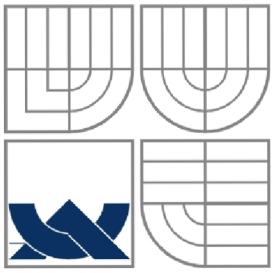


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

Faculty of mechanical engineering
Institute of Automation and Computer Science

POČÍTAČOVÁ SIMULACE POHYBU A PLÁNOVÁNÍ TRAJEKTORIE MOBILNÍHO ROBOTU V PSEUDO 3D PROSTŘEDÍ

COMPUTER PATH PLANNING SIMULATION AND CALCULATION OF A MOBILE ROBOT IN PSEUDO-3D ENVIRONMENT

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE — **Bc. ZDENĚK KOCH**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. RADOMIL MATOUŠEK, Ph.D.**
SUPERVISOR

BRNO 2008

List zadání – přední část
(bude vloženo)

Chtěl bych touto formou poděkovat vedoucímu své diplomové práce Ing. Radomilu Matouškovi, Ph.D. za věcné připomínky ohledně zpracování této práce. Speciální poděkování také patří moji přítelkyni Ivaně Kapounové, která mi pomohla s korekcí a díky jejím připomínkám při psaní je text snad dobře srozumitelný. Taktéž bych chtěl poděkovat své rodině, která mě v mém životě a studiu vždy plně podporovala.

Abstrakt

Tato práce se zabývá návrhem a realizací softwarové aplikace „Mobile robot studio“ pro plánování trajektorie mobilního robotu v pseudo 3D prostředí. Obsahuje několik nástrojů, z nichž nejdůležitějšími jsou: ovládání simulace, plánování trajektorie, editor světa a editor příkazů pro CAN. Aplikace je vytvořena pomocí technologie .NET 2.0 a pro 3D zobrazení je využito rozhraní Microsoft DirectX 9.

Diplomová práce vznikla při řešení výzkumného záměru Inteligentní systémy v automatizaci podporovaného MŠMT ČR pod registračním číslem MSM 0021630529.

Klíčová slova

RRT, RRPC, XML, C#, .NET, CAN, CANopen, DirectX, mobilní robot, 3D prostředí, plánování trajektorie

Abstract

This thesis deals about design and realization software application "Mobile robot studio" for planning path mobile robot in pseudo 3D world. It contains several tools, witch most important are: simulation control, path planning, world editor and commands editor for CAN. Application was made by technology .NET 2.0 and for 3D design was used Microsoft DirectX 9 API.

This thesis has been supported by the Czech Ministry of Education in the frame of MSM 0021630529 Research Intention Inteligent Systems in Automation.

Keywords

RRT, RRPC, XML, C#, .NET, CAN, CANopen, DirectX, mobile robot, 3D environment, motion planning

Obsah

1 Úvod

Robot je obvykle elektromechanický stroj, který je schopen „vnímat“ svoje okolí a toto okolí ovlivňovat. Mobilní robot je takový robot, který se dokáže pohybovat v prostoru. Mobilní robot může mít rozdílnou míru autonomie – od teleoperované až po plně autonomní.

Tato práce se zabývá realizací simulačního software pro plánování trajektorie mobilního robotu v daném prostředí.

Navržený simulační software „Mobile robot studio“ pracuje s projektovým souborem (*Mobile robot project*), který obsahuje několik XML struktur pro popis prostředí a robotu. XML struktury jsou dostatečně flexibilní pro vytvoření jakéhokoliv prostředí. Simulace může proběhnout ve 2D nebo 3D prostředí. Plánování trajektorie je řešeno pomocí RRT algoritmu, přičemž k optimalizaci nalezené trajektorie je využito RRPC algoritmu. Aplikace „Mobile robot studio“ je doplněna o modul sériové komunikace, jehož prostřednictvím je možno přenést vypočtenou pohybovou trajektorii do řídících jednotek pohonu robotu.

Pro porozumění obsahu kapitoly 2, zejména části o sběrnici CAN a protokolu CANopen se vyžadují jisté základy o dané problematice, které lze naleznout např. v [4].

2 Počítačové modelování mobilního robotu

2.1 Co je to mobilní robot?

Mobilní robot je inteligentní stroj, který je schopen vykonávat nějaké úkoly. Mobilní roboti jsou řízeni ručně nebo svoji činnost vykonávají autonomně, podle toho se taky dělí na roboty řízené a na roboty autonomní. Jako vhodná aplikace se jeví situace: „Kam nemůže jít člověk, tam může jít robot.“ S tímto lze uvést použití robotů v nebezpečném prostředí (např. sonda na Marsu nebo robot na odstraňování min). Roboti jsou ovšem nejvíce vyvíjeni a nasazeni pro obsluhu automatických linek v průmyslu.

Obr 1. Ukázka mobilního robotu

2.2 Počítačové modelování

Počítačové modelování je silným inženýrským nástrojem. Reálný model je nahrazen modelem simulačním, kdy je uvažováno nad problémy pouze v užším měřítku – do simulačního modelu jsou zahrnuty pouze důležité parametry. Toto představuje většinou přípustné zjednodušení, které výrazně ulehčí, nebo dokonce umožní, řešení.

Počítačové modelování se promítá do několika vědních oborů, z nichž asi nejdůležitější jsou informatika (modelovací algoritmy), matematika (obecné i složitější výpočty), kybernetika (využití umělé inteligence), mechanika (kinematický a dynamický model) a elektrotechnika (řízení a komunikace).

2.2.1 Simulace

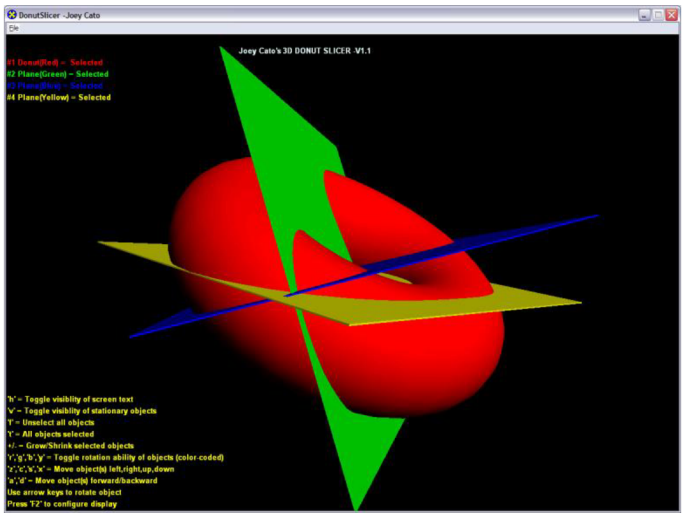
Simulací robotu můžeme zkoumat změny vstupních nebo jiných podmínek, zkoumat změny a varianty chování objektu a předpovídat tak jeho reálnou činnost. Simulace také usnadňuje programování a návrh robotů, protože víme, co můžeme očekávat od skutečného robotu.

Proč zrovna počítačové modelování? Protože právě počítač je vhodným prostředkem pro provedení simulace. Počítač disponuje dostatečnou výpočetní silou a výsledky je snadné reprezentovat (např. vizualizace pohybu robotu).

2.2.2 Vizualizace a Microsoft DirectX

Vizualizace je grafickým zobrazením dat, která dávají výsledkům simulace grafickou podobu. Zobrazovat je možno jak ve 2D tak i ve 3D.

Microsoft DirectX je soubor aplikačního rozhraní (API) pro manipulaci s úkoly souvisejícími s multimedii, zejména programování her a video, na platformě Microsoft. Největší součástí je rozhraní Direct3D, které se stará o zobrazení trojrozměrných objektů. Tohoto se hojně využívá ve hrách, ale také v jiných softwarových aplikacích pro vizualizaci (např. ve strojírenství pro CAD / CAM).



Obr 2. Ukázka zobrazení 3D objektů pomocí Microsoft DirectX

V této diplomové práci je vizualizace pohybu robotu realizována právě aplikačním rozhraním Microsoft Direct3D. Toto řešení bylo zvoleno, protože se snadno implementuje na platformě Microsoft Windows a také díky podpoře v technologii .NET, ve které je vytvořena celá aplikace „Mobile robot studio“.

2.2.3 Optimální trajektorie

V oblasti plánování trajektorie pohybu mobilního robotu figuruje základní otázka. Jak vytvořit optimální trajektorii robotu? Optimální trajektorie by měla být taková trajektorie, která vyhovuje počátečním podmínkám a je nalezena v konečném čase. Plánováním trajektorie pro mobilní roboty více pojednává kapitola 3.

