwarem.

Název balíčku	**Popis obsahu balíčku**
debhelper(>=5)	Automatizace typických úloh vytváření debianích balíků
autotools-dev	Update infrastruktury souborů config.guess,sub
libtool	Obecná knihovna podpory skriptů
libmagick++9-dev	C++ API pro ImageMagick knihovnu
libgsl0-dev	GNU Scientific Library (GSL)
libcv-dev	Knihovny pro OpenCV (Open Computer Vision)
libhighgui-dev	Knihovny GUI pro OpenCV (Open Computer Vision)
libgtk2.0-dev	Multiplatformní sada nástrojů (toolkit) pro GUI
libdc1394-22-dev	Programovací rozhraní pro IEEE1394 digitální kamery
libboost-signals-dev	Správa signálů a slotů pro knihovny C++
libboost-thread-dev	Knihovna pro C++ více vláknové aplikace
swig	Generuje skriptovací rozhraní pro C/C++
libjpeg62-dev	Knihovny pro práci se soubory IJG JPEG
python-central (>=0.5)	Registrace a vytváření nástrojů pro Python
doxygen	Dokumentační systém pro C, C++, Java, Python, ...
linux-kernel-headers	Virtuální balíčky Linuxu
libgnomecanvas2-dev	Výkonný objektově-orientovaný zobrazovací nástroj
python-dev	Knihovny pro Python
freeglut3-dev	OpenGL Utility Toolkit
python-setuptools	Rozšíření pro Python - utility pro komplexní distribuce

Tabulka 4.4: Vývojové balíčky potřebné pro instalaci Playeru v Ubuntu

4.2.4 Instalace Stage

Pro simulaci robotických aplikací vytvořených v prostředí Player je potřeba nainstalovat modul Stage. Instalace se opět provádí nejdříve konfigurací, kde se zjistí závislosti knihoven a potom kompilací a instalací. Největším problémem při instalaci Stage na Ubuntu8.10 bylo to, že v systému chyběl soubor, ve kterém jsou uvedeny kódy barev "rgb.txt". Tento soubor je potřeba stáhnout z internetu a přidat do Ubuntu. V nejnovější verzi Ubuntu 9.04 Jaunty je již chybějící soubor doplněn.

Pokud se Player a Stage správně nainstalují, je potřeba přidat jejich umístění do konfiguračních cest pro správné vyhledání Playeru a Stage. Po kompletní instalaci a konfiguraci systému je vhodné ověřit správné fungování Playeru pomocí předem připravených

příkladů.

4.2.5 Knihovny poskytované prostředím Player a Stage

Po nainstalovaní prostředí Player a Stage jsou dostupné nástroje a knihovny s ovládáním pro roboty a senzory. Seznam těchto nástrojů a knihoven je uveden v tabulkách 4.5 a 4.6. Přesné podrobnosti o knihovnách lze nalézt na webová stránce [8].

Název balíčku	**Popis obsahu balíčku**
libstage2	Knihovny pro simulační prostředí Stage
libstageplugin1	Plugin pro spojení Playeru a Stage
stage	Simulační prostředí Stage

Tabulka 4.5: Knihovny poskytované po instalaci Stage

4.3 Porovnání instalace Player/Stage a MRDS

Protože prostředí MRDS vychází ve třech verzích, může se instalace u jednotlivých verzí lišit. Instalace MRDS je běžná instalace programu ve Windows. Nepřináší příliš možností nastavení, ale takový typ instalací je pod Windows zvykem, protože jsou koncipovány i pro méně znalé uživatele. Při instalaci prostředí Player/Stage si uživatel může vybrat, které ovladače nainstaluje a které ne. Takto lze sestavit prostředí pro běh jediné aplikace, například pro mobilní zařízení. Nevýhodou instalace prostředí MRDS je, že velikost nainstalovaných souborů je téměř schodný s velikostí instalačního souboru (400MB). Naproti tomu instalační soubor Playeru je veliký kolem 3MB a nainstalovaný zabírá pouze o 2MB více. Instalace prostředí Player/Stage se tak hodí na malá a nevýkonná zařízení. Vytvoření malého distribučního balíčku je v MRDS také možné, stále ale vyžaduje OS s podporou .NET nebo .NET Compact Framework. Čas změřený při instalaci nahrává prostředí MRDS, ale není to tak výrazný rozdíl. MRDS kolem čtyř minut a Player/Stage kolem sedmi minut. Player/Stage se instaluje téměř o polovinu času více, ale z uživatelského hlediska se zdá instalace Player/Stage mnohem delší, protože většinu času se zobrazuje kompilace ovladačů a obrazovka je stále stejná.

Název balíčku	**Popis obsahu balíčku**
liblodo0	SLAM (simultaneous localisation and mapping - současné lokalizování a mapování) algoritmus k vytváření odometrické pozice s odchylkou
libplayerc++2	C++ knihovna rozhraní pro Player
libplayerc2	C knihovna rozhraní pro Player
libplayercore2	Knihovna jádra pro překlenutí rozdílů mezi zařízeními podporovanými Playerem a rozhraními
libplayerdrivers2	Knihovna obsahuje ovladače potřebné pro ovládání hardware přes Player
libplayererror2	Knihovna pro práci s chybami
libplayerjpeg2	Knihovna pro práci s jpeg soubory
libplayertcp2	Přesouvá zprávy Playeru do TCP soketů a naopak
libplayerxdr2	Konvertuje strukturu dat Playeru do XDR representace (eXternal Data Representation - mezi platformní reprezentace dat)
libpmap0	Používá filtrování po částech (Monte Carlo metoda) k určení možných map simulačních scén
python-playerc	Vytváření balíků pro jazyk Python
robot-player	Vývojové prostředí Player
Robot-player-doc	Dokumentace pro vývojové prostředí Player (dokumentace)

Tabulka 4.6: Knihovny poskytované po instalaci Player

Kapitola 5

Instalace aktualizací

5.1 Instalace aktualizací MRDS

5.1.1 Distribuce aktualizací

Aktualizace MRDS je nezávislá na Windows Update, proto se aktualizace musí ručně stáhnout a nainstalovat. Instalace je jednoduchá, pouze se spustí aktualizační soubor .exe a aktualizace sama vyhledá složku, kde je MRDS nainstalováno a nainstaluje nové soubory a opravy. Tento způsob aktualizací je možný pouze v plné verzi Academic/Standart. Verze Express aktualizace nepodporuje a je třeba stáhnout celý nový instalační balík a MRDS přeinstalovat. Aktualizace pro MRDS nevycházejí příliš často. Od dubna 2008, kdy byla vydána verze MRDS 2008 do současné doby vyšly tři aktualizace. Před verzí MRDS 2008 Microsoft řešil aktualizace vypuštěním celé nové verze programu, protože MRDS bylo stále ve vývojovém stádiu, bylo pojmenováno odlišně „Robotics Studio 1.x (x je číslo verze)“. Nyní s příchodem stabilní verze MRDS 2008 budou pravděpodobně aktualizace vycházet jako samostatné soubory. Možná se aktualizace integrují i do Windows Update a budou se instalovat automaticky. Názvy aktualizací jsou CTP (Community Technical Preview - Komunitní technický přehled) a název měsíce, kdy aktualizace vyšla. Přehled změn v jednotlivých verzích a aktualizacích je dostupný na webové stránce [9].

5.1.2 Problém s kompatibilitou starších verzí MRDS

Některé projekty vytvořené ve starších verzích MRDS mají problém s kompatibilitou nové verze MRDS. To znamená, že starší projekty nejdou v nové verzi MRDS načíst. I přesto, že Microsoft vydal nástroj na převod starších projektů do MRDS 2008, některé

projekty nelze převést a k jejich editaci je nutné používat starší verzi MRDS. Převážná většina těchto problémů se vyskytuje u převodu MRDS 1.5('Refresh') do verze MRDS 2008. Vzhledem k tomu, že starší projekty není možné vyvíjet v nejnovější verzi MRDS, je možné nainstalovat do operačního systému více různých verzí MRDS.

5.1.3 Vydané verze MRDS

Seznam vydaných verzí MRDS je uveden v tabulce 5.1.

Název verze	**Datum vydání**	**Popis změn**
Robotics Studio 1.0	2006 prosinec	první verze Robotics Studia
Robotics Studio 1.5	2007 květen	vylepšení kompilace v prostředí VPL, nové návody, oprava chyb DSS, úprava dokumentace
Robotics Studio 1.5 'Refresh'	2007 prosinec	aktualizace dokumentace, knihoven CCR a DSS
Robotics Developer Studio 2008	2008 duben	první verze MRDS 2008
Robotics Developer Studio 2008	2008 červenec	přidání nových částí dokumentace, podpora 64 bitových OS, oprava chyb CCR a DSS
MRDS update Collada	2009 leden	možnost importovat do simulačního prostředí entity ve formátu Collada
Robotics Developer Studio 2008 R2	2009 červen	přidání nástroje DSS Log Analyzer pro analýzu logovacích souborů, oprava chyb dokumentace a VSE

Tabulka 5.1: Přehled verzí MRDS

5.2 Instalace aktualizací Player a Stage

5.2.1 Možnosti aktualizací

Možnost aktualizace prostředí Player je pouze kompletním přeinstalováním celého prostředí novou verzí. Na serveru Player jsou uvedeny téměř všechny vydané verze a je možné si je všechny stáhnout. Tato možnost je proto, že když vyjde nová verze Playeru s překompilovaným jádrem, nemusí na nové verzi fungovat starší projekty. Player/Stage podporuje i instalaci více různých verzí na jeden počítač.

5.2.2 Přehled verzí Player a Stage

V tabulce 5.2 je přehled verzí, které se dají stáhnout ze serveru SourgeForge.net [10]. První verze Playeru, která je na serveru umístěna, je verze 1.4rc1 a byla na server umístěna 15. července 2003 a první umístěná verze Stage je 1.3.3 z 7. prosince 2003. Podle data umístění na server je frekvence vydávání nových verzí průměrně tři až čtyři verze za rok pro obě prostředí.

5.3 Porovnání aktualizací Playeru a MRDS

Ze seznamu vydávaných verzí Playeru v tabulce 5.2 vyplývá velká zkušenost vývojového týmu *The Player Project*. Nové verze Playeru vychází čtyřikrát v roce a vývojáři tedy mají možnost odstranit chyby v nové verzi častěji. Nevýhodou aktualizace Playeru je, že se musí nainstalovat celá nová verze. Podle historie MRDS vychází nová verze dvakrát za rok, z čehož vyplývá, že každá nová verze MRDS je velmi dobře připravena a neobsahuje zásadní chyby. Výhodou MRDS je, že se aktualizace řeší vydáním aktualizačního balíku, který opraví pouze chybné soubory a je v řádek několika MB. Když porovnáme velikost instalačního a aktualizačního souboru je aktualizační soubor relativně malý. Zásadní aktualizace MRDS vychází jako celý nový program a to v periodě jednoho roku. Obě vývojová prostředí podporují více nainstalovaných verzí na jednom počítači.

Název verze	**Datum vydání**		**Název verze**	**Datum vydání**
Player 2.1.2	2009-01-15 19:11		Stage 2.1.1	2009-01-16 06:55
2.1.1	2008-06-16 03:39		3.0.1	2008-07-29 18:25
2.1.0	2008-06-14 03:14		3.0.0	2008-07-13 06:00
2.1.0rc2	2008-04-16 07:24		2.1.0	2008-06-14 03:17
2.1.0rc1	2007-12-13 23:00		2.1.0rc2	2008-04-16 06:52
2.0.5	2007-12-13 22:56		2.1.0rc1	2007-12-13 23:01
2.0.4	2007-05-02 22:44		2.0.4	2007-12-13 22:58
2.0.3	2006-09-26 16:13		2.0.3	2006-09-26 16:19
2.0.2	2006-06-09 20:09		2.0.2	2006-06-09 20:10
2.0.1	2006-03-25 01:34		2.0.1	2006-03-25 01:54
2.0.0	2006-02-28 07:09		2.0.0	2006-02-28 07:14
2.0-pre7	2005-10-24 23:41		2.0.0-d	2005-10-24 23:43
2.0-pre6	2005-10-21 17:00		2.0.0a	2005-08-11 20:35
1.6.5	2005-07-29 00:51		1.6.2	2005-02-09 07:10
1.6.4	2005-05-14 20:21		1.6.1	2005-01-02 07:53
1.6.3	2005-05-03 21:02		1.6	2004-12-30 07:32
1.6.2	2005-01-31 00:58		1.3.5	2004-11-12 17:13
1.6.1	2005-01-25 02:07		1.3.4	2004-06-12 07:00
1.6	2004-11-12 09:47		1.3.3	2003-12-07 08:00
1.5	2004-05-31 07:00			
1.4rc2	2003-12-07 08:00			
1.4rc1	2003-07-15 07:00			

Tabulka 5.2: Přehled verzí Player

Kapitola 6

Podpora pro uživatele

6.1 Podpora pro uživatele MRDS

6.1.1 Podpora od Microsoftu

Podpora od Microsoftu je převážně v anglickém jazyce. Microsoft vydává s každou verzí MRDS aktualizovanou nápovědu. Formát nápovědy je ve dvou vyhotoveních. Jako webová stránka na serveru Microsoft [11] a jako formát nápovědy Windows, který je distribuován s instalačním souborem MRDS. Nápověda MRDS je založena na popisu částí MRDS a velkém množství návodů a příkladů k sestavení robotických aplikací. Návody na robotické aplikace jsou koncipovány od jednoduchých až po složité. Ve většině případů je návod podrobný. Návody a nápovědy pro jazyk C# převažují ve všech částech nápovědy.