2.3 Řízení pomocí CAN a protokol CANopen

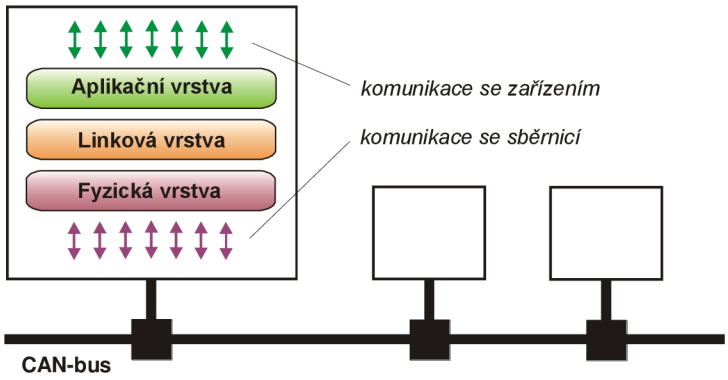
Jedním z dalších problémů vyplývajících z celkové koncepce robotů je návrh vhodného způsobu komunikace mezi pohonnými jednotkami robotu a komunikace robotu s nadřazeným systémem (počítač, programovatelný automat).

U mobilních robotů může být využita sběrnice typu CAN, která má rozsáhlé možnosti aplikace a je standardem v průmyslové komunikaci. Sběrnice CAN je využita i v případě této diplomové práce, pro komunikaci mezi počítačem a reálným mobilním robotem.

2.3.1 Sběrnice CAN

CAN - Controller Area Network. Tato sběrnice byla původně vyvinuta firmou Bosch pro rychlé přenosy dat v automobilové technice. Hlavním omezením při vývoji byl především požadavek na co nejmenší náklady. Díky cenové efektivnosti, spolehlivosti, vysoké přenosové rychlosti, snadnému nasazení a dostupnosti potřebné součástkové základny, se sběrnice CAN začala uplatňovat i v ostatních průmyslových aplikacích.

Protokol CAN je definován normou ISO 11898, která popisuje fyzickou vrstvu protokolu a také specifikaci CAN 2.0A. Později byla ještě vytvořena specifikace CAN 2.0B. Tyto specifikace definují pouze fyzickou a linkovou vrstvu protokolu podle referenčního modelu ISO/OSI (viz Obr. 3. Aplikační vrstva protokolu CAN je definována několika vzájemně nekompatibilními standardy (CAL/CANopen, DeviceNet, atd.).



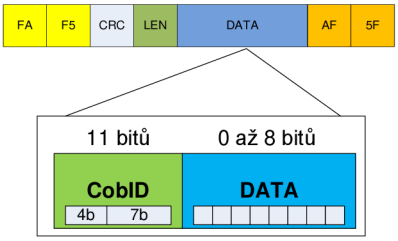
Obr 3. ISO-OSI třívrstvý model pro sběrnici CANopen

2.3.2 Protokol CANopen

CANopen je nezávislý popisný jazyk určený pro komunikaci po sběrnici CAN. CANopen zajišťuje základní služby datové komunikace podle třívrstvového modelu (viz Obr 3.).

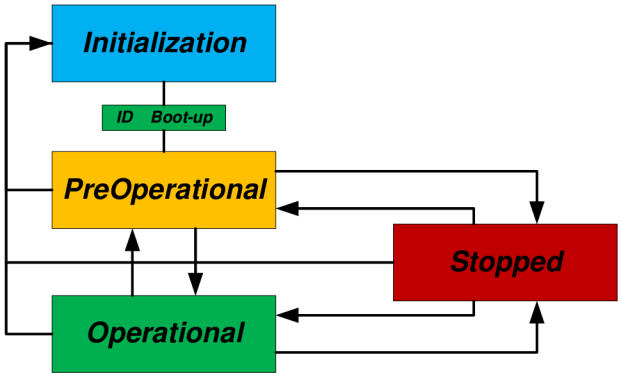
Aplikační vrstva poskytuje aplikačním procesům přístup ke komunikačnímu systému a umožňuje provádění základních funkcí jako je vytváření, udržování a rušení spojení mezi zařízeními, čtení a zápis přenášených dat.

Síťový management CANopen výrazně zjednodušuje návrh řídicích systémů, poskytuje síťové služby *NMT*, časovou synchronizaci procesů *SYNC* a *TIME STAMP*, zabezpečení provozu *NODE GUARDING*, chybová hlášení *EMERGENCY,* přenos řídicích dat *PDO* (*Proces Data Objects*) a především poskytuje možnost rozsáhlých servisních nastavení prostřednictvím *SDO* (*Service Data Objects*).



Obr 4. CAN 2.0A a CANopen zpráva
CobID – identifikátor zprávy (objektu)
DATA – data zprávy (objektu)

NMT – Síťový management (Network Management) je soubor nástrojů pro správu sítě, jeho objekty jsou sítový objekt (Network object), vzdálený uzel (Remote Node) a uzel (Node), poskytované služby jsou Control, Error a Configuration services.



Obr 5. Stavový diagram stavů uzlu v CANopen síti – přepnutí pomocí NMT služby
Initialization – inIclAlizace zařízení
PreOperational – stav ve kterém je zařízení schopno přijímat příkazy
Stopped – zařízení je zastaveno
Boot-up – identifikace zařízení v síti

PDO – Přenos datových objektů (*Proces data objects*). Tyto objekty slouží pro rychlý přenos dat v síti CAN. Jsou tedy určeny především pro aplikace pracující v reálném čase. Objekty jsou T_PDO1, R_PDO1, T_PDO2, R_PDO2.

SDO – (*Service data objects*). Jsou určeny pro přístup do objektových slovníků zařízení – čtení / zápis. Objekty jsou T_SDO a R_SDO.

LMT – (*Layer management objects*). Objekty vrstvového managementu. Obsahují služby pro nastavení bázových adres a přenosových rychlostí.

EMERGENCY – Bezpečnostní objekty pro zobrazování chyb zařízení nebo jejich periférií.

SYNC – Objekty pro synchronizaci síťových zařízení. Odesláním synchronizačního objektu jsou v uzlech akceptovány všechny synchronní zprávy. Tím je možno např. spustit tři pohony najednou bez prodlevy komunikace

NODE GUARDING – zprávy zabezpečení

Podrobnější popis jednotlivých služeb protokolu CANopen a formátování zpráv je popsáno např. v [4].

3 Plánování trajektorie

Plánování trajektorie je jednou z důležitých částí celého počítačového modelování. V drtivé většině případů se používají algoritmy, které prohledávají stavový prostor robotu a snaží se nalézt cestu od startu k cíli. Stavový prostor může být úplný, kdy je známo prostředí, ve kterém se bude robot pohybovat (případ diplomové práce), anebo neúplný, kdy je známo pouze malé okolí robotu a dalším pohybem robotu se tento stavový prostor zvětšuje (např. plně autonomní robot vybavený senzory).

Prohledávacích algoritmů je známo několik, ať už jsou to neinformované (např. breadth-first search, depth-first search), nebo informované (např. hill-climbing algorithm, best-first search, A star), tak nikoliv všechny jsou vhodné pro plánování trajektorie. Jedním z důležitých požadavků je čas, za který by měla být cesta nalezena. Zde získávají vysoký potenciál algoritmy využívající nějakým způsobem rozdělení pravděpodobnosti nad stavovým prostorem. Jedním takovým algoritmem je i algoritmus RRT, který bude rozebrán v další části textu.

V aplikaci „Mobile robot studio“ se na plánování trajektorie podílí generování pomocí RRT a optimalizace pomocí RRPC.

3.1 RRT – generování stromu

3.1.1 Co je RRT?

Rapidly-exploring Random Tree (RRT, rychle rostoucí náhodný strom) je datová struktura a algoritmus vytvořený pro efektivní vyhledávání v nekonvexních více dimenzionálních prostorách. RRT je vytvářen přírůstkově a to takovým způsobem, který rychle snižuje očekávanou vzdálenost mezi uzlem stromu a náhodným bodem. RRT jsou obzvláště vhodné pro problémy plánování cest, které zahrnují překážky a jiná omezení (statická i dynamická).

RRT mohou být použity pro tvorbu otevřených trajektorií v nelineárních systémech. RRT se běžně považuje za Monte-Carlo způsob hledání ve velkých Voronoiových diagramech. Některé varianty lze považovat za stochastické fraktály.

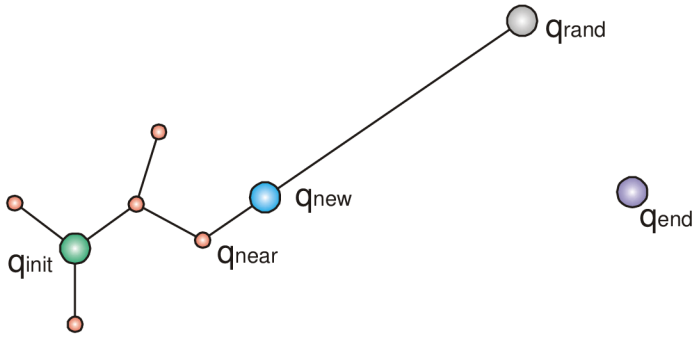
Obvykle je samotný algoritmus RRT nedostatečný k vyřešení problému plánování a je jej potřeba doplnit o další techniky. RRT lze proto považovat za součást, která může být začleněna do vytváření různých plánovacích algoritmů.