Pro zákazníky firmy Microsoft je několik zákaznických linek. Jsou zaměřeny obecně na všechny produkty Microsoft a podpora na nich je téměř vždy placená.

Pokud uživatel nalezne problém, který nelze vyřešit s pomocí nápovědy, je možné získávat informace a rady od ostatních uživatelů na diskusních fórech zaměřených na problematiku MRDS.

- *Microsoft Robotics - Community*
  Hlavní fórum zaměřené na MRDS
  `http://social.msdn.microsoft.com/Forums/en-US/roboticscommunity/threads/`

- *Microsoft Robotics - Concurrency and Coordination Runtime (CCR)*
  Fórum zaměřené na problémy kolem CCR

`http://social.msdn.microsoft.com/Forums/en-US/roboticsccr/threads/`

- *Microsoft Robotics - Decentralized Software Services (DSS)*
  Fórum zaměřené na problémy kolem DSS
  `http://social.msdn.microsoft.com/Forums/en-US/roboticsdss/threads/`

- *Microsoft Robotics - Simulation*
  Fórum zaměřené na problémy se simulačním prostředí VSE
  `http://social.msdn.microsoft.com/Forums/en-US/roboticssimulation/threads/`

- *Microsoft Robotics - Visual Programming Language*
  Fórum zaměřené na problematiku programovacího prostředí VPL
  `http://social.msdn.microsoft.com/Forums/en-US/roboticsvisualprogramminglanguage/threads/`

- *Microsoft Robotics - Hardware Configuration and Troubleshooting*
  Fórum zaměřené na problematiku konfigurace hardware
  `http://social.msdn.microsoft.com/Forums/en-US/roboticstroubleshooting/threads/`

- *Microsoft Robotics - All Forums -*
  Seznam všech diskusních fór s problematikou MRDS
  `http://social.msdn.microsoft.com/Forums/en-US/category/robotics/`

6.1.2 Podpora od komunitních skupin

MRDS se od svého vzniku hodně rozšířilo. Microsoft se snaží svůj produkt masivně propagovat. Především v posledních letech do propagace vložilo velké množství peněz. Microsoft zapojuje do projektu firmy vyrábějící robotický hardware a vyvíjí pro tento hardware rozhraní. V podporovaném hardwaru se vyskytuje především robotický hardware, který se sériově vyrábí a je dostupný na trhu v řádech tisíců korun a je tedy dostupný i pro běžné uživatele. Jedním z důležitých hledisek šíření MRDS je i to, že Microsoft Windows stále drží většinový podíl na trhu s operačními systémy. Navíc je MRDS určeno i pro nezkušené uživatele a je tedy vhodné pro proniknutí do robotické problematiky. Díky těmto krokům Microsoftu se MRDS rychle rozšiřuje mezi uživateli.

Z uvedených důvodů je možné na internetu najít několik diskusních fór o problematice MRDS.

- *MSRS Community in Turkey*
  Turecké fórum zabývající se několika robotickými projekty vytvářených v MRDS. Nejznámější projekt je bezpilotní vzdušné vozidlo
  `http://roboticsnedir.com/`

- *Norwegian .NET User Group*
  Norská skupina, která se soustředí kolem platformy .NET
  `http://www.nnug.no/`

- *Conscious-Robots.com pages for MSRS*
  Skupina zaměřující se na autonomní roboty, MRDS, kódy v C# a roboty platformy Pioneer 3 DX
  `http://www.conscious-robots.com/en/robotics-studio/index.php`

- *Seattle Robotics Society*
  Bezpříspěvková organizace The Seattle Robotics Society založená v roce 1982 zabývající se vývojem robotických aplikací. Skupina se skládá z amatérů, studentů a vysokoškolských profesorů a inženýrů
  `http://www.seattlerobotics.org/`

6.1.3 Blogy vývojářů MRDS

Dalším zdrojem informací jsou blogy vývojářů MRDS. Jedna z výhod je, že jsou zde aktuální informace.

- *Microsoft Robotics Developer Studio Blog*
  Hlavní blog Microsoft Robotics Developer Studia
  `http://blogs.msdn.com/msroboticsstudio/default.aspx`

- *Robert Burke*
  Weblog Microsoft Robotics Developer Studio
  `http://blogs.msdn.com/robburke/search.aspx?q=robotics&p=1`

- *Walter Stiers*
  Vztahy Microsoftu v akademické robotické sféře
  `http://blogs.msdn.com/walterst/archive/tags/Robotics/default.aspx`

- *Chewy Chong*
  Skupina robotických nadšenců v Singapuru
  `http://blogs.msdn.com/chewyc/default.aspx`

- *Chris Kilner*
  Experimenty s algoritmy SLAM, MSRS, robotickým viděním
  `http://dotnetrobot.blogspot.com/`

- *Martin R. Calsyn* Dobrodružství v Robotice
  `http://robotsoftware.blogspot.com/`

6.2 Podpora pro uživatele Player/Stage

Dokumentace i podpora Playeru je převážně v anglickém jazyce. Na webové stránce [12] je dostupná dokumentace pro nejnovější verze i některé dokumentace pro starší verze Playeru. Dokumentace je ve formátu HTML stránky a je rozdělena na čtyři části.

- Popis projektu Player
- Podpora pro uživatele, rady jak nainstalovat Player, podporovaná zařízení, návody a další podpora
- Detailní popis všech knihoven implementovaných do prostředí Player
- Podpora uživatelům, stažení Player, Stage, Gazebo, Mezzanine. Odkazy na rozpracované projekty, systém pro nahlašovaní chyb a nápověda.

Jednou z největších výhod projektu Player je, že zdrojové kódy jsou šířeny pod GNU licencí a proto jsou volně dostupné. Pokud se při používání Playeru vyskytují chyby, je možné je nahlásit pomocí systému hlášení chyb na webu Playeru [13]. Stejně tak pokud máte nápad, co do Playeru přidat.

Jako další zdroj informací je k projektu Player připojena mailová diskuse, kde se umísťují příspěvky stejně jako do webového fóra a je tak možné získat odpovědi na otázky. Mailová diskuse je rozdělena na čtyři stupně příspěvků

- playerstage-commit - záznamy nahrávání souborů od vývojářů přes verzovací systémy

- playerstage-developers - diskuse vývojářů
- playerstage-gazebo - diskuse vývojářů Gazebo
- playerstage-users - diskuse o stabilních verzích Player, hlavní diskuse

Další podpora uživatelům je na různých diskusních fórech, která se zaměřují na robotiku, Linux, nebo vývoj software. Jedny z nejrozsáhlejších jsou fóra Nabble [14]. Nejvhodnější je použít vyhledávač Google [15] a hledat uživatele s podobnou chybou nebo její řešení.

6.3 Srovnání podpory uživatelů Playeru a MRDS

MRDS se rychle rozšiřuje mezi uživateli, především díky jednoduchosti vyvíjení aplikací. Proto je pro MRDS velmi rozsáhlá komunita uživatelů, i přes to, že je MRDS vyvíjeno pouze 3 roky. Vzhledem k tomu, že je MRDS nabízeno v placené verzi, musí firma Microsoft poskytovat podporu pro uživatele a někdy se postřehy z podpory uživatelů dostanou i mimo placenou sekci. Protože se MRDS stále vyvíjí, prezentují vývojáři nové poznatky na svém blogu a tak se tyto informace šíří mezi uživateli. Poznatky se dále rozvíjí a vývojáři je používají do nových verzí MRDS. Tímto způsobem si Microsoft zajišťuje rozsáhlou základnu uživatelů a budují tak silné vývojové prostředí, protože při vývoji se vždy vyskytnou problémy a když se dá rychle sehnat odpověď zvyšuje to oblibu programu.

Diskusních fór pro Player je velmi málo. Prakticky se dá podpora získat z mailové diskuse Playeru a nebo na serveru Nabble [14]. Na jiných místech jsem se odezvy na řešení problému nedočkal. I když jsem se odpovědi dočkal, první odezva přišla až po uplynutí měsíce od vznesení dotazu. Protože to bylo v době, kdy jsem s Player/Stage začínal, nevěděl jsem jak postoupit dále a prostředí Player/Stage jsem na dlouhý čas odložil, než jsem se v programování zdokonalil a na řešení problému jsem přišel.

Kapitola 7

Srozumitelnost dokumentace

7.1 Srozumitelnost dokumentace MRDS

7.1.1 Druhy manuálů a návodů

Manuály pro MRDS jsou ve většině případech napsané v elektronické podobě. Jedny z mála výjimek jsou knihy *Professional Microsoft Robotics Developer Studio* od autorů Kyle Johns a Trevor Taylor [16] a kniha *Programming Microsoft Robotics Studio* od Sara Morgan [17]. Většina manuálu a návodů pro MRDS je napsaná v anglickém jazyce. V českém jazyce je napsáno několik článků, které obecně popisují MRDS, případně uvádějí příklady kódu. Mnoho návodů pro MRDS je často spojováno s robotem LEGO NTX 2.1, jako v případě článku pana Petříčka [18].

7.1.2 Kvalita dokumentace

Soubory dokumentace jsou umístěny na webovém serveru MSDN [11], nebo po instalaci MRDS ve složce *documentation* v adresáři MRDS. Nápověda ve složce MRDS je v klasickém souboru nápovědy Windows (soubor .chm) s rejstříkem a vyhledáváním. Nápověda je rozdělena na části pro MRDS, CCR, DSS, VPL a VSE.

Dokumentace MRDS je jedna z mála částí, která není tak kvalitně zpracována. Projevují se v ní ambice firmy Microsoft chválit svůj produkt, tak jako je tomu i u ostatních produktů. Například v popisech služeb CCR a DSS je až příliš vět, které vyzdvihují kvalitu služby. Díky těmto větám se pak v textu ztrácí a těžko se chápe vlastní princip služby.

V dokumentaci je u významných služeb uveden příklad, jak se služba používá. Tyto

příklady jsou naopak zpracovány kvalitně krok po kroku a doprovázeny obrázky, takže i pro začátečníka jsou snadno pochopitelné.

7.2 Srozumitelnost dokumentace Player/Stage

Dokumentace projektu Player/Stage je na vysoké úrovni stejně jako většina projektů pod licencí GNU. Dokumentace je v anglickém jazyce a česky se nedá najít prakticky žádný příspěvek. V dokumentaci jsou detailně popsané všechny knihovny a jejich služby. Ovladače pro podporované robotické aplikace mají v dokumentaci připravené příklady pro snazší pochopení jejich používání. Nechybí návody, které pomohou začátečníkům s instalací, nebo vytvořením prvních aplikací. Dobrý dojem z dokumentace kazí některé chybějící stránky především v návodech. Pro simulační prostředí Stage jsou připravené demonstrační aplikace. Tyto aplikace pomáhají pochopit jak Player a Stage fungují. Na webovém serveru Playeru [7] je dokumentace v několika verzích. Poslední nejnovější verze a několik starších verzí Playeru.

7.3 Srovnání dokumentace Playeru a MRDS

V nápovědě pro prostředí Player/Stage je velmi dobře zpracován popis knihoven, jejich služeb, popis ovladačů a příklady jejich použití. V dokumentaci pro prostředí Player/Stage občas chybí stránky. Je to v části dokumentace, kde jsou návody, jak implementovat aplikace. Microsoft vydává několikrát za rok aktualizovanou verzi nápovědy pro prostředí MRDS. Především do nápovědy přidávají příklady a průvodce vytváření nových aplikací. Vývojáři MRDS také vydávají knihy, ve kterých popisují prostředí MRDS a přidávají návody, jak programovat v MRDS. Díky rychle se rozšiřující komunitě uživatelů rychle přibývá především internetových stránek s návody, jak vyvíjet aplikace v prostředí MRDS.

V prostředí MRDS je možné vyvíjet aplikace několika možnými způsoby. Nápověda MRDS se snaží popisovat všechny možnosti programování. Potom se nápověda stává trochu nepřehledná a postupy k vytváření aplikací se prolínají dohromady. Takto může nápověda uživatele spíše zmást než mu pomoci. Naproti tomu v Player/Stage je jen jeden přístup k programování a tak je pochopení dokumentace mnohem snazší.

Kapitola 8

Podporované robotické platformy

8.1 Podporované robotické platformy MRDS

Pokud má být nástroj pro vývoj robotických aplikací úspěšný a používaný, je třeba, aby podporoval spousty různých platforem. Vzhledem k rychlosti postupu vývoje elektronických a robotických systémů musí programátoři do vývojového nástroje implementovat podporu pro velké množství robotických systémů. Navíc je jen málo robotických systémů, které mají jednotné série. Při vývoji robotických systémů v podnicích, nebo na akademické půdě, jsou používané různé specifické součásti a nespočet variabilit. Nejdůležitější požadavek pro úspěšný vývojový nástroj pro robotické aplikace je tedy jednoduchost implementace nového vlastního popisu pro robotický systém. Každý výrobce do vývojového nástroje implementuje podporu sériově vyráběného robota. Nejen proto, aby se mohl vývojový nástroj představit jednoduchým spuštěním předem připravené aplikace, ale implementovaná podpora může být použita pro rychlý vývoj a tím přispět k oblíbenosti vývojového prostředí.

Microsoft do prostředí MRDS implementoval podporu pro několik sériově vyráběných robotických systémů. Prostředí MRDS podporuje výrobky od více než 52 firem, se kterými Microsoft spolupracuje na vývoji prostředí MRDS. Přímo po instalaci je do prostředí MRDS vložena podpora pro robotické systémy několika firem.

8.1.1 Fischertechnik

Fischertechnik vyrábí robotické hračky, které se skládají z kostiček. Základem je mikroprocesorová deska umístěná v „základní kostce“, která se dá zabudovat na nějaké

místo do stavebnice a naprogramovat. Fotografie několika stavebnic Fischertechnik je na obrázku 8.1. Velikost sestavené aplikace se velmi liší díky variabilitě stavebnic. Více o stavebnici lze nalézt na webové stránce výrobce [19].

V prostředí MRDS je zabudovaná podpora pro RF(Radio Frequency) komunikaci se základní "kostkou"a vytvořené manifesty pro základní kostku, enkodér, kontaktní spínač a motor.

Běžná cena zařízení je kolem 6 000 Kč.

Obrázek 8.1: Pohled na některé stavebnice Fischertechnik

8.1.2 iRobot *Create* a *Roomba*

Firma iRobot vyrábí roboty pro domácnosti. Roboty jsou však oblíbené i jako levné robotické systémy, ve kterou se můžou roboti Create a Roomba přeměnit různými doplňky. Na obrázku 8.2 je iRobot Roomba v podobě v jaké se prodává, to znamená automatický vysavač. Velikost iRobota Roomba je kružnice s průměrem 34 cm a výškou 9 cm. Více podrobností o robotech iRobot lze nalézt na webové stránce [20].

Prostředí MRDS poskytuje pro roboty Create a Roomba manifest pro hlavní modul, komunikaci, řízení motoru, doplňky, senzory a skriptování.

Běžná cena zařízení je kolem 10 000 Kč.

8.1.3 Lego Mindstorms NXT

Stavebnice Lego jsou známé po celém světě. Množství různých variací kostek se dá kombinovat do nekonečných tvarů a proto vznikl NXT a jeho variace. Je to kostka s integrovaným displayem a 32bitovým mikroprocesorem. Pohyb mohou zajistit 3 přídavné

Obrázek 8.2: iRobot Create v konfiguraci automatického vysavače

motory. Vnímání okolí zajistí senzor světla a barvy, zvuku, dotykový senzor a ultrazvukový sonar. NXT je velmi oblíbený u lidí začínajících s robotikou, protože se do NXT lehce naprogramuje jakýkoliv program a pořizovací cena není příliš vysoká. Velikost robota záleží na tom, jak je robot sestaven z Lega. Velikost základní „kostky“ je, výška 14 cm a šířka 6 cm. Na fotografii 2.1 je zobrazen NXT v konfiguraci Tribot, pro který jsou použity všechny motory a čidla, které NXT běžně nabízí. Podrobnosti o robotech Lego NXT a dalších senzorech můžeme nalézt na webové stránce [21].

Z uvedených skutečností se dá uhodnout, proč je v MRDS vypracovaná velice silná podpora právě pro NXT. Jedná se o spoustu návodů a průvodců, jak aplikace s NXT programovat. Manifesty jsou pro všechny prodávané části NXT. Microsoft vždy sázel na co nejjednodušší uživatelské prostředí a když se použije Visual Programming Language, je to společně s NXT ideální a jednoduché vývojové prostředí pro robotické aplikace.

Běžná cena zařízení je kolem 7 000 Kč.

8.1.4 Kondo KHR-1

Kondo KHR-1 je autonomní humanoid Japonské výroby. Robot je 34 cm vysoký se 17 servomotory s nastavením otáčení po jednom stupni. Tyto motory zajišťují široký rozsah pohybů, dokonce i rychlý kung-fu styl pohybu. Robota vyrábí firma Kondo, která se zabývá výrobou malých robotů, stejného systému jako je KHR-1. Na fotografii 8.3 je zobrazen KHR-1. Je to klasická verze humanoida a ostatní typy robotů, které firma Kondo prodává, mají podobný vzhled, jako KHR-1. Další podrobnosti o robotech Kondo jsou na webové stránce firmy [22].

V MRDS je implementované rozhraní pro KHR-1. Toto rozhraní poskytuje řízení servomotorů, zjištění pozice natočení servomotoru, uspání robota pro šetření baterie

a opětovné probuzení.

Běžná cena zařízení je kolem 20 000 Kč.

Obrázek 8.3: Robot KHR-1

8.1.5 MobileRobots Pioneer P3DX

Pioneer P3DX je robot pro výzkumné účely, který má v sobě instalovanou počítačovou desku, baterie, několik senzorů pro měření vzdálenosti. Komunikace probíhá přes WiFi a pohyb zajišťuje dvojice kol s enkodérem a jedno malé kolečko pro zatáčení a stabilitu. Obsahuje i sloty pro rozšíření o další počítačové komponenty. Velikost robota je přibližně 50 cm a podle toho jaké senzory robot nese se může velikost měnit. Fotografie robota P3DX je na obrázku 8.4. Zde je P3DX v klasické konfiguraci a na sobě má umístěné reproduktory, laserový měřič vzdálenosti a na vrchu je instalovaná kamera. Roboty Pionner vyrábí firma MobileRobots. Další podrobnosti jsou na webové stránce [23].

Prostředí MRDS podporuje komunikaci s Pioneerem P3DX ve všech směrech a dovoluje i nainstalovat MRDS přímo do integrovaného počítače Pioneeru P3DX. V prostředí MRDS je několik manifestů pro připojení obecných zařízení k robotovi P3DX. Jde o různé kamery, čidla světla zvuku a další množství zařízení. Velmi dobře je připravena i simulace P3DX v simulačním prostředí VSE a to včetně laserového měřiče vzdálenosti.

Běžná cena zařízení je kolem 80 000 Kč.

Obrázek 8.4: MobileRobots Pioneer P3DX

8.2 Podporované robotické platformy Player/Stage

Výrobce	**Robot**	**Ovladač**
Acroname	Garcia	garcia
Botrics	Obot d100	obot
Evolution Robotics	ER1 and ERSDK robots	er1
iRobot	Roomba vaccuming robot	roomba
K-Team	Robotics Extension Board (REB)	reb
K-Team	Khephera	khepera
Mobilní roboti	PSOS/P2OS/AROS-based, Pioneer, AmigoBot	p2os
Nomadics	NOMAD200 a podobní roboti	nomad
RWI/iRobot	RFLEX-based a příslušenství	rflex
Segway	Robotic Mobility Platform (RMP)	segwayrmp
UPenn GRASP	Clodbuster	clodbuster
Videre Design	ERRATIC mobile robot platform	erratic
White Box Robotics	914 PC-BOT	wbr914

Tabulka 8.1: Přehled podporovaných robotů projektem Player

Při instalaci Playeru si uživatel může vybrat, které ovladače chce nainstalovat. Seznam podporovaných robotických platforem je uveden v tabulce 8.1 a podrobný přehled je v dokumentaci Player/Stage na webu [7].

Prostředí Player/Stage podporuje velké množství hardwaru. Seznam hardwarových součástí je obsáhlý a proto je vhodné uvést pouze základní platformy. Díky licenci GNU je možné najít na internetových vyhledávačích ovladače pro velké množství hardware.

8.3 Porovnání podporovaných robotických platforem Playeru a MRDS

Větší množství podporovaného hardware v prostředí Player/Stage vyplývá z delší tradice prostředí Player/Stage. Za dobu vývoje Player/Stage, více než šest let, byli napsané ovladače pro různý hardware. Díky tomu, že jsou ovladače v Linuxu ve velké míře kompatibilní s různým hardware, se počet podporovaných robotických platforem rychle rozrůstá. Dlouhá tradice projektu Player/Stage jej masivně rozšířila do akademické sféry, protože před příchodem prostředí MRDS nebyl k dostání tak rozsáhlý program na řízení robotů.

Microsoft podporuje šíření prostředí MRDS i mezi uživateli, kteří nemají zkušenosti s programováním robotických aplikací. v prostředí MRDS jsou podporované robotické platformy, které nejsou cenově příliš náročné jako je Fischertechnik nebo Lego NXT. Podpora je i pro dražšího robota Pioneer P3DX, který se ve velké míře používá v akademickém výzkumu, kde se prostředí MRDS také snaží prosadit.

Kapitola 9

Přenositelnost vytvořené aplikace

9.1 Přenositelnost vytvořené aplikace MRDS

MRDS je založeno na platformě .NET. Ze všech projektů vyvíjených v prostředí MRDS je možné vytvořit program, který bude pracovat na systémech podporujících platformu .NET. V současné době jsou to všechny operační systémy firmy Microsoft, pro kterou je .NET stěžejní vývojová platforma. Jsou to systémy klasických operačních systému Windows od verze XP, přes Vista a přicházející Windows7. Dále je tu podpora pro mobilní operační systémy Windows Mobile, které obsahují rozhraní .NET Compact Framework.

Když máme sestavenou robotickou aplikaci v prostředí MRDS jsou pro její funkci potřeba robotické knihovny MRDS. Pomocí nástroje *DssDeploy.exe* se může vytvořit autonomní balíček, do kterého se vloží všechny potřebné knihovny z prostředí MRDS. Poté můžeme aplikaci spustit i na počítači kde MRDS není nainstalováno. Stále je ale vyžadována podpora platformy .NET.

9.2 Přenositelnost vytvořené aplikace Player/Stage

Díky nenáročnosti konzolových systémů Linux je možné nasazení aplikací Player/Stage na i na méně výkonných výpočetních systémech jako jsou telefony, autonomní počítačové desky a jednoduché výpočetní jednotky s operačním systémem Linux. Protože prostředí Player/Stage podporuje i jiné platformy než Linux, je možné vytvořenou aplikaci spouštět i na dalších operačních systémech, Solaris, Mac OS X, BSD. Tam kde chceme používat

aplikace vytvořené v prostředí Player/Stage musí být toto prostředí nainstalováno. Díky velké variabilitě instalace prostředí Player/Stage je místo zabrané instalací pouze několik MB. Klient Playeru se dá v určitých případech používat i na systému Windows a to znamená, že server Playeru a klient mohou být spuštěny na rozdílným operačních systémech.

9.3 Porovnání přenositelnosti aplikací Playeru a MRDS

Velká výhoda prostředí Player/Stage je to, že podporuje několik rozdílných operačních systémů a většina z nich je dostupná pod licencí GNU. Naproti tomu firma Microsoft se zabývá vývojem operačního systému Windows pro téměř libovolnou výpočetní jednotku a vyvinutá aplikace by měla být mezi všemi operačními systémy Windows kompatibilní.