Plánovaní pohybu a algoritmu RRT se velmi podrobně věnuje [1], kde je na příkladech ukázán potenciál tohoto algoritmu v nespočetném množství modifikací.

3.1.2 Popis algoritmu

Zde je stručný popis metody pro obecný konfigurační prostor C (to lze považovat za obecný stavový prostor, který by mohl zahrnovat polohu a další informace o aktuální konfiguraci robota např. natočení ramene, dynamické vlastnosti apod.). RRT má kořen v konfiguraci q_{init} a má K vrcholů který jsou vytvořeny dle následujícího algoritmu:

```
BUILD_RRT(q_init, K, Δq)
  1. G.init(q_init);
  2. for i = 1 to K
  3.       q_rand ← RAND_CONF();
  4.       q_near ← NEAREST_VERTEX(q_rand, G);
  5.       q_new  ← NEW_CONF(q_near, Δq);
  6.       G.add_vertex(q_new);
  7.       G.add_edge(q_near, q_new);
  8. return G;
```



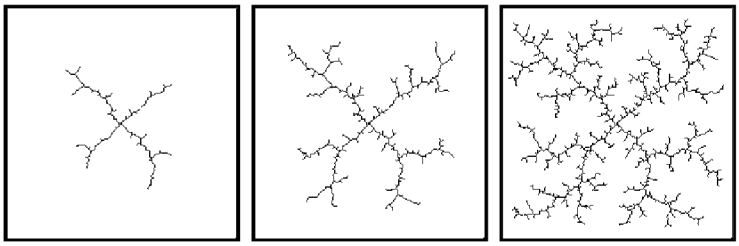
Obr 6. Vytvoření nové konfigurace pomocí RRT.

V kroku 3 je výběr náhodné konfigurace q_{rand} z prostoru C. Je možné přepsat metodu RAND_CONF() na RAND_FREE_CONF(), která vybírá náhodnou konfiguraci z prostoru omezeného nějakými podmínkami (např. použití detekce kolizí při obcházení překážek).

V kroku 4 se vybírá q_{near} vrchol z RRT, který je nejblíže náhodné konfiguraci q_{rand}. Implementace metody NEAREST_VERTEX(q_{rand}, G) může být následující:

```
NEAREST_VERTEX(q_rand, G)
1. d ← ∞;
2. for each v ∈ V
3.       if p(q_rand, v) < d then
4.            v_new = v;
5.            d ← p(q_rand, v);
6. return G;
```

V kroku 5 metoda NEW_CONF(q_{near}, Δq) vybírá novou konfiguraci q_{new}, která je od q_{near} posunuta o vzdálenost Δq. Předpokládá se ovšem, že pohyb v jakémkoli směru je možný. Pokud existuje omezení, pak jsou vstupy pro odpovídající kontrolní systém a nové konfigurace získané metodou numerické integrace. A nakonec v kroku 6 a 7 se vytvoří nový vrchol a nová hrana, které jsou přidány do grafu G. Po vytvoření K vrcholů je graf G vygenerován a může posloužit k vyhledání cesty.

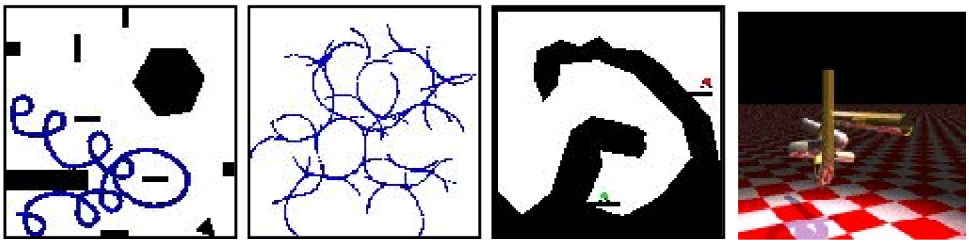


Obr 7. Růst stromu s pomocí algoritmu RRT ve 2D prostoru.

3.1.3 Modifikace

Generování RRT může být velice dobře přizpůsobeno, aby splnilo všechny požadavky konkrétního problému plánování.

Největšího ovlivnění růstu stromu můžeme dosáhnout v metodě NEW_CONF(q_{near}, Δq), která může vytvořit, za pomoci podmínek, pouze správnou konfiguraci. Podmínkami může být např.: detekce kolizí mezi objekty, transformace objektu (natočení, posunutí) a jiné.

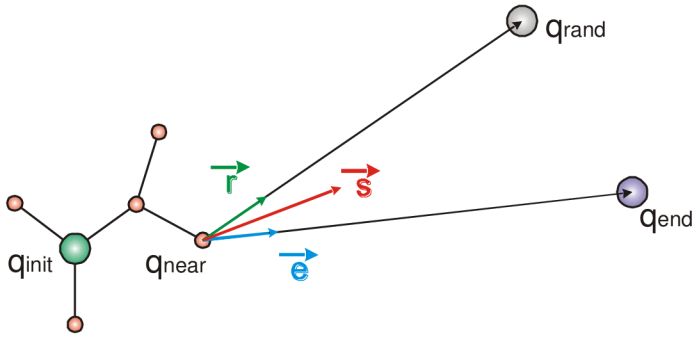


Obr 8. Ukázky modifikací generování RRT.

Generovaní nemusí probíhat v předem určeném rozsahu, ale může se uzavřít do nekonečného cyklu, kde bude nějaká platná podmínka pro dokončení cyklu. Příkladem podmínky může být nalezení cíle, kdy vzdálenost mezi novou konfigurací q_{new} a cílovou konfigurací q_{end} je menší než d.

3.1.4 Optimalizace

Jak optimalizovat generování RRT? Na to není snadná odpověď, protože záleží na konkrétním problému plánování a někdy je to spíše modifikace než optimalizace. Pokud např. se jedná o pohyb v prostoru, můžeme využít vektorové algebry a modifikovat náhodný vektor $\vec{r}(q_{near}, q_{rand})$ vektorem cíle $\vec{e}(q_{near}, q_{rand})$ za vzniku vektoru $\vec{s}$, který se použije při vytváření nové konfigurace.



Obr 9. RRT - modifikace náhodného vektoru.

Dále je možno použít např. Gaussovo rozdělení pravděpodobnosti, pro ovlivnění náhodné konfigurace v určité oblasti, nebo za zmínku stojí i variabilní krok v průběhu generování.

Všechny optimalizační metody mají jedno společné a to urychlit konvergenci k cíli a tím zkrátit čas na řešení problému.

3.2 RRPC – optimalizace trajektorie

3.2.1 Co je RRPC?

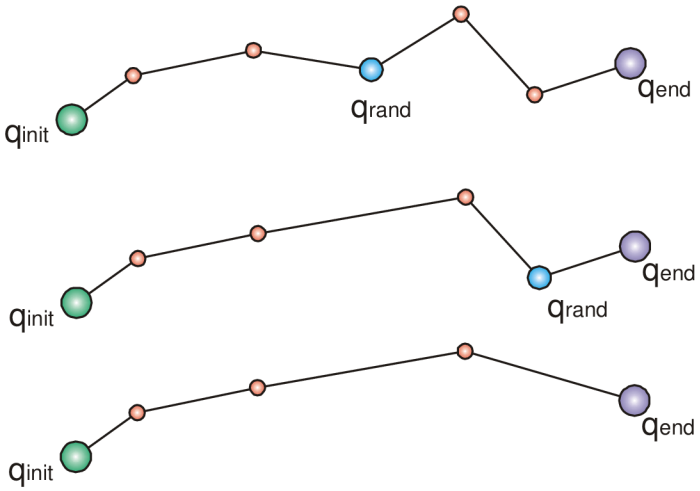
Random removing path configuration (RRPC, náhodné odebrání konfigurace cesty) je algoritmus vytvořený pro potřeby optimalizace trajektorie. RRPC se snaží nalézt takovou konfiguraci v cestě, která může být odebrána, aniž by to nějakým způsobem porušilo vstupní podmínky.

Tímto algoritmem se razantně sníží počet konfigurací v cestě, což má za následek zjednodušení cesty. V aplikaci „Mobile robot studio“ se dosahovalo zjednodušení trajektorie v řádu 50 – 90 %, dle aktuálně vygenerované cesty.

3.2.2 Popis algoritmu

Algoritmus je dvou-průchodový. V prvním průchodu se odstraní všechny přebytečné konfigurace. Ve druhém průchodu se vloží mezi každou konfigurační dvojici (konfigurace jsou vedle sebe), nová konfigurace a opět jsou odstraněny všechny přebytečné konfigurace, stejně jako v prvním průchodu.

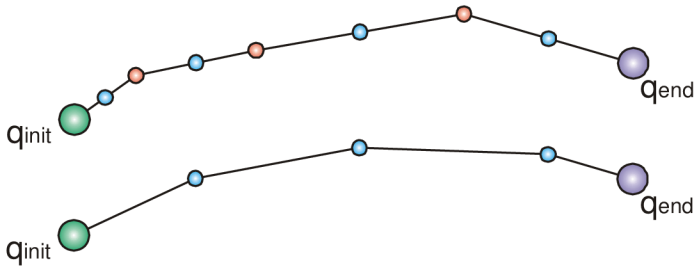
```
FIRST_PASS(P, E_max)
1. err ← 0;
2. while err < E_max
3.       q_rand ← RAND_CONF(P);
4.       if REMOVE_TEST(q_rand, P) then
5.            P.remove_conf(q_rand);
6.       else
7.            err ← err + 1;
8. return P;
```



Obr 10. RRPC – provedení prvního průchodu

V prvním průchodu se vybírá náhodná konfigurace q_{rand} metodou RAND_CONF(P) z trajektorie P. Náhodná konfigurace je otestována metodou REMOVE_TEST(q_{rand},P), zdali je možné náhodnou konfiguraci q_{rand} odebrat. Pokud ano - je odebrána, pokud nikoliv - zvětší se chyba err o jedničku. Pokud algoritmus dosáhne celkové chyby E_{max} je ukončen.

```
SECOND_PASS(P, E_max)
1. for i = 1 to P.length - 1
2.       q_new ← NEW_CONF(q_i, q_i+1);
3.       P.insert_conf(q_new, i * 2);
4. FIRST_PASS(P, E_max);
5. return P;
```



Obr 11. RRPC – provedení druhého průchodu

Ve druhém průchodu je mezi každou aktuální q_i a následující q_{i+1} konfigurací vložena nová konfigurace q_{new}. V grafické reprezentaci je q_{new} středem úsečky mezi q_i a q_{i+1}. Po té je opět proveden první průchod pro odstranění přebytečných vrcholů.

3.2.3 Vznik algoritmu

Tento algoritmus vznikl v době psaní aplikace pro potřeby optimalizace vygenerované cesty RRT algoritmem. Optimalizačních algoritmů tohoto charakteru jistě existuje celá řada, ale osobním cílem bylo vnést do aplikace tento zajímavý algoritmus.

4 Praktická realizace aplikace „Mobile robot studio“

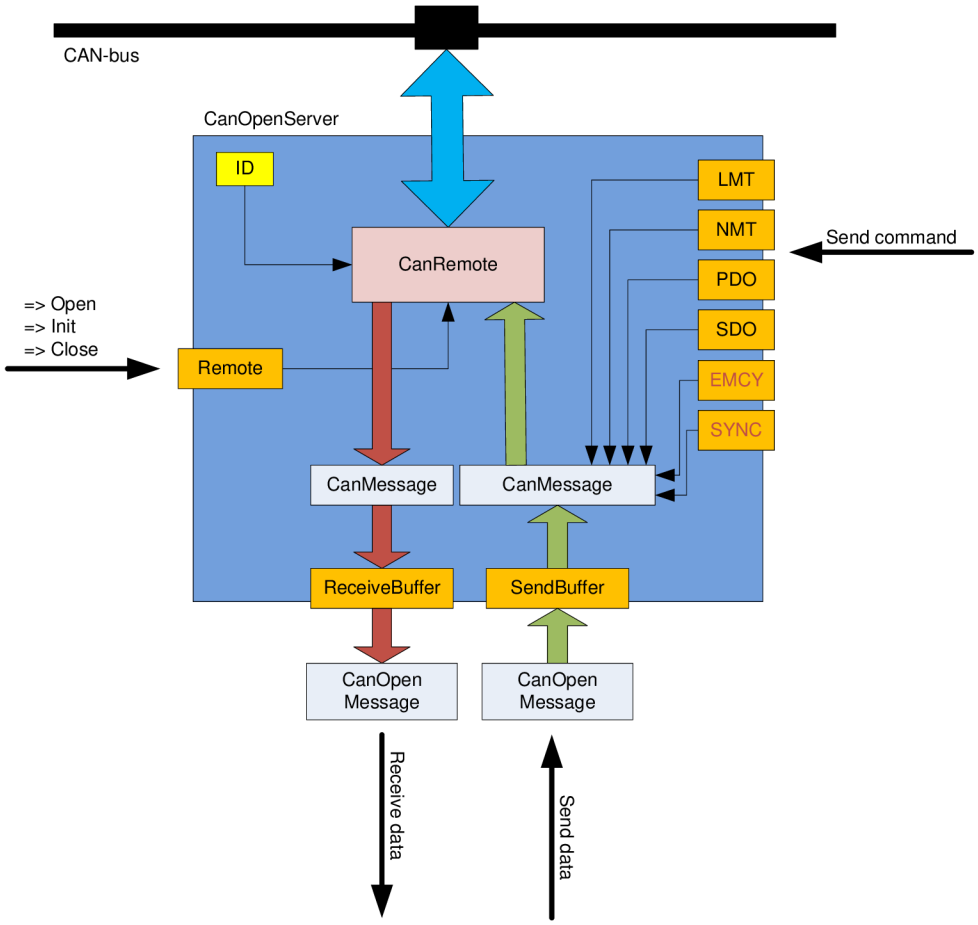
4.1 Modul „CANopen“

4.1.1 Popis modulu

Modul CANopen je tvořen samostatnou knihovnou MobileRobot.CanOpen.dll. Tento modul disponuje širokou škálou objektů pro zajištění komunikace po lince CAN a implementuje CANopen protokol 2.0 A. Nejdůležitějšími objekty jsou CanOpenServer a CanOpenMessage. Struktura modulu je uvedena v příloze [A] kapitola 1.

4.1.2 Objekt „CanOpenServer“

CanOpenServer je komunikačním rozhraním, spojujícím rozhraní CAN a aplikaci. Jádrem serveru je objekt *CanRemote*, který odesílá a přijímá fyzická data. Disponuje také metodami *Open*, *Init* a *Close*, které přistupují přímo na komunikační port. Všechna ostatní data jsou filtrována přes objekt *CanMessage*.



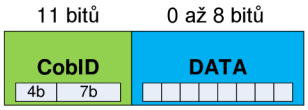
Obr 12. Schéma „CanOpenServer“

4.1.3 Objekt „CanOpenMessage“

CanMessage objekt při příjmu dat odebere všechny režijní bity a vrátí pouze reálnou zprávu. Při odesílání naopak zprávu doplní o příslušné bity tak, aby jej bylo možné odeslat.

Obr 13. Formát zprávy *CanMessage* 2.0 A

Objekty typu CanOpenMessage jsou větší abstrakcí od *CanMessage*. Zde už se pracuje pouze s datovou oblastí, která je rozdělena na dvě části CobID (identifikátor zprávy) a data.



Obr 14. Format zprávy *CanOpenMessage*

4.2 Modul „LinearAlgebra“

4.2.1 Popis modulu

Je tvořen samostatnou knihovnou MobileRobor.LinearAlgebra.dll. Tento modul obsahuje datové struktury potřebné pro popis světa ve 2D a také třídu řešící detekci kolizí mezi různými objekty. Struktura modulu je uvedena v příloze [A] kapitola 2.

4.2.2 Datové struktury

Bod (Point) – je dvousložková struktura obsahující souřadnice v ose x a v ose y. Jsou implementovány metody pro posunutí bodu a výpočet vzdálenosti dvou bodů.

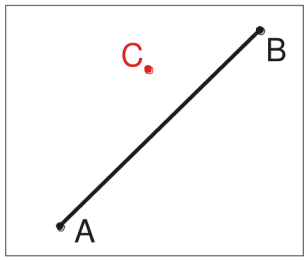
Úsečka (Segment) – je dvousložková struktura obsahující počáteční a koncový bod. Implementován je výpočet délky úsečky.

Vektor (Vector) – je dvousložková struktura obsahující souřadnice v ose x a v ose y. Jsou implementovány metody pro násobení, sčítání vektoru a normalizaci vektoru.

Polygon (Polygon) – je univerzální struktura obsahující kolekci bodů tvořící vrcholy polygonu. Jsou implementovány metody pro transformaci (posun, změna měřítka, rotace), generování primitiv (trojúhelník, čtverec, n-úhelník, kruh) a vytvoření ekvidistance.