Tato výhoda může trochu přejít do pozadí, protože systémy Windows jsou dostupné pouze v placených verzích. I když je získání operačního systému podloženo financemi, zase za to zákazník dostane podporu od firmy Microsoft. Proto je otázka, kterou cestou se vydat. Když používáme robotické prostředí pro akademický vývoj je v současné době většinou možné použít licence MSDNAA na produkty Microsoft a tím se výhoda licence GNU pro Player/Stage stává bezcenná.

Vlastnost prostředí Player/Stage, že klientská a serverová část může být na odlišných operačních systémech se může uplatnit i pro prostředí MRDS. Služby z knihovny DSS, které zajišťují přesuny dat mezi částmi aplikace v MRDS je také založena TCP spojení a proto není problém umístit řídící aplikaci například na linuxový operační systém. Hlavní část MRDS, která sbírá data a ovládá robotické části musí být stále na platformě Windows a .NET.

Kapitola 10

Praktické využití vývojových prostředí

10.1 Projekty vyvíjené v prostředí MRDS

MRDS je prostředí pro vývoj robotických aplikací, které je určené pro pokročilé programátory, ale i začátečníky, protože je zde jednoduchá implementace aplikace přes VPL. I když vývoj prostředí MRDS neprobíhá tak dlouhou dobu již se objevují firmy, které nasazují prostředí MRDS do svého vývoje i pro řízení praktických aplikací. Jedna z prvních firem používající prostředí MRDS byla francouzská firma ROBOSOFT již v roce 2006. První roboti řízení prostředím MRDS byli robuLAB80, robuROC4 a robuROC6. Jsou to roboty pro přemísťování malých objektů a roboty pro vojenské účely. Další informace je možné nalézt na webové stránce firmy ROBOSOFT [24]. Spolupráci na prostředí MRDS rozvíjí Microsoft se společnostmi CoroWave, Fischertechnik, iRobot, KUKA Roboter, Lego, Segway, VIA Technologies a dalšími více než 50 firmami [25].

10.2 Projekty vyvíjené v prostředí Player/Stage

Reálné nasazení prostředí Player/Stage je hlavně ve výzkumném sektoru, kvůli jednoduché upravitelnosti a dostupnosti zdrojových kódu. Výhodou prostředí Player/Stage je i licence GNU a nenáročnost prostředí na výkon hardware. Projekty, které využívají ve vývoji vlastnosti Player/Stage jsou například robotické jednotky White Box Robotics 9, projekt implementace Player/Stage do liveCD Knoppix, projekt robotOS i projekt, který

aplikuje Player/Stage v prostředí Win32 přes systém cygwin a několik dalších projektů. Protože je Player/Stage GNU projekt a nemá silnou propagační kampaň, tak prezentace projektů, které prostředí Player/Stage využívají se omezuje na webové stránky robotických fakult nebo komunit. Z tohoto důvodu jsou projekty s použitím Player/Stage těžko vyhledatelné.

Na webové stránce [26] je abecední seznam některých významných uživatelů Player/Stage. Seznam je rozdělen podle kontinentů a převážná většina uvedených institucí jsou výzkumné ústavy nebo univerzity. Jedna z mála firem v seznamu je třeba firma Robotnik, zde je její webová stránka [27]. Firma se zabývá vývojem a výrobou, robotických ramen, robotických rukou a autonomními robotickými vozidly. Další firmou je White Box Robotics a na jejich webu [28] je popis vývojových prostředí, které používají k vývoji. Prostředí Player/Stage používají pro vývoj robota 914 PC-BOT. Player/Stage používá k vývoji i firma Intel, nebo Boeing, ale není přesně popsáno k jakým účelům. Další firmou, která používá Player/Stage je firma MobileRobots, která vyrábí například roboty Pionner viz. webová stránka výrobce [23]. Firma CoroWave používá Player/Stage pro venkovního robota Explorer, nebo robota s rukou CoroBot. Zde je webová stránka Explorer [29] a zde pro CoroBot [30].

10.3 Porovnání použití

Player/Stage se více uplatňuje na akademické půdě po celém světě. Díky své jednoduchosti, podpoře linuxových systémů, volným kódů a GNU licenci, ale i některé firmy Player/Stage implementovaly do svých výrobních postupů.

Na druhou stranu MRDS se ve větší míře využívá mezi firmami. Používá se mezi začátečníky na robotickém poli, díky jednoduchosti vývoje aplikací. Do akademické sféry se teprve pomalu dostává a konkuruje tak prostředí Player/Stage.

Kapitola 11

Cena prostředí

11.1 Cena prostředí MRDS

MRDS je vydáváno ve třech verzích jejichž rozdíly jsou popsané v kapitole 4.1.1. Verze Express Edition je distribuována zdarma s několika omezeními popsaných v kapitole 4.1.2. Verze Academic Edition je distribuována přes službu poskytování produktů pro akademické použití Microsoft MSDNAA a Microsoft DreamSpark. Proto je pro akademické použití plná verze prostředí MRDS zdarma. Verze Standart Edition je určena pro komerční využití a její cena se pohybuje kolem 499.95 USD (přibližně 10 000 Kč). Pokud si uživatel pořídí placenou verzi prostředí MRDS Standart Edition dostane telefonickou a mailovou podporu k produktu.

11.2 Cena prostředí Player/Stage

Projekt Player je vyvíjen pod licencí GNU a proto je zdarma a jeho kódy jsou volně šiřitelné. Nevýhodou GNU licence je, že nemůže uživatel vyžadovat žádnou podporu. Podpora je ze strany dobrovolníků a lidí, kteří projektu Player věnují svůj volný čas.

11.3 Porovnání ceny prostředí Player/Stage a MRDS

V akademické sféře se dbá na šetření s finančními prostředky a proto se hledal alternativní software, který je zdarma. Tento požadavek velmi kvalitně splňoval operační systém Linux, který přináší alternativu k Windows. Díky tomu, že Player/Stage podporuje Linux a navíc má velkou historii, je v akademické sféře velmi rozšířen. Proto Microsoft zavedla služby MSDNAA, přes kterou jsou do akademické sféry distribuovány produkty Microsoft. Tím se Microsoft snaží si získat budoucí potencionální uživatele z řad studentů. Tímto krokem se prostředí MRDS zařadilo v akademické sféře vedle prostředí Player/Stage. Nejlepší vlastnost Linuxu a Player/Stage, kterou jsou distribuce zdarma, služba MSDNAA narušila. V budoucnu se může stát MRDS na akademické půdě běžným prostředím pro programování robotů. Především i díky tomu, že je určeno i pro uživatele začátečníky, což je jedna z důležitých vlastností pro výuku.

Prostředí Player/Stage tak zůstává výhoda v tom, že se Linux nechá jednoduše instalovat jako velmi nenáročný operační systém a pak můžeme Player/Stage na Linuxu používat na nenáročných výpočetních systémech. MRDS stále potřebuje pro svou funkci platformu .NET a ta je náročnější, než jádro Linuxu.

MRDS se pro komerční použití prodává v ceně 10 000 Kč. Například rýsovací program AutoCad stojí i několik desítek až stovek tisíc. Cena za prostředí MRDS je naopak vykoupena podporou pro uživatele. Když se MRDS používá pro vývoj v nějaké firmě, je nutné aby se případné chyby odstranily co nejrychleji, protože jinak dochází ke ztrátě zisku. Placená verze programu tedy dává jistou záruku, že nedojde dlouhodobějšímu výpadku vývoje.

Kapitola 12

Implementace aplikace

Pro porovnání způsobu a náročnosti implementace projektu v obou prostředích MRDS a Player/Stage byla zvolena aplikace pro řízení pohybu robota v prostředí. Řídící aplikace řídí pohyb robota při projíždění prostředím tak, aby nenarazil do objektů nebo stěn v prostředí a včas se překážkám vyhnul. Aplikaci je možné implementovat na robota, který má schopnost pohybu a musí mít minimálně dva měřiče vzdálenosti, nebo laserový měřič pro snímání prostředí, ve kterém se robot pohybuje.

Aplikace vyhýbání se překážkám funguje tak, že senzory umístěné na robotu pro měření vzdálenosti jsou na levé a pravé straně, tak aby se snímací úhel překrýval před středem robota. Když se robot blíží k překážce a vzdálenost k překážce je menší než *maximální měřená vzdálenost*, dávají senzory na levé a pravé straně robota rozdílné hodnoty vzdálenosti. Když je vzdálenost z jednoho čidla menší než vzdálenost z druhého, robot sníží rychlost tak aby mohl bezpečně zatočit a zatočí na tu stranu, kde je vzdálenost větší a vyhne se tak překážce. Zatočení je omezené hodnotou maximálního zatočení. Na příkladě 12 je uveden pseudokód vyhýbání se překážkám.

```
Pseudokód vyhýbání se překážkám

// nekonečná smyčka
 for(;;)
 {
  // blokování programu dokud nepřijdou nová data (10Hz)
   čekej na data ze senzoru();

   // načtení minimální vzdálenosti z~čidel
    minVpravo = přečti pravou vzdálenost();
    minVlevo = přečti levou vzdálenost();

   // omezení velikosti vzdálenosti
    if (vlevo > maximální měřená vzdálenost)
        vlevo = maximální měřená vzdálenost;
    if (vpravo > maximální měřená vzdálenost)
        vpravo = maximální měřená vzdálenost;

   // výpočet nové rychlosti, tak aby robot zpomalil a stihl se vyhnout
    nová rychlost = (vpravo + vlevo) / hodnota pro úpravu rychlosti;
   //výpočet nové hodnoty
    nové zatočení = (vpravo - vlevo);
   // maximální uhel zatočeni
    nové zatočení = limit(nové zatočení,
                          maximální zatočení vlevo,
                          maximální zatočení vpravo);
   // zapsat příkazy do robota
    nastaveni rychlosti(nová rychlost, nové zatočení);
 }
```

Příklad 12: Pseudokód vyhýbání se překážkám

12.1 Implementace aplikace v MRDS

Tvorba nového projektu v MRDS se může rozdělit na několik části. Tento postup vytvoří zcela prázdny projekt s importem základních knihoven MRDS.

1. Vytvoření nové aplikace nástrojem *DssNewService*
2. Sestavení simulace a vložení entit do simulační scény v prostředí VSE
3. Úprava vytvořeného manifestu pro simulaci a pro robota (pomocí DSSME)
4. Naprogramování a kompilace řídící aplikace pro řízení robota v prostředí MVS
5. Spuštění vytvořené aplikace

12.1.1 Vytvoření nového projektu nástrojem *DssNewService*

Nový projekt se vytvoří příkazem napsaným v DSSCP: *dssnewservice /language: "Csharp" /service:"Vyhybani" /dir:"Vyhybani"*. Možností pro vytváření nových projektů je více a podrobnosti o parametrech lze získat v nápovědě *dssnewservice help*. Pro fungující implementaci stačí popsané nastavení. Projekt musí být umístěn ve složce prostředí MRDS, jinak nebudou fungovat cesty ke knihovnám a funkcím MRDS. Takto sestavíme nový projekt pro MVS s importovanými robotickými knihovnami s vytvořeným manifestem Vyhybani.Manifest.xml, hlavním souborem C# projektu .csproj a dalšími soubory potřebnými pro C# projekt. Projekt hned po vytvoření zkompilujeme, aby jsme ho mohli přidávat do manifestů.