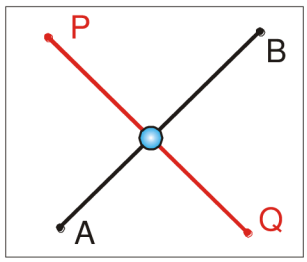
4.2.3 Detekce kolizí

Bod-úsečka – při hledání této kolize postačí dosadit bod do rovnice přímky určené úsečkou. Pokud vyjde nula a souřadnice bodu leží mezi souřadnicemi bodů A a B, tak bod leží na úsečce.



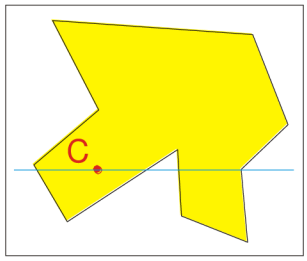
Obr 15. Kolize: bod a úsečka

Úsečka-úsečka – při hledání této kolize se nejprve určí, jestli úsečky nejsou rovnoběžné, pokud ano, je zapotřebí ověřit, jestli se překrývají. Pokud jsou úsečky různoběžné, lze určit jejich průsečík, který musí ležet na obou úsečkách.



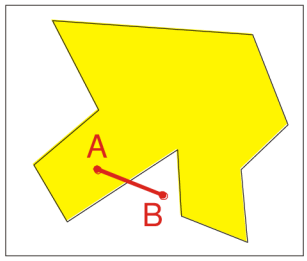
Obr 16. Kolize: úsečka a úsečka

Bod-polygon – na detekci této kolize se úspěšně aplikuje průsečíková metoda. Bod C rozděluje přímku (rovnoběžná s osou x) na dvě polopřímky, pro každou polopřímku se spočítá počet průsečíků s hranami polygonu. Pokud platí, že počet průsečíků je lichý na obou polopřímkách, leží bod C uvnitř. Pokud platí, že počet průsečíků je sudý na obou polopřímkách, leží bod C vně. V ostatních případech je nerozhodnuto a je nutno použít např. rozdělení přímkou rovnoběžnou s osou y.



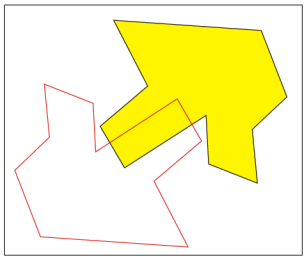
Obr 17. Kolize: bod a polygon

Úsečka-polygon – na detekci této kolize lze úspěšně aplikovat metodu úsečka-úsečka n - krát (n – počet stran polygonu) a ověřit průsečík úsečky s každou stranou polygonu.



Obr 18. Kolize: úsečka polygon

Polygon-Polygon – při hledání této kolize se n^2 - krát použije metoda úsečka-úsečka a ověří se vzájemná kolize mezi stranami polygonů.

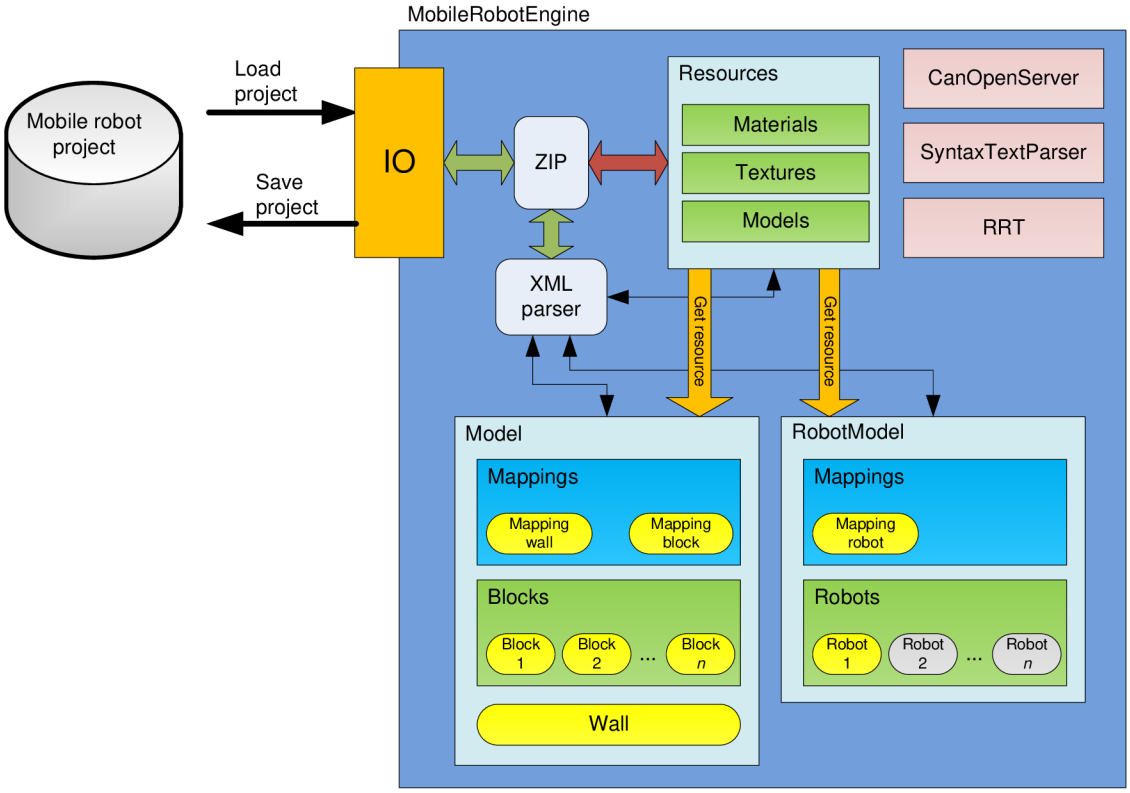


Obr 19. Kolize: polygon-polygon

4.3 Modul „Core“

4.3.1 Popis modulu

Modul *Core* je tvořen samostatnou knihovnou MobileRobot.Core.dll. Tento modul je ústředním členem celé aplikace. Je základem modelování a počítačové simulace robotu. Významným objektem je typ *MobileRobotEngine*, přes který je možno přistupovat k jakékoliv entitě projektu (*Mobile robot project*) v hierarchickém uspořádání. Struktura modulu je uvedena v příloze [A] kapitola 3.



Obr 20. Struktura objektu *MobileRobotEngine*.

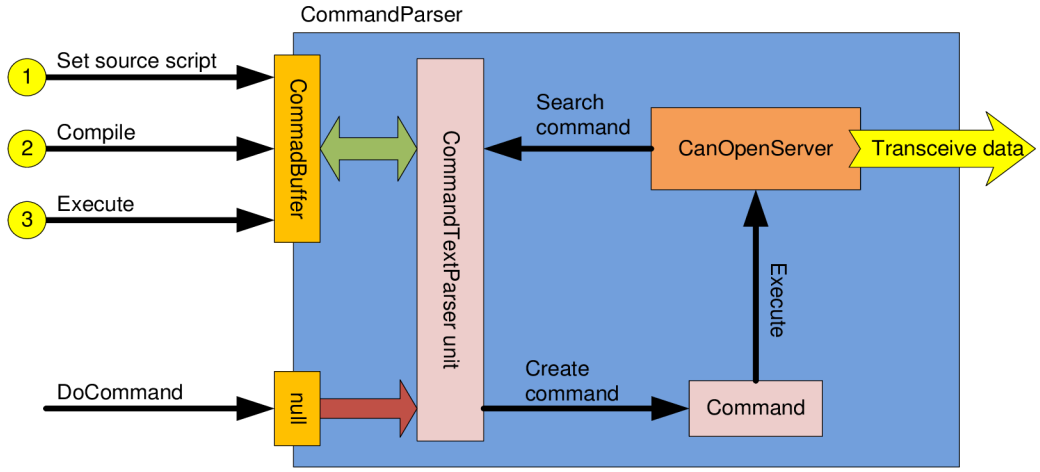
Práce s projektovým souborem probíhá pouze na úrovni metod *Load project* a *Save project*. Při volání metody *Load project* dochází k dekompresi projektu, který obsahuje několik XML souborů, které jsou poté zpracovány objektem *XML parser*, čímž jsou vytvořeny patřičné struktury. Obdobně při volání metody *Save project* dochází k postupnému vygenerovaní XML struktur a následně je provedena komprese do jednoho projektového souboru.

Projektový soubor může obsahovat i další vložené soubory, jako třeba bitmapy a trojrozměrné x-modely. Tyto soubory jsou zdrojem (*resources*) pro mapování objektu v 3D scéně.

Architektura umožňuje využití více robotů. V další části textu je použito modelování pouze s jedním robotem. Rovněž tak i způsob mapovaní (*mappings*) je uměle omezen pouze na 3 druhy (*wall*, *block*, *robot*).

4.3.2 CAN server a textový parser příkazů

CAN server neboli *CanOpenServer* je výkonnou jednotkou pro odesílání a přijímaní zpráv prostřednictvím linky CAN. Tento server je popsán v předchozí kapitole. Server je v tomto modulu doplněn o textový parser příkazů (*CommandParser*), který rozšiřuje využití CAN server o univerzální textové vstupní rozhraní. Struktura CAN serveru je uvedena v kapitole 4.1.



Obr 21. Struktura objektu *CommandParser* a princip spuštění příkazu

Spuštění příkazu v CAN serveru je možné třemi způsoby:

- Přímé zavolání příslušné metody v objektu CAN serveru – nepraktické!
- Provedení příkazu *DoCommand* v objektu *CommandParser* – pro textový příkaz bude vyhledána příslušná metoda (*Search command*), vytvoří se příkaz (*Create command*), ihned se provede (*Execute*) a data se odešlou (*Transceive data*).
- Nastavením skriptu, jeho kompilací a následným spuštěním, ukázka skriptu je uvedena v příloze A v kapitole 10.
    - *Set source script* – připraví zdrojová data (textové příkazy) pro vytvoření příkazů. Při tomto kroku se provede odstranění komentářů.
    - *Compile* – pro každý textový příkaz se nalezne odpovídající metoda v CAN serveru (*Search command*) a nastaví se parametry pro spuštění, pokud není metoda nalezena, *CommandParser* zahlásí chybu kompilace na příslušném příkazu.

- *Execute* – pokud předchozí krok dopadl bez chyby, je možné všechny příkazy spustit a tím dojde k přenosu dat (*Transceive data*).

Druhy příkazů odpovídají povaze struktury CAN serveru, dělí se na tyto:

- Příkazy CAN: *Open*, *Init*, *Close*, *Sleep*, *Send*.
- Příkazy NMT: *EnterPreOperational*, *ResetCommunication*, *StartRemoteNode*, *StopRemoteNode*, *ResetNode*.
- Příkazy PDO: příkazy dle dané implementace.
- Příkazy SDO: příkazy dle dané implementace.
- Příkazy LMT: příkazy dle dané implementace.

4.3.3 Generování RRT a optimalizace trajektorie RRPC

RRT rozhraní je důležitým prvkem při navrhování cesty robotu. Implementuje v sobě modifikovaný RRT algoritmus pro potřeby vygenerování vhodné trajektorie a také optimalizaci RRPC (viz kapitola 3), která cestu významně zjednoduší.

Při inIclAlizaci RRT rozhraní je načteno prostředí (překážky a stěna) a dále se načítá konfigurace robotu (tvar, počáteční a cílová pozice).

Generování RRT probíhá v nekonečné smyčce, dokud nenastane některý z případů přerušení cyklu. Následující modifikace algoritmu je zjednodušená od skutečné aplikace (jsou zde pouze důležité kroky).

Případy pro přerušení cyklu:

- cesta je nalezena, je splněno CHECK_END_CONF(c_{new})
- ruční zastavení $abort \leftarrow$ true
- vyčerpaní maximálního počtu iterací
- překročení maximálního času, ve kterém se měla cesta nalézt

Akceptovatelný je pouze výsledek, kdy je cesta nalezena. Jedná se o náhodné generování, takže každý výsledek hledání je originální.

```
BUILD_TREE()
   1. G.init();
   2. finded ← false;
   3. abort ← false;
   4. while !abort
   5.       p_rand ← GET_RANDOM_POINT();
   6.       c_nearest ← GET_NEAREST_CONF(p_rand);
   7.       c_new ← CREATE_NEW_CONF(c_nearest, p_rand);
   8.       if !IS_COLLISION(c_new) then
   9.            G.add_new_conf(c_new);
   10.           if CHECK_END_CONF(c_new) then
   11.                finded ← true
   12.                break;
   13.      if finded then
   14.           c_last ← G.last;
   15.           P.clear();
   16.           while c_last ≠ null
   17.                P.add(c_last)
   18.                c_last ← c_last.last;
   19.      return P;
```

4.3.4 Správa zdrojů

Správa zdrojů probíhá v objektu *ResourceLoader*. V objektu probíhá zpracování XML, čímž se vytvoří příslušné zdroje dat. V případě uvedení externího souboru je zapotřebí do něj přistupovat přes objekt *ZIP*. Tento přístup se týká pouze bitmap a x-modelů, materiály jsou vytvořeny pouze z XML. Ukázka XML souboru je uvedena v příloze [A] kapitola 9.3.

4.3.5 Popis modelu světa

Svět je popsán pomocí jednoho XML souboru, který je součástí projektu (*Mobile robot project*). Ukázka XML souboru je uvedena v příloze [A] kapitola 9.1.

4.3.6 Popis modelu robotu

Robot je popsán pomocí jednoho XML souboru, který je součástí projektu (*Mobile robot project*). Ukázka XML souboru je uvedena v příloze [A] kapitola 9.2.

4.4 Aplikace „Mobile robot studio“

4.4.1 Popis

Aplikace *Mobile robot studio* je simulační software pro navrhování trajektorie mobilních robotů. Simulace a výpočetní báze pohybu robotu je řešena 2D prostorem, zobrazení je již ve 3D za podpory rozhraní Microsoft DirectX. Pro plánování trajektorie pohybu robotu je využito RRT algoritmu a RRPC optimalizace (viz kap 3 a kap. 4.3.3).

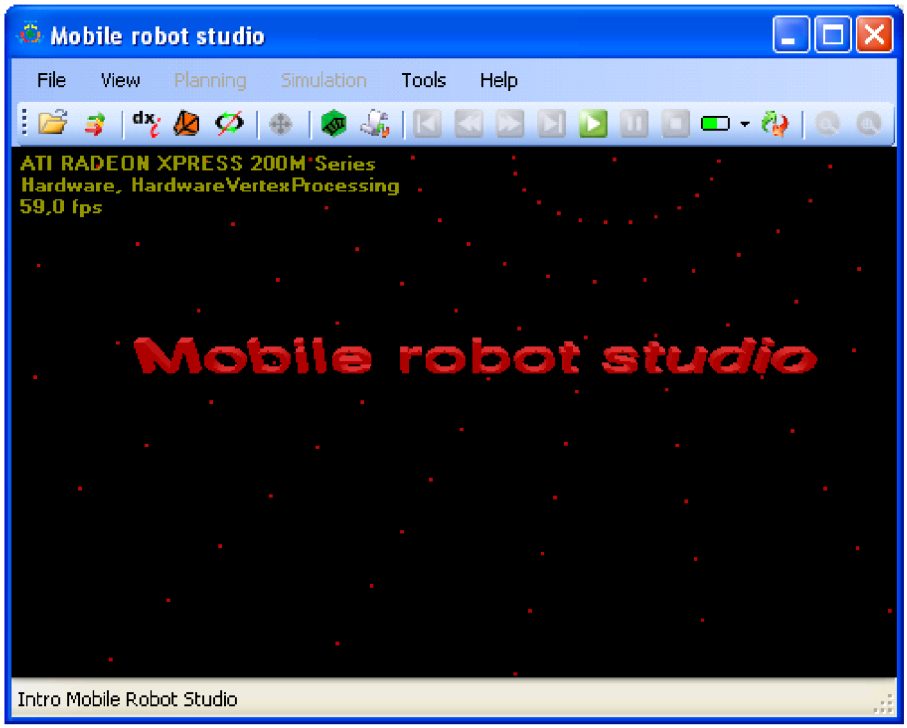
Aplikace je složena z několika modulů a knihoven. Objektová struktura aplikace je popsána v příloze [A] kapitole 1, 2, 3, 4, 5 a 6 dle příslušné programové knihovny.

Je-li trajektorie naplánováná, může se začít simulovat pohyb robotu za pomoci ovládacího panelu *SimulationControl*.

Součástí této aplikace jsou i dva editory:

- *World editor* – umožňuje editaci jednotlivých překážek a celé stěny.
- *Command editor* - editor příkazů, které je možno provést CAN serverem a tím přenést kinematický model ze simulačního prostředí na reálný stroj.

Při spuštění aplikace je ověřeno 3D zařízení, pokud je vše v pořádku, je zobrazeno *intro*.