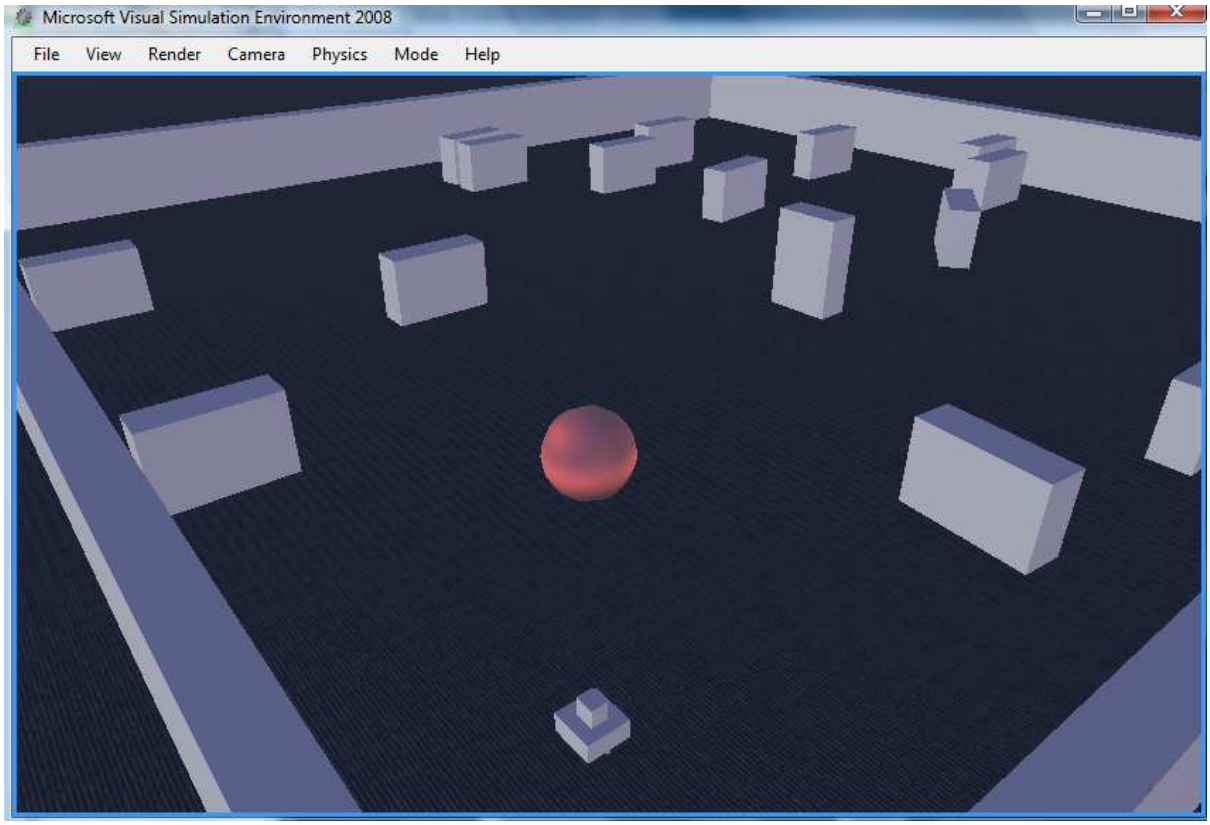
Při vytváření nové aplikace je v nastavení projektu MVS cesta ke spouštěnému manifestu v absolutní podobě, proto při přesunutí projektu do jiné složky projekt nefunguje. Proto při přesunutí projektu do jiné složky je potřeba změnit nastavení projektu v záložce *Debug*. Nastavit je potřeba pracovní adresář *Working directory* a argumenty pro DSSCP *Command line arguments* tak, aby projekt našel soubor manifest. Tímto způsobem je možné k projektu připojit i manifest pro simulaci.

Podle druhu robotické aplikace se do kódu musí přidat reference na knihovnu *RoboticsCommon.proxy.dll*, která obsahuje ovladače pro přístup k obecným (generic) robotickým zařízením, jako je motor, kamera, čidlo, měřič vzdálenosti a další. Protože používáme v aplikaci motor i měřiče vzdálenosti, importujeme tuto knihovny do referencí v MVS. Podrobnosti o knihovně *RoboticsCommon.proxy.dll* lze najít použitím příkazu *dssinfo bin\RoboticsCommon.Proxy.dll /o:adresar /s:html* z příkazové řádky DSSCP.

12.1.2 Sestavení simulační scény

Pro ověření funkčnosti naprogramované aplikace použijeme simulační prostředí VSE. Aby jsme si vytvořili simulační scénu musíme spustit prostředí VSE. Jednoduchá cesta se nabízí spuštěním připraveného prostředí přes zástupce ze složky MRDS v nabídce Start systému Windows. Nejvhodnější je zástupce pro spuštění simulačního prostředí *Simple Simulated Robot*. Tato simulace obsahuje pouze robota s podvozkem a ovládací aplikací, pro vyzkoušení robota. Nyní můžeme začít upravovat prostředí tak, aby vyhovovalo naší požadované simulaci. Nejdříve je třeba přepnout prostředí VSE do editačního módu, přes položku *Mode -> Edit*. V tomto módu můžeme přidávat, editovat a odebírat jednotlivé entity. Přesunování překážek myší provedeme tak, že musíme mít entitu vy-

branou v seznamu entit. Stiskneme klávesu CTRL a pravým tlačítkem myši si vybereme v jakých osách chceme entitu přesouvat a levým tlačítkem entitu přesouváme. Pro aplikaci vyhýbání se překážkám necháme v prostředí robota a přidáme překážky, tak jako na obrázku 12.1. U statických entit nepotřebujeme žádné služby a tak je stačí do scény vložit přes položku *Entity -> Load Entities*.



Obrázek 12.1: Visuální simulační prostředí s entitami v editačním módu

Když omylem smažeme entitu nebe, země nebo slunce je v MRDS nadefinován soubor manifest s jehož pomocí tyto entity importujeme zpět do VSE. Tento manifest se jmenuje *EntityUI*. Importujeme ho do simulačního prostředí VSE přes *File -> Open Manifest* ze složky *MRDS\samples\Config* a tlačítkem na Dialogu *Add default scene* vložíme základní entity, nebe, zemi a slunce. Stejným postupem můžeme k simulačnímu prostředí VSE spustit okno Dashboard pro otestování řízení robota.

Když upravíme pozice entit tak, aby odpovídali našim požadavkům, simulační scénu uložíme přes *File -> Save Scene As* a simulační scénu nazveme *VyhybaniSim*. Takto vygenerujeme dva soubory manifest. Jeden s popisem scény *VyhybaniSim.xml* a druhý, který popisuje potřebné služby, pojmenovaný *Vyhybani.Manifest.xml*. Tento manifest pak používáme při spouštění simulace projektu. Pokud je simulační prostředí spuštěné a po-

kusíme se o spuštění ještě jedné instance, tak DSSCP zobrazí chybové hlášení a prostředí se nespustí.

12.1.3 Vytváření vlastní entity pro VSE

Tato kapitola popisuje postup jak vytvořit jednoduchou entitu, například bludiště, z bitmapového obrázku. Vytvoříme půdorys bludiště v bitmapovém obrázku a převedeme do vektorového formátu například programem *WinTopo* a uložíme ve formátu *DXF* (formát AutoCad Drawing Exchange Format). Pomocí programu *GoogleSketchUp* načteme soubor *DXF*. Vytvoření 3D objektu se provede spojením všech čar a nástrojem Tlačit / táhnout vytvoříme 3D model, který se exportuje do souboru *Google Earth 4 (.kmz)*. Potom musíme soubor .kmz přejmenovat na archiv .zip. Po extrahování archivu zip nalezneme soubor .dae (soubor Collada) který můžeme již přímo importovat do simulačního prostředí VSE přes položku *Entities -> Load entities*.

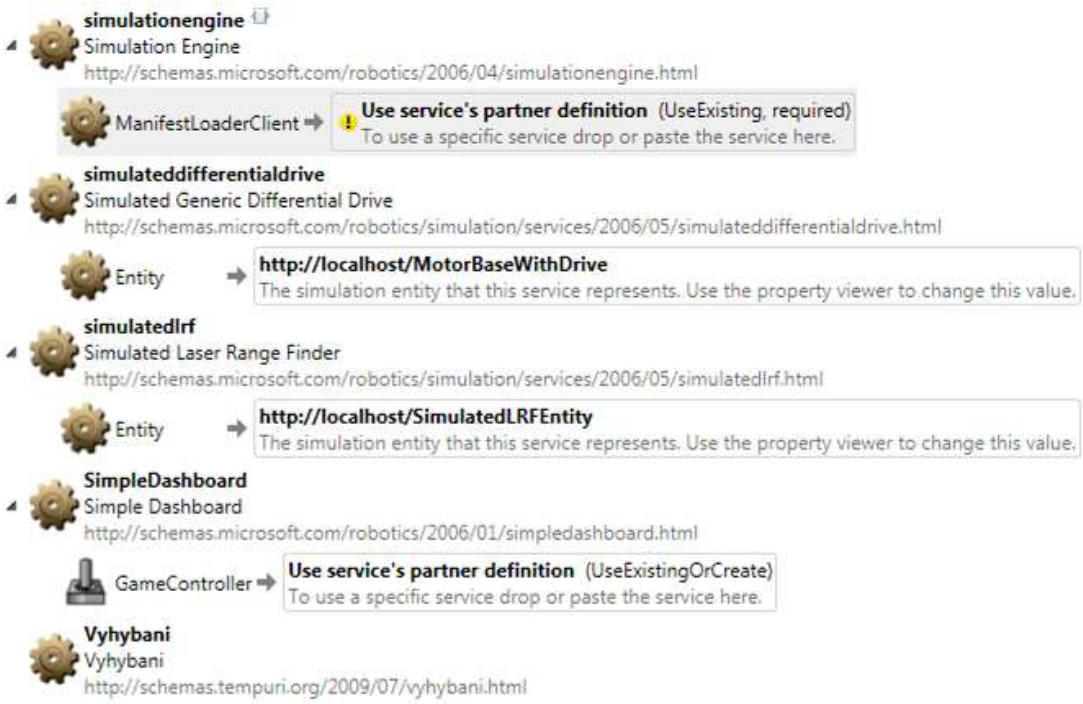
Existují i další postupy pro vytváření entit, například převodem výkresu CAD do souboru .obj nebo souboru .bos. Soubor BOS i OBJ obsahují vzhled robotů pro simulační prostředí. Entity se mohou vytvářet i pomocí animačních nebo rýsovacích programů, vždy je ale nutné soubory převést na formát BOS nebo OBJ.

Na vytváření simulačních scén se specializují i firmy jako je například firma SimplySim [31]. Cena za jednu větší místnost (školní třída) se pohybuje kolem 120 Eur.

12.1.4 Úprava manifestů

Do manifestu vytvořeného při uložení simulační scény se musí přidat manifest vygenerovaný při vytváření nového projektu. To se provede tak, že se manifest *VyhybaniSim.manifest.xml* se načte do DSSME a přidá se do něj služba *Vyhybani*, což je manifest *Vyhybani.manifest.xml*, který v sobě nese odkazy na služby, které budou řídit robota. Pro *simulationengine* musíme přidat počáteční stav simulační scény, který byl vytvořen. Dvakrát klikneme na *simulationengine* a zvolíme *Import initial state*, změníme druh souboru na xml a nalezneme *VyhybaniSim.xml* a načteme jej. Tímto máme spojený manifest *VyhybaniSim.manifest.xml* se simulačním prostředím a stavem scény, tak jak jsme jí sestavili. *VyhybaniSim.manifest.xml* je na obrázku 12.2.

Pro reálný hardware se musí vytvořit nový manifest v DSSME a opět do něj přidat manifest *Vyhybani.manifest.xml*. Pro Lego NX musíme vložit služby pro komunikaci s dvojicí



Obrázek 12.2: Manifest pro komunikaci se simulačním prostředím

motorů pro pohyb *LegoNXTDrivev2* a pro komunikaci se sonarem *LegoNXTUltrasonicSensorv2*. Pro obě služby se musí nastavit počáteční podmínky. Pro službu *LegoNXTDrivev2*, je to vzdálenost mezi koly, porty na řídící „kostce“ NXT ke kterým jsou připojeny motory, průměr kol a frekvence obnovování. Pro službu sonaru *LegoNXTUltrasonicSensorv2* je to port pro připojení k řídící „kostce“ NXT a frekvence čtení dat. Ještě se musí vložit služba řídící „kostky“ NXT, přes kterou robot komunikuje s počítačem. Sestavený manifest je uveden na obrázku 12.3.

Tyto dva manifesty budeme používat při spouštění řídící aplikace přes nástroj *dsshost*.

12.1.5 Implementace robota

Když vyvíjíme novou robotickou aplikaci, je důležité mít ovladače pro hardware. V prostředí MRDS jsou ovladače nahrazeny službami a reference na služby jsou uloženy v souborech manifest. Manifestem může k aplikaci připojit reálný hardware nebo simulační aplikaci. Pokud chceme použít pro stejnou aplikaci jiný hardware, pouze použijeme jiný manifest. Pro jsme si vytvořili dva manifesty. Pokud již jednou zkompilujeme robotickou aplikaci, stačí pro její opětovné spuštění pouze nadefinované manifesty, ale je třeba je spouštět z adresáře MRDS.