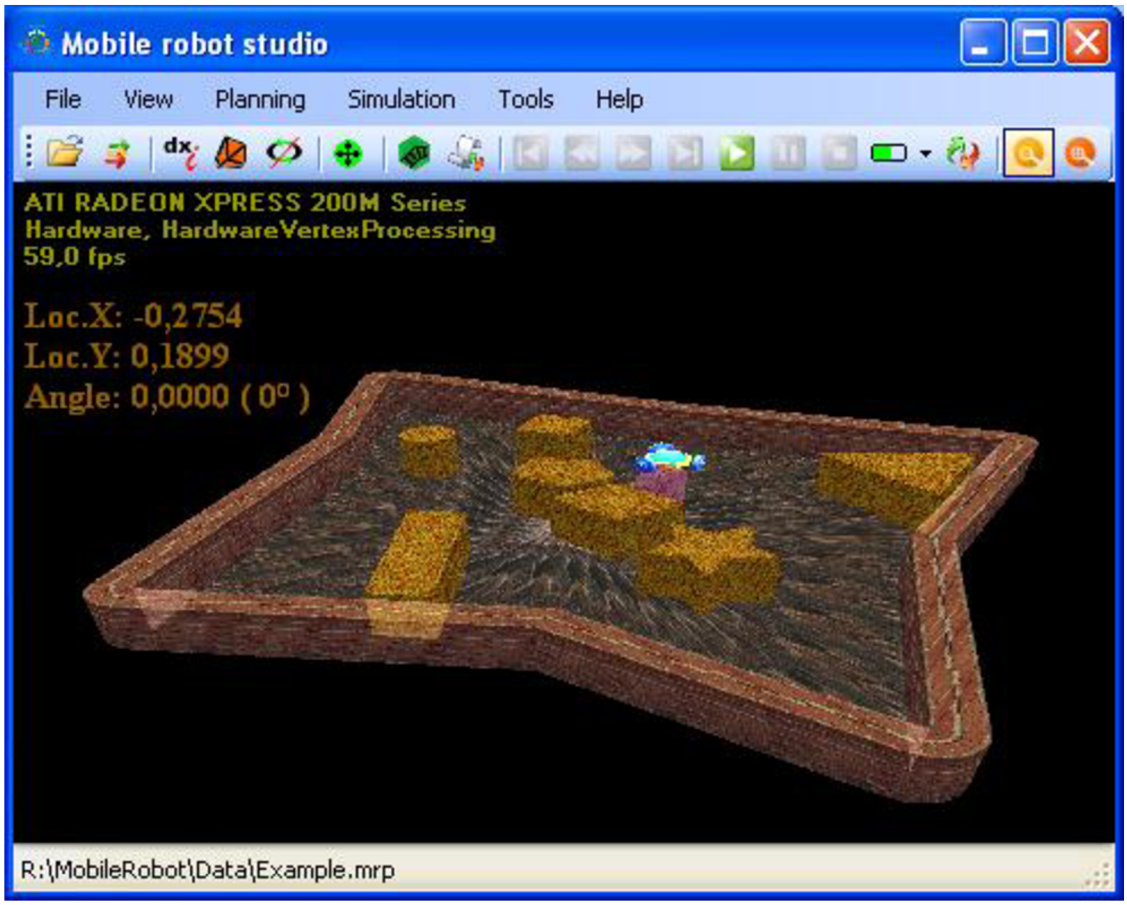


Obr 22. Hlavní okno aplikace „Mobile robot studio“

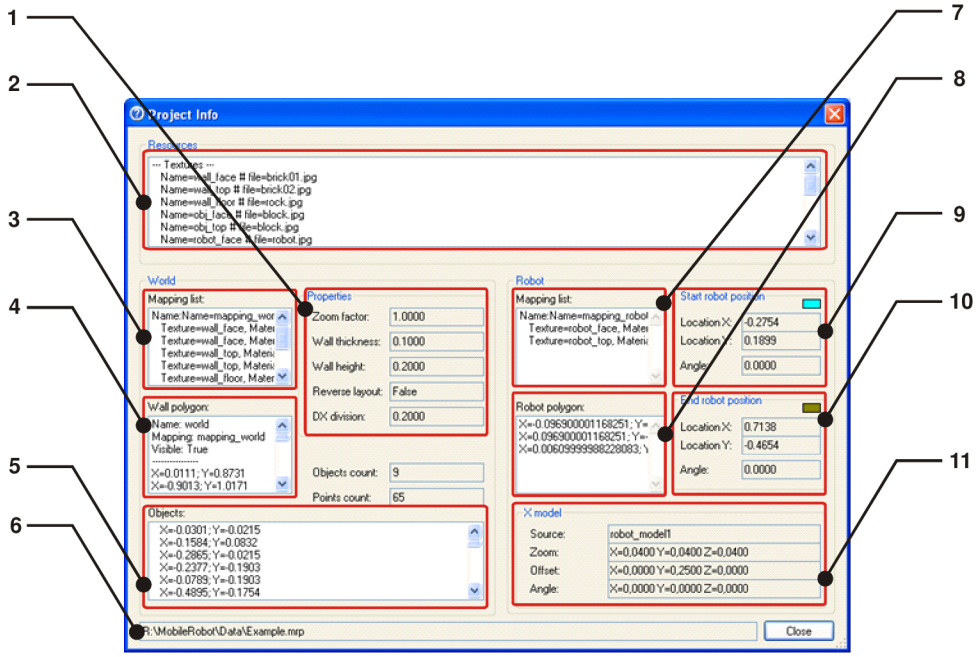
4.4.2 Otevření projektu

Otevřením projektu (*Mobile robot project*) je okamžitě inIclAlizováno jádro aplikace (*Core*) a tím jsou přístupné všechny entity v projektu. Po inIclAlizaci jádra je provedeno sestavení 3D scény a je zapnuta automatická rotace pohledu kamery. Tímto vznikne „vynikající“ pohled na celý svět, ve kterém bude probíhat simulace.

Jak projekt této aplikace vypadá, je uvedeno v příloze [A] kapitola 9. Stejně tak je i ukázkový projekt je vložen na CD – příloha [B].



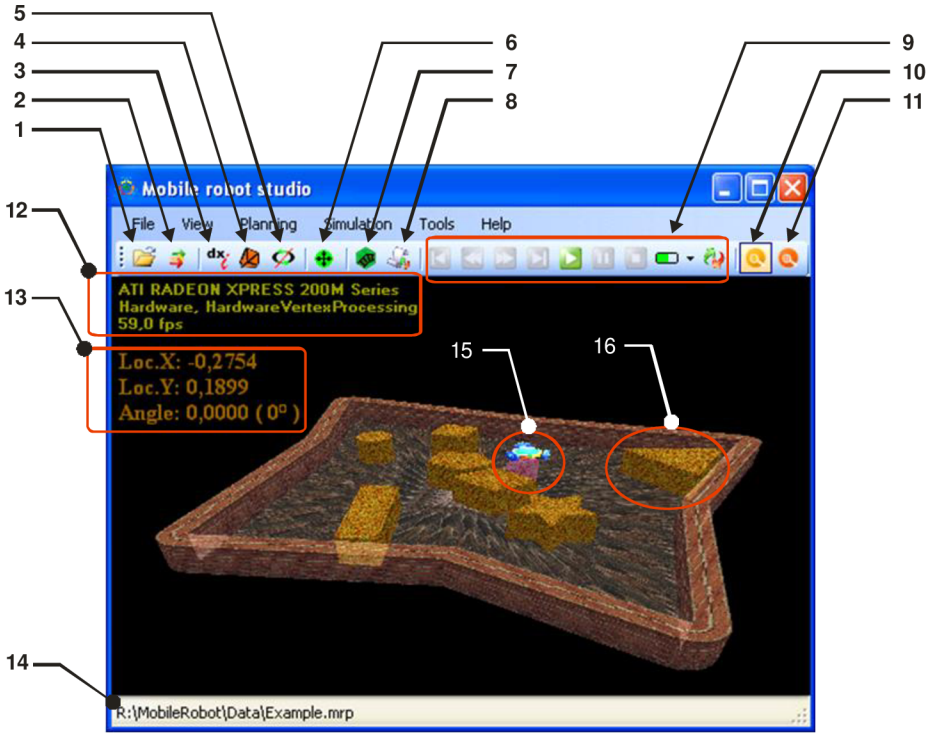
Obr 23. Otevřený projekt – zobrazený 3D model světa v hlavním okně



Obr 24. Informace o otevřeném projektu
1 – obecné parametry 3D scény
2 – načtené zdroje
3 – mapovaní objektu stěny a překážek
4 – polygon stěny
5 – polygony překážek
6 – cesta k souboru projektu
7 – mapování robotu
8 – polygon robotu
9 – startovní konfigurace robotu
10 – cílová konfigurace robotu
11 – x-model robotu