Obrázek 12.3: Manifest pro komunikaci s NXT v programu DSSME

Při vývoji řídící aplikace upravujeme pouze soubor *DiagramService.cs*. Soubor musí obsahovat několik bloků naprogramovaného kódu. Většina bloků kódu se přidá automaticky při vytváření nové aplikace. Pro vlastní psaní aplikace je potřeba najít metodu Start() a do této metody postupně psát aplikaci podle kódu přiloženého v příloze B. Jednotlivé části kódu jsou popsané v komentářích kódu. Uvedený kód není kompletní. Kompletní kód je možné naleznout na CD přiloženém k práci.

Při kompilaci se vytvoří knihovny, které obsahují služby pro generovanou aplikaci a uloží se do adresáře *MRDS\bin*. Pokud v prostředí MVS aplikaci změníme je potřeba ji zkompilovat, aby se knihovny přepsali. Generované knihovny mají stejný název jako uložený projekt v prostředí MVS. Když projekt VPL pojmenujeme *Vyhybani* knihovny budou pojmenovány *Vyhybani.Y2009.M07*. Y znamená rok a M měsíc kdy byla aplikace sestavená. Generované knihovny jsou tři a spolu s nimi se generují soubory programové databáze (.pdb - programová databáze) a soubor manifest pro dokumentaci. Soubor PDB udržuje informace o ladění programu a informace o stavu projektu. Dále soubor uchovává seznam všech symbolů použitých v knihovně, jejich cesty, názvy a řádky kde jsou nadefinovány. Tyto informace dovolují inkrementálně spojovat nastavení ladění programu.

12.1.6 Spuštění vytvoření aplikace

Když máme simulační prostředí sestavené, všechny vlastnosti a entity nadefinované v .xml souboru, vlastní spuštění simulace se provede příkazem *dsshost /port:50000 /manifest: "VyhybaniNXT.Manifest.xml"* pro reálnou aplikaci NXT nebo *dsshost /port:50000 /manifest:"VyhybaniSim.Manifest.xml"* pro simulaci v simulačním prostředí VSE z příkazové řádky DSSCP.

12.2 Implementace aplikace v Player/Stage

Vývoj robotických aplikací v prostředí Player/Stage je stejné jako v ostatním linuxovém programování. Všechny konfigurační soubory a soubory aplikace pro řízení robota jsou v textovém formátu a jsou jednoduše upravitelné libovolným textovým editorem. Jediná část, která se kompiluje, je řídící program napsaný v programovacím jazyce.

12.2.1 Vytváření řídícího programu

Řídící aplikace se vyvíjí v programovacím prostředím, které podporuje programovací jazyk, ve kterém je aplikace programována. Implementace vyhýbání se překážkám v prostředí Player/Stage je napsaná v jazyce C++. Pro vývoj řídících aplikací je vhodné například prostředí *KDevelop* nebo *NetBeans*, podle použitého programovacího jazyka. Možnost psát aplikaci je i v klasickém textovém editoru pro Ubuntu. Tento editor se jmenuje Gedit a při vytvoření souboru s příponou programovacího jazyka barevně zvýrazňuje syntaxi kódu. V Geditu jsou podporované například jazyky C, C++, Java, HTML, XML, Python, Perl, Tex.

Při psaní řídící aplikace je potřeba importovat do kódu knihovny Playeru a knihovnu pro datovou komunikaci *<libplayerc++playerc++.h>, include <iostream>*. Řídící aplikace je zobrazena na příkladu 12.2.1.

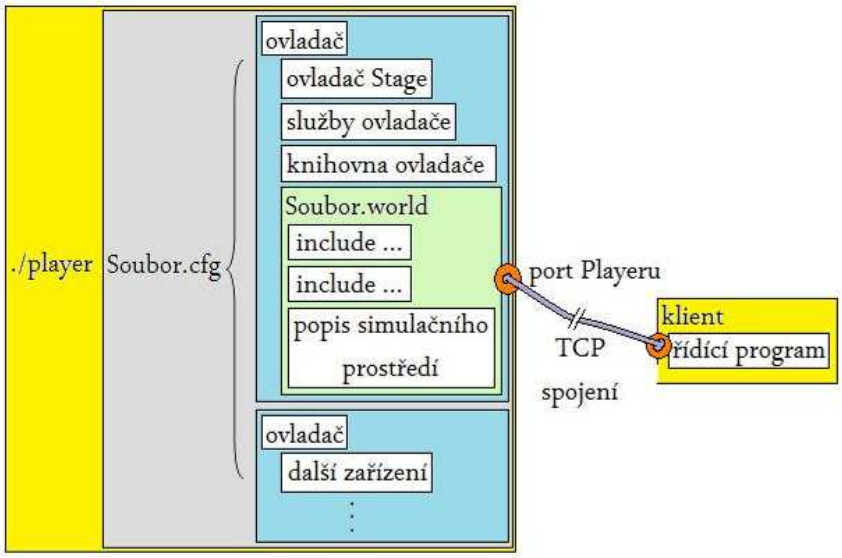
```
Zdrojový kód řídící aplikace "vyhybani.cc"

#include <libplayerc++/playerc++.h> // knihovna pro přístup k Playeru
#include <iostream>     // knihovna poskytující proudový přenos dat
#include "args.h"       // import příkazů pro práci s argumenty

int main(int argc, char **argv) // získá dva argumenty zadané při spuštění
{
  parse_args(argc,argv);        // rozdělení argumentů

  try                           // zahazování výjimek při chybách
  {
    using namespace PlayerCc;   // použije PlayerCc, z knihovny playerc++ pro přístup
                                // klienta k rozhraním hardware
    PlayerClient robot(gHostname, gPort); // připojení klienta k serveru Player
    Position2dProxy pp(&robot, gIndex);   // požadavek pro přístup k pozičnímu hardware
    LaserProxy lp(&robot, gIndex);        // požadavek pro přístup k laserovému měřiči
    std::cout << robot << std::endl;      // výpis připojení k serveru Player
    pp.SetMotorEnable (true);             // spuštění motorů, aby se robot rozjel

    for(;;)                     // nekonečná smyčka
    {
      double newspeed = 0;      // definování hodnoty pro novou rychlost
      double newturnrate = 0;   //definování hodnoty zatočení robota
      robot.Read();             // blokování programu dokud nepřijdou nová data čtení 10Hz
      double minR = lp.GetMinRight(); // získat nejmenší hodnotu z pravého laserového měřiče
      double minL = lp.GetMinLeft();  // získat nejmenší hodnotu z levého laserového měřiče
      std::cout << "minR: " << minR   // výpis minR a minL na terminál
                << "minL: " << minL
                << std::endl;
      double l = (1e5*minR)/500-100;
      double r = (1e5*minL)/500-100;
      if (l > 100) l = 100;           // omezení l na maximální hodnotu 100
      if (r > 100) r = 100;           // omezení r na maximální hodnotu 100
      newspeed = (r+l)/1e3;           // spočtení nové rychlosti
      newturnrate = (r-l);            // spočtení úhlu zatáčení
      newturnrate = limit(newturnrate, -40.0, 40.0);  // maximální hodnota zatočení je +-40°
      newturnrate = dtor(newturnrate);                // převod stupňů na radiány
      std::cout << "speed: " << newspeed    // výpis rychlosti na konzoly
                << "turn: " << newturnrate
                << std::endl;
      pp.SetSpeed(newspeed, newturnrate);   // nastavení rychlosti a zatočení do robota
    }
  }
  catch (PlayerCc::PlayerError e)  // zachytnutí výjimek
  {
    std::cerr << e << std::endl;   // vypsání chyby
    return -1;                     // chybný konec programu
  }
}                                  // konec programu
```

Příklad 12.2.1: Řídící aplikace vyhýbání se překážkám

Když máme napsanou řídící aplikaci, je třeba jí zkompilovat a zkontrolovat práva pro spouštění. Po kompilaci již můžeme soubor spouštět klasicky, pomocí příkazu *./vyhybani* z terminálu. Nejdříve však musíme mít spuštěný server Player, aby se řídící program mohl k serveru Player připojit. Pro testování je vhodné aplikaci spustit v simulačním prostředí Stage. Pro simulaci si musíme připravit konfigurační soubor *vyhybani.cfg*, soubor *vyhybani.world* s popisem simulačního prostředí a simulovaného robota a bitmapový obrázek simulovaného prostředí.

12.2.2 Simulace v prostředí Stage

Na obrázku 12.4 je znázorněná struktura sestavení serveru Player, na kterém běží simulační prostředí Stage a k serveru se připojuje klientská řídící aplikace. Stejně tak jako je ovladač pro reálný hardware, je ovladač i pro Stage.



Obrázek 12.4: Struktura připojení řídící aplikace k simulačnímu prostředí

Pro spuštění simulačního prostředí Stage je potřeba vytvořit soubor konfigurace, ve kterém budou reference na ovladač Stage. Příklad 12.2.2 představuje obsah souboru *vyhybani.cfg* s implementací ovladače pro Stage a rozhraním pro měření pozice a laserové měření polohy.

```
Obsah konfiguračního souboru vyhybani.cfg

driver   #načte ovladač simulačního pluginu Stage
(
  name "stage"               #jméno ovladače Stage
  provides ["simulation:0"]  #poskytování simulace
  plugin "libstageplugin"    #načtení knihovny pluginu Stage
  worldfile "simulace.world" #načte soubor popisující simulační scénu
)

driver           #vytvoří ovladač stage a připojí position2d a laser
(                #rozhraní k modelu robota"garcia"
  name "stage"   #název ovladače
                 #poskytuje sonar, poziční a laserový senzor
  provides ["position2d:0" "laser:0" "sonar:0"]
  model "garcia" #název modelu robota
)
```

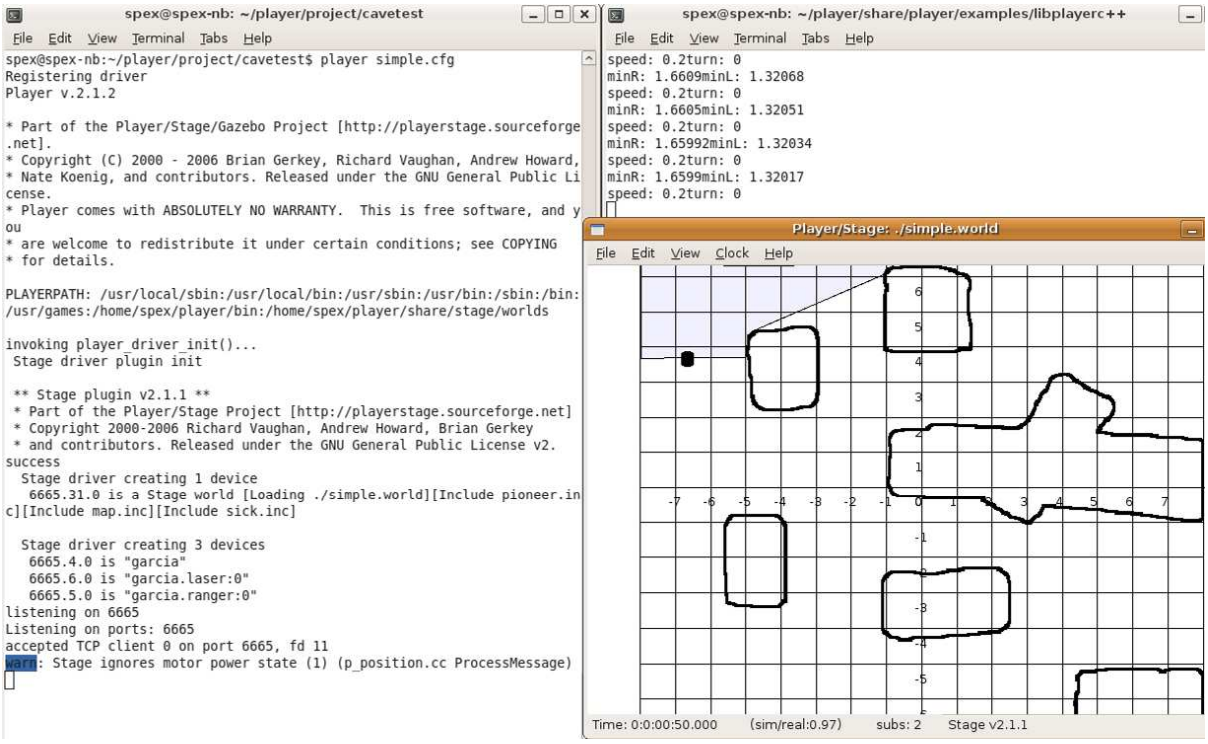
Příklad 12.2.2: Konfigurační soubor pro simulaci v prostředí Stage

Konfigurační soubor z příkladu 12.2.2 zobrazuje i referenci na soubor *vyhybani.world*. Soubor *vyhybani.world* obsahuje definici simulační scény. V hlavičce souboru musí být odkazy na soubory, ve kterých jsou definovány vlastnosti pro simulovaného robota a senzory. Soubor *vyhybani.world* popisující simulační prostředí je na příkladě 12.2.2.

```
Obsah souboru "vyhybani.world" pro popis simulační scény

include "pioneer.inc" #definuje vlastnosti simulovaného robota Pioneer
include "map.inc"     #definuje objekt 'map' použitý pro podkladovou mapu
include "sick.inc"    #definuje sick laser scaner

size [16 16]      #velikost simulačního prostředí (v metrech)
resolution 0.02   #nastaví rozlišení laserového měřice (v metrech)
interval_sim 100  #doba trvání simulačního intervalu
interval_real 100 #doba trvání reálného intervalu

window            #nastaveni vzhledu okna GUI Stage
(
  size [ 695.000 693.000 ] #velikost okna
  center [-0.010 -0.040]   #pozice středu okna
  scale 0.028              #poměr velikosti okna
)

map   #načte bitmapu vzhledu prostředí
(
  bitmap "cave.png" #název podkladové bitmapy
  size [16 16]      #velikost prostředí (v metrech)
  name "cave"       #název prostředí
)

pioneer2dx        #vlastnosti simulovaného robota
(
  name "garcia"   #název modelu robota
  color "blue"    #barva modelu robota
  pose [-7 -6 90] #základní pozice robota na mapě
  sick_laser()    #implementovaný laserový měřič
  watchdog_timeout -1.0  #doba hlídání komunikace s robotem
)
```

Příklad 12.2.2: *Vyhybani.world* popisující vlastnosti simulační scény

Pro dokončení vytvořené simulační scény je potřeba vytvořit bitmapu pro simulační scénu. Když máme nadefinované soubory CFG a WORLD, je potřeba ještě vytvořit podkladový obrázek pro zobrazení simulovaného prostředí, ve kterém Stage rozpozná čáry a vytvoří z nich překážky, viz. obrázek 12.5. Pro spuštění simulace je třeba z terminálu nejdříve spustit Player s argumentem konfiguračního souboru *./player vyhybani.cfg*. Tímto způsobem vytvoříme server Player, na kterém běží simulace. Když chceme simulaci ovládat, musíme se k serveru Player připojit. Připojení obstará knihovna *PlayerClient* implementovaná ve zdrojovém kódu řídící aplikace. Pokud při spouštění serveru Player neudáme port, na kterém pak server očekává připojení, je port definován na základní hodnotu 6665. Při připojování řídící aplikace pak nemusíme zadávat žádný port a aplikace se připojí na základním portu 6665. Klientskou aplikaci spustíme z terminálu příkazem *./vyhybani* a získáme simulaci s připojenou řídící aplikací podle obrázku 12.5.



Obrázek 12.5: Simulování vyhýbání se překážkám pro Player/Stage

12.3 Porovnání způsobů implementace

Když porovnáme oba způsoby implementace nových projektů, je implementace aplikace v prostředí Player/Stage jednodušší, nicméně nemáme žádné pomocné nástroje a všechny soubory musí být sepsány ručně. Pro sestavení simulačního prostředí můžeme jako pomocný nástroj použít pouze editor obrázků, který používáme při simulaci prostředí. V MRDS je nejtěžší sestavit manifest, který propojí všechny potřebné služby dohromady. Je velmi vhodné použít editor DSSME, který hlídá všechny závislosti a nabízí přehled již sestavených služeb podle toho kterou službu právě editujeme. Z těchto služeb můžeme vybírat a vkládat je do editovaného manifestu. Po uložení manifestu se všechny služby sloučí do jednoho souboru, který použijeme při spouštění nové služby.

Pokud používáme pro založení projektu prostředí VPL, ušetříme množství práce s psaním kódu aplikace a ošetřováním výjimek. Po tom, co exportujeme projekt z prostředí VPL do prostředí MVS, máme k dispozici zdrojový kód, ve kterém již můžeme jen vyvíjet řídící aplikaci a nemusíme importovat knihovny ani ošetřovat výjimky.

Pokud porovnáme simulační prostředí, tak Stage nabízí hned v základu zobrazení trasy, sledování výpočtu snímání laseru, zobrazení detekce a převodu obrázku na překážky a další nástroje, které v prostředí VSE v základní konfiguraci nejsou a je třeba je na-

programovat. Největší výhodou Stage je to, že se zde velmi rychle implementuje nová simulační scéna, pouhým načtením obrázku. Výhodou VSE je, že podporuje akcelerační hardware a můžeme prostředí upravovat s použitím prostředí MVS. Když vytváříme simulační prostředí pomocí MVS, můžeme si VSE upravovat pomocí naprogramování nových funkcí.

Výhodou Player/Stage je jednoduchost programování aplikací pouze za pomocí textového editoru. MRDS musí používat náročnější grafické nástroje, protože například soubory manifest jsou většinou složité a bylo by opravdu náročné je sestavovat ručně. Na druhou stranu tyto nástroje zjednodušují práci, která je časově náročná, například hledání chyb, protože tyto nástroje hlídají správnost vytvářených souborů.

Kapitola 13

Závěr

Cílem práce bylo porovnání dvou prostředí na řízení mobilních robotů. Porovnávána byla prostředí Microsoft Robotics Developer Studio běžící na platformách Windows. Druhé prostředí je Player/Stage, které podporuje platformy Linux, Mac OS X, BSD, a další platformy standardu POSIX. Porovnávání se týká architektury, instalace prostředí a aktualizací, podpory pro uživatele, srozumitelnosti dokumentace, podporovaných platforem, přenositelnosti vytvořených aplikací, projektů při jejichž vývoji se obě prostředí používají, ceny prostředí, způsobu a vývoje nových aplikací.

První kroky při seznamování prostředí, vyznívají ve prospěch prostředí Player/Stage, když pominu problémy s instalací, chybějícím souborem kódů barev a špatným ovladačem, které byly v následující verzi odstraněny. Pro začátečníka jsou vhodnější příklady aplikací, které jsou jednoduché. Příklady Player/Stage jsou koncipovány pouze v několika souborech a jsou tedy přehledné. Předpokladem při procházení příkladů je základní znalost programování.

Začátky v MRDS jsou koncipované pro prostředí VPL a pro použití s reálným roboty. Především se mi zdálo, že vývojáři předpokládají používání robota Lego NXT. Po nainstalování prostředí MRDS je dostupné velké množství různých příkladů, ale jen málo příkladů jak si vytvořený příklad upravit pro vlastní aplikaci. Simulační úlohy jsou také připraveny v příkladech, ale pro začátečníka je vytvoření vlastní simulace složitá úloha. Až při zevrubnějším zkoumání prostředí dokumentace a dalších dostupných materiálů je příprava simulačního prostředí srozumitelnější. Především chybí souhrnný popis příkladů, který by napověděl co všechno je v systému možné. V dokumentaci, kde se popisuje prostředí MRDS převládají komerční informace, které chválí produkt a použité prvky systému a jen málo informací, které opravdu přiblíží funkci systému. Přesto pro začátečníky je vhodnější prostředí MRDS, protože obsahuje visuální prostředí VPL, ve

kterém se velice přehledně a jednoduše sestaví jednoduché aplikace.

Další výhodou prostředí MRDS jsou dostupné pomocné nástroje. Pro generování nového projektu export z prostředí VPL, nebo nástrojem DssNewService. Důležitý nástroj je grafické prostředí DSSME, pro sestavování manifestů, které dovolí provádět pouze operace, které jsou dovolené. Nabízí služby, které se vztahují k dané operaci. Prostředí MRDS nabízí několik dalších nástrojů popsaných v práci, které používají především v příkazové řádce DSSCP.

Pokud by si uživatel pořizoval prostředí pro výrobu aplikací pro komerční účely, je velká výhoda pořídit si prostředí pro které je podpora ze strany výrobce. Vzhledem k tomu, že Player/Stage je vyvíjen komunitou a dobrovolníky, není možné očekávat podporu pro Player/Stage v každé situaci. Naproti tomu MRDS se prodává v ceně 10 000 Kč a zde již uživatel podporu vyžadovat může, protože je obsahem licenčního ujednání.

Zpracovaná práce shrnuje informace z několika zdrojů do jednoho textu a přináší subjektivní pohled na práci s vývojovými prostředími. Vzhledem k tomu, že práce byla psána více než rok mohl jsem sledovat vývoj obou prostředí. Během psaní práce prostředí Player/Stage neprošlo příliš velkými změnami. Chybějící části webové stránky jsou celou dobu stejné, vyšla jedna aktualizace a chování prostředí se nezměnilo. Zatímco prostředí MRDS prodělalo bouřlivé změny. Vyšly hned tři nové verze prostředí, byla změněna webová stránka, přibylo množství nových informací na různých webových stránkách a hlavně přibyl více než dvojnásobek nových příkladů pro aplikace.

Díky dlouhé historii se prostředí Player/Stage rozšířilo hlavně do akademické sféry, kde převážilo nad ostatními vývojovými prostředími. Katedra kybernetiky na fakultě elektrotechnické v Praze, kde tato práce vznikala také používá při vývoji prostředí Player/Stage, proto se práce více zaměřila na zkoumání vlastností a chování vývojového prostředí MRDS.

Literatura

[1] Microsoft Corporation. *Webová stránka Microsoft Robotics Developer Studia [online]*. http://www.microsoft.com/robotics/, 2009.

[2] Professional Microsoft Robotics Developer Studio. *Webové stránky Professional Microsoft Robotics Developer Studio [online]*. http://www.promrds.com/, 2009.

[3] Microsoft Corporation. *Porovnání verzí Microsoft Robotics Developer Studia [online]*. http://www.microsoft.com/robotics/#ProductMatrix, 2009.

[4] Microsoft Corporation. *Stažení Microsoft Robotics Developer Studia [online]*. http://msdn.microsoft.com/cs-cz/robotics/aa731520(en-us).aspx.

[5] Sun Microsystems. *Webová stránka projektu VirtualBox [online]*. http://www.virtualbox.org, 2009.

[6] Microsoft Corporation. *Knihovny poskytované Microsoft Robotics Developer Studiem [online]*. http://msdn.microsoft.com/en-us/library/microsoft.ccr.adapters.winforms.aspx, 2009.

[7] The Player Project. *Webová stránka projektu Player [online]*. http://playerstage.sourceforge.net/, 2009.

[8] The Player Project. *Webová stránka knihoven Player [online]*. http://playerstage.sourceforge.net/doc/Player-2.1.0/player/architecture.html, 2009.

[9] Microsoft Corporation. *Novinky Microsoft Robotics Developer Studia [online]*. http://www.microsoft.com/feeds/msdn/en-us/robotics/rss.xml, 2006.

[10] The Player Project. *Webová stránka pro stažení Player [online]*. http://sourceforge.net/project/showfiles.php?group_id=42445, 2003.

[11] Microsoft Corporation. *Manuál pro Microsoft Robotics Developer Studio [online]*. http://msdn.microsoft.com/robotics, 2006.

[12] The Player Project. *Dokumentace projektu Player [online]*. http://playerstage.sourceforge.net/index.php?src=doc, 2009.

[13] The Player Project. *Webová stránka pro nahlášení chyb v projektu Player [online]*. http://sourceforge.net/tracker/?group_id=42445, 2009.

[14] Nabble. *Webová stránka Nabble fórum [online]*. http://www.nabble.com/Player-Stage-Gazebo-f4302.html, 2009.

[15] Google. *Internetový vyhledávač Google [online]*. http://www.google.com, 2009.

[16] Kyle Johns and Trevor Taylor. *Professional Microsoft Robotics Developer Studio*. Wiley Publishing, 2008.

[17] Sara Morgan. *Programming Microsoft Robotics Developer Studio*. Microsoft Press, 2008.

[18] Tomáš Petříček. *Roboti od LEGA a Microsoft Robotics Developer Studio [online]*. http://www.vyvojar.cz/Articles/429-roboti-od-lega-a-microsoft-robotics-studio.aspx, 2006.

[19] Fischertechnik. *Webové stránky firmy Fischertechnik [online]*. http://www.fischertechnik.de/, 2009.

[20] iRobot. *Webové stránky firmy iRobot [online]*. http://www.irobot.cz/, 2009.

[21] Lego. *Webové stránky firmy Lego Midstorms[online]*. http://mindstorms.lego.com, 2009.

[22] Kondo. *Webové stránky firmy Kondo [online]*. http://www.kondo-robot.com/EN/, 2009.

[23] MobileRobots. *Webové stránky firmy MobileRobots [online]*. http://www.mobilerobots.com/, 2009.

[24] ROBOSOFT. *Webové stránky firmy ROBOSOFT [online]*. www.robosoft.fr, 2009.

[25] Microsoft Corporation. *Partneři Microsoft, kteří používají MRDS [online].* http://msdn.microsoft.com/en-us/robotics/bb383566.aspx, 2009.

[26] The Player Projecta. *Wiki uživatelé Player [online].* http://playerstage.sourceforge.net/wiki/PlayerUsers, 2009.

[27] Robotnik. *Webová stránka firmy Robotnik [online].* http://www.robotnik.es/automation/robotnik-e.php, 2009.

[28] White Box Robotics. *Webová stránka firmy White Box Robotics [online].* http://www.whiteboxrobotics.com/Product/index.html, 2009.

[29] CoroWave. *Webová stránka CoroWave - robot Explorer [online].* http://www.coroware.com/explorer.aspx, 2009.

[30] CoroWave. *Webová stránka CoroWave - robot CoroBot [online].* http://www.corobot.net/, 2009.

[31] SimplySim 3D Simulation Experts. *Webová stránka firmy SimplySim [online].* http://www.simplysim.net/, 2009.

Příloha A

Zkratky používané v textu

.NET	dot NET	Programovací platforma Microsoft
API	Application Programming Interface	Rozhraní pro programování aplikace
CCR	Concurrency and Coordination Runtime	Runtime pro souběžné a koordinované aplikace
C#	Csharp	Objektově orientovaný jazyk Windows
CTP	Community Technical Preview	Komunitní technický přehled
DSS	Decentralized Software Services	Decentralizované softwarové služby
DSSCP	DSS Command Prompt	DSS příkazová řádka
DSSME	DSS Manifest Editor	Editor manifest souboru
DSSP	DSS Protocol	Protokol DSS služeb
FSE	FujitsuSiemens Esprimo V5545	Notebook FS Esprimo V5545
GNU	GNU General Public License	všeobecná veřejná licence GNU
HTTP	Hypertext Transfer Protokol	Protokol pro přenos data
MRDS	Microsoft Robotics Developer Studio	Robotické vývojové studio
MSDNAA	MSDN Academic Aliance	v rámci licence mohou studenti bezplatně používat software od firmy Microsoft
REST	Representational state transfer	Styl softwarové architektury pro distribuované systémy jako je WWW
SLAM	Simultaneous Localisation And Mapping	Algoritmus současné lokalizace a mapování
SOAP	Simple Object Access Protocol	Protokol pro výměnu zpráv založených na XML přes síť, hlavně pomocí http
Tcl	Tcl	programovací jazyk založený na Lispu
URI	Uniform Resource Identifier	Jednotný identifikátor zdroje
VPL	Visual Programming Language	Vizuální programovací jazyk v RDS
VSE	Visual Simulation Environment	Simulační prostředí pro roboty
WPF	Windows Presenation Framework	Knihovna pro vytváření grafického rozhraní

Tabulka A.1: Zkratky používané v textu

Příloha B

Zdrojový kód vyhýbání se překážkám - MRDS

```
Část aplikace pro řízení robota v prostředí MRDS

// Import potřebných knihoven s robotickými službami
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Xml;
using ccr = Microsoft.Ccr.Core;
using dss = Microsoft.Dss.Core;
using dssa = Microsoft.Dss.Core.Attributes;
using dssh = Microsoft.Dss.Core.DsspHttp;
using dssm = Microsoft.Dss.ServiceModel.DsspServiceBase;
using dssp = Microsoft.Dss.ServiceModel.Dssp;
using soap = W3C.Soap;
using submgr = Microsoft.Dss.Services.SubscriptionManager;
using drive = Microsoft.Robotics.Services.Drive.Proxy;
using sicklrf = Microsoft.Robotics.Services.Sensors.SickLRF.Proxy;

namespace Robotics.Vyhybani.Diagram
{
[DisplayName("Vyhybani")]   /// Jméno služby
[Description("Robot se vyhýbá překážkám")]  // Popis služby
```

```
[dssa.Contract(Contract.Identifier)]
public class DiagramService : dssm.DsspServiceBase
{
   // Inicializace stavu aplikace
   [dssa.InitialStatePartner(Optional = true)]
   private DiagramState _state;

   // Port pro služby, lze se na něj připojit v internetovém
   // prohlížeči na http://localhost/Vyhybani
   [dssa.ServicePort("/Vyhybani", AllowMultipleInstances = true)]
   private DiagramOperations _mainPort = new DiagramOperations();

   // Port pro zabezpečenou komunikaci, přihlášení vyžaduje zadat
   //  heslo Windows pro prohlížení v internetovém prohlížeči
   [dssa.Partner("SubMgr",
      Contract = submgr.Contract.Identifier,
      CreationPolicy = dssa.PartnerCreationPolicy.CreateAlways)]
   private submgr.SubscriptionManagerPort _subMgr =
   new submgr.SubscriptionManagerPort();

   // Definice pro spojení s laserovým měřiče
   // Partner: SimulatedLaserRangeFinder
   // Contract: http://schemas.microsoft.com/xw/2005/12/sicklrf.html
   [dssa.Partner("SimulatedLaserRangeFinder",
    Contract = sicklrf.Contract.Identifier,
    CreationPolicy =
      dssa.PartnerCreationPolicy.UsePartnerListEntry)]
    sicklrf.SickLRFOperations _simulatedLaserRangeFinderPort =
     new sicklrf.SickLRFOperations();
    sicklrf.SickLRFOperations _simulatedLaserRangeFinderNotify =
     new sicklrf.SickLRFOperations();

   //Definice pro spojení s nápravou robota s dvěma motory
   //Partner: SimulatedGenericDifferentialDrive
   //Contract:http://schemas.microsoft.com/robotics/2006/05/drive.html
```

```
[dssa.Partner("SimulatedGenericDifferentialDrive",
Contract = drive.Contract.Identifier,
CreationPolicy = dssa.PartnerCreationPolicy.UsePartnerListEntry)]
 drive.DriveOperations _simulatedGenericDifferentialDrivePort =
   new drive.DriveOperations();

// Vytvoření portu pro komunikaci
public DiagramService(dssp.DsspServiceCreationPort creationPort)
  : base(creationPort)
  {}
// Hlavní metoda, která inicializuje celou aplikaci
 protected override void Start()
 {  // Pokud není žádný partner pro inicializaci, stav bude null
 if (_state == null)
{ // Stav služby musí být vytvořen  před tím, než se spustí
 // posílání zpráv mezi procesy
   _state = new DiagramState();
}

// Zbytek startovního procesu požaduje schopnost čekat na odezvy
// od služeb a od spouštěcího handleru, pokud nějaký existuje.
SpawnIterator(DoStart); }

/// Start aplikace. Vrací chybu aplikace, pokud se vyskytne
private IEnumerator<ccr.ITask> DoStart()
{  soap.Fault fault = null;

// Připojení k partnerům
yield return ccr.Arbiter.Choice(
_simulatedLaserRangeFinderPort.Subscribe
   (_simulatedLaserRangeFinderNotify,
    typeof(sicklrf.Replace)
  ),
 EmptyHandler,
 delegate(soap.Fault f)
```

```
 {
  fault = f;
 }
);

// Aktivace nezávislých služeb
Activate<ccr.ITask>(
   ccr.Arbiter.ReceiveWithIterator<sicklrf.Replace>(true,
   _simulatedLaserRangeFinderNotify,
   SimulatedLaserRangeFinderReplaceHandler)
);  yield break;  }

private void StartHandlers()
{
// Aktivace zpráv pro manipulaci se službami
base.Start();  }

// Manipulátory pro řídící zprávy
IEnumerator<ccr.ITask>
SimulatedLaserRangeFinderReplaceHandler(sicklrf.Replace message)
{
 OnSimulatedLaserRangeFinderReplaceHandler handler =
  new OnSimulatedLaserRangeFinderReplaceHandler(this,
 Environment.TaskQueue);
return handler.RunHandler(message);
}

// Vlastní řídící aplikace///
// Řídí robota tak, aby nenarazil do překážky odbočil na volné místo
class OnSimulatedLaserRangeFinderReplaceHandler : HandlerBase
{

public OnSimulatedLaserRangeFinderReplaceHandler(
 DiagramService service, ccr.DispatcherQueue queue)
: base(service, queue)
```

```
{}

public IEnumerator<ccr.ITask> RunHandler(sicklrf.Replace message)
{
// Získání vzdálenosti z čidla vzdálenosti. Vyberou se
// dva body pod úhlem 45° na předku robota
// podle kterých se uhýbá robot před překážkami ve
//vzdálenosti 1500mm před překážkou.
if (((    // Uhýbání před překážkou doleva
message.Body.DistanceMeasurements[
(message.Body.DistanceMeasurements.Length / 8) * 3]) >
(message.Body.DistanceMeasurements[
(message.Body.DistanceMeasurements.Length / 8) * 5])) &&
((message.Body.DistanceMeasurements[
(message.Body.DistanceMeasurements.Length / 8) * 5]) <
1500))
{
// Při přiblížení se k překážce, která je zprava, zatočí
// o úhel 45° na levou stranu,
// při uhýbání sníží rychlost na 60%, aby se nepřevrátil
drive.RotateDegreesRequest request =
 new drive.RotateDegreesRequest();
 request.Power = 0.3D;           // rychlost na 30% maxima
 request.Degrees = (double)-40;  // zatočení na levou stranu
// zapsání příkazu do robota
SimulatedGenericDifferentialDrivePort.RotateDegrees(request);
}
else if (((    // Uhýbání před překážkou do prava
 message.Body.DistanceMeasurements[
(message.Body.DistanceMeasurements.Length / 8) * 3]) <
(message.Body.DistanceMeasurements[
(message.Body.DistanceMeasurements.Length / 8) * 5])) &&
((message.Body.DistanceMeasurements[
(message.Body.DistanceMeasurements.Length / 8) * 3]) <
1500))
```

```
{
// Při přiblížení se k překážce, která je zleva, zatočí o úhel
// 45° na pravou stranu,
// při uhybání sníží rychlost na 60%, aby se nepřevrátil
drive.RotateDegreesRequest requestA =
  new drive.RotateDegreesRequest();
requestA.Power = 0.3D;          // rychlost na 30% maxima
requestA.Degrees = (double)40;  // zatočení na pravou stranu
SimulatedGenericDifferentialDrivePort.RotateDegrees(requestA);
}
else
{ // Pokud není žádná překážka v dosahu jede robot přímým směrem
 drive.SetDrivePowerRequest requestB =
  new drive.SetDrivePowerRequest();
 requestB.RightWheelPower = 0.5D;  // rychlost při přímém pohybu
 requestB.LeftWheelPower = 0.5D;  // rychlost při přímém pohybu
 SimulatedGenericDifferentialDrivePort.SetDrivePower(requestB);
 }

// Odešle kompletní příkaz do robota
base.Complete.Post(ccr.EmptyValue.SharedInstance);
yield break;
}

class JoinAlpha // Funkce pro zápis rychlosti do robota
{
 public string LeftWheelPower;
 public string RightWheelPower;
 public JoinAlpha()
 {}
public JoinAlpha(object[] args)
{
 LeftWheelPower = args[0].ToString();
 RightWheelPower = args[1].ToString();
}
```

```
}

 // Vytvoření portu pro zápis rychlosti do robota
 ccr.Port<object>[] _joinAlphaPorts = new ccr.Port<object>[2]{
  new ccr.Port<object>(),
  new ccr.Port<object>()
 };
 // Funkce pro zápis informací o otáčení do robota
 class JoinBeta
 {
  public string Degrees;
  public string Power;
  public string RotateDegreesStage;
  public JoinBeta()
  {}
  public JoinBeta(object[] args)
  {
  Degrees = args[0].ToString();
  Power = args[1].ToString();
  RotateDegreesStage = args[2].ToString();
  }
 }
 // Vytvoření portu pro zápis otáčení do robota
 ccr.Port<object>[] _joinBetaPorts =
  new ccr.Port<object>[3]{
  new ccr.Port<object>(),
  new ccr.Port<object>(),
  new ccr.Port<object>()
 };}
```

Příloha C

Obsah přiloženého CD

- Implementovaná aplikace
  - Vyhýbání se pro MRDS
  - Vyhýbání se pro Player/Stage
- Instalační soubor MRDS
- Instalační soubory Player/Stage
- Zdrojové soubory TEX