4.4.3 Popis hlavního okna a 3D zobrazení

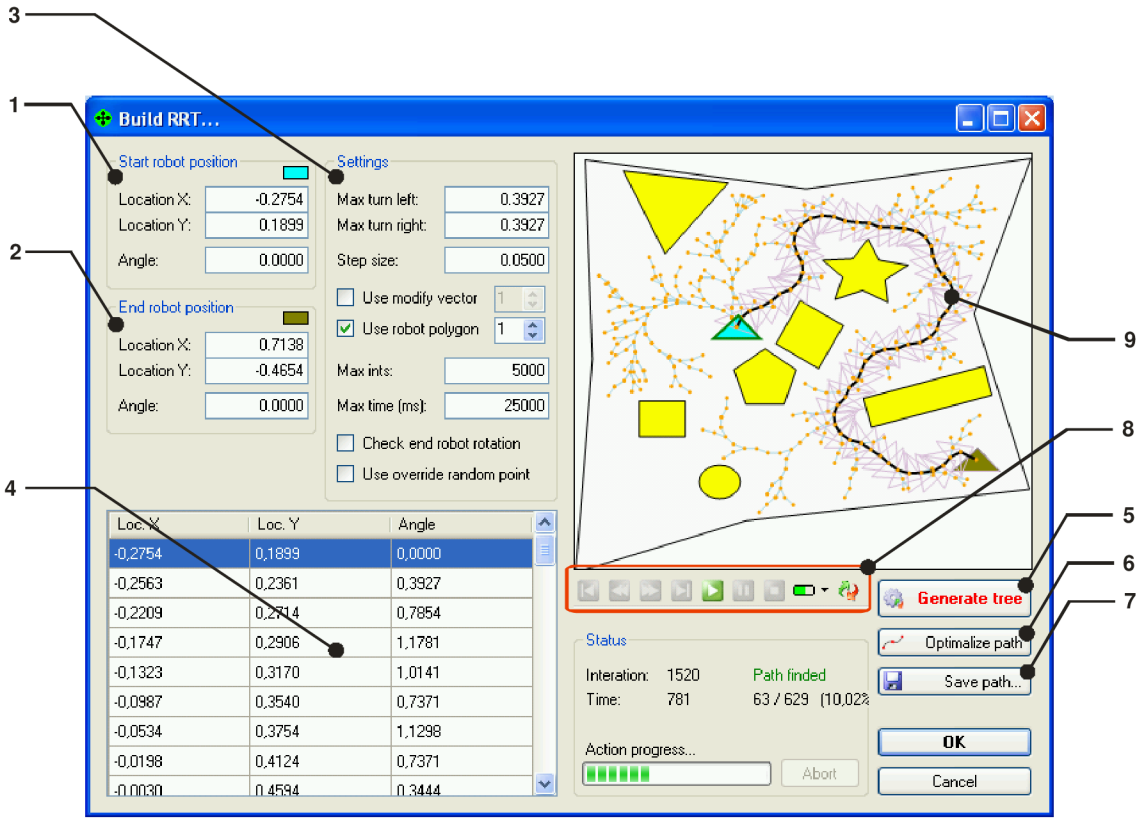
Hlavní okno slouží zejména k zobrazování 3D scény a k prezentaci simulace mobilního robotu. Z hlavního okna lze dále spustit modul plánování trajektorie (*Build RRT*), editaci světa (*World editor*) a příkazový editor (*Command editor*).



Obr 25. Popis ovládacích prvků v hlavním okně:
1 – otevření projektu
2 – obnovení 3D scény
3 – zobrazení informací o hardwaru
4 – drátový model
5 – vypnutí automatické rotace a přepnutí na ruční režim
6 – plánování trajektorie (*Build RRT*)
7 – editor světa (*World editor*)
8 – editor příkazů (*Command editor*)
9 – ovládání simulace (*SimulatonControl*)
10 – přiblížení scény
11 – oddálení scény
12 – informace o hardwaru
13 – informace o aktuální pozici robotu
14 – aktuálně otevřený projekt
15 – model robotu
16 – překážka

4.4.4 Plánování trajektorie

Plánování trajektorie probíhá v okně *Build RRT* (viz obrázek 26). Pro vytvoření RRT využívá upravený RRT algoritmus (uvedený v kapitole 4.3.3) a optimalizaci pomocí RRPC algoritmu (popsaný v kapitole 3.2).



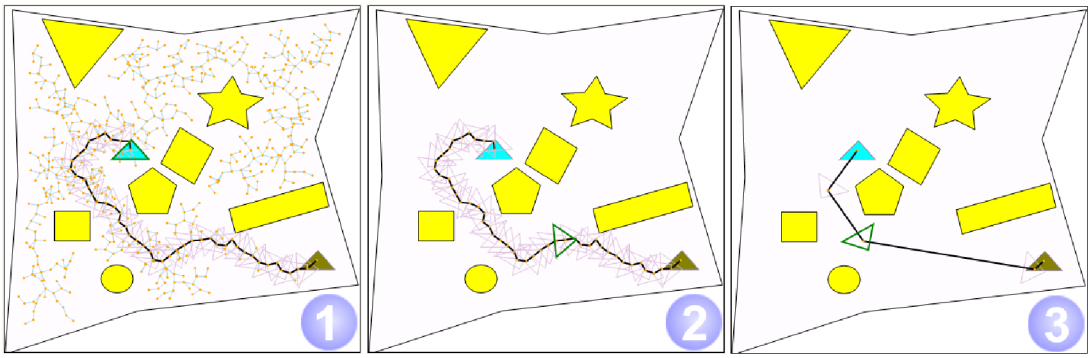
Obr 26. Popis okna *Build RRT*.
1 – startovní konfigurace robotu
2 – cílová konfigurace robotu
3 – nastavení inIclAlizačních parametrů pro generování RRT
4 – nalezená trajektorie – numerické vyjádření
5 – spustí vygenerování stromu
6 – optimalizuje nalezenou cestu
7 – uloží nalezenou cestu
8 – ovládaní simulace (*SimulationControl*)
9 – nalezená cesta – grafické vyjádření

Generování RRT probíhá s animací, kdy je vidět postupný růst stromu. Pokud je cesta nalezena, je možno tuto cestu optimalizovat (viz Obr 27. a Obr 28.). V tomto okně (viz Obr 26.) lze provést i jednoduchou simulaci, kdy je vidět, jak se zjednodušený model robotu pohybuje po definovaném světě. K vizuálnímu pohledu nechybí ani číselné vyjádření tabulkou. Při volbě určité pozice v tabulce je ihned ukázaná poloha robotu ve vizuálním pohledu.

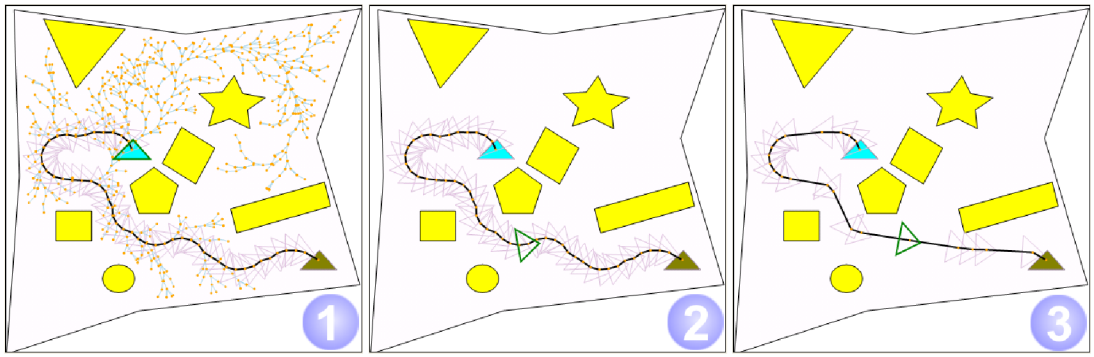
Obr 27. ukazuje vygenerování cesty a její optimalizaci pro všesměrového mobilního robota. Omezujícími podmínkou je pouze detekce kolizí mezi polygony. Obr 28. naproti tomu ukazuje vygenerování cesty a její optimalizaci pro robota s dvěma stupni volnosti (pohyb vpřed a rotace), kde je navíc ještě rotace omezené o určitý úhel.

Se vstupní podmínky velmi ovlivňují generovaní cesty a její optimalizaci. V prvním případě (viz Obr 27.) se vygenerovalo 44 konfigurací robotu a optimalizací se dosáhlo zmenšení na pouhých 5 konfigurací, to znamená úspěšnost RRPC algoritmu

89 %. V druhém případě se vygenerovalo 50 konfigurací a optimalizací se dosáhlo zmenšení na 23 konfigurací, to znamená s úspěšnost algoritmu RRPC 54 %.



Obr 27. Plánování trajektorie pro volný pohyb
1. vygenerování RRT a nalezená cesta
2. nalezená cesta (44 konfigurací robotu)
3. optimalizace nalezené cesty (5 konfigurací robotu)



Obr 28. Plánování trajektorie s omezujícími podmínkami
1. vygenerování RRT a nalezená cesta
2. nalezená cesta (50 konfigurací robotu)
3. optimalizace nalezené cesty (23 konfigurací robotu)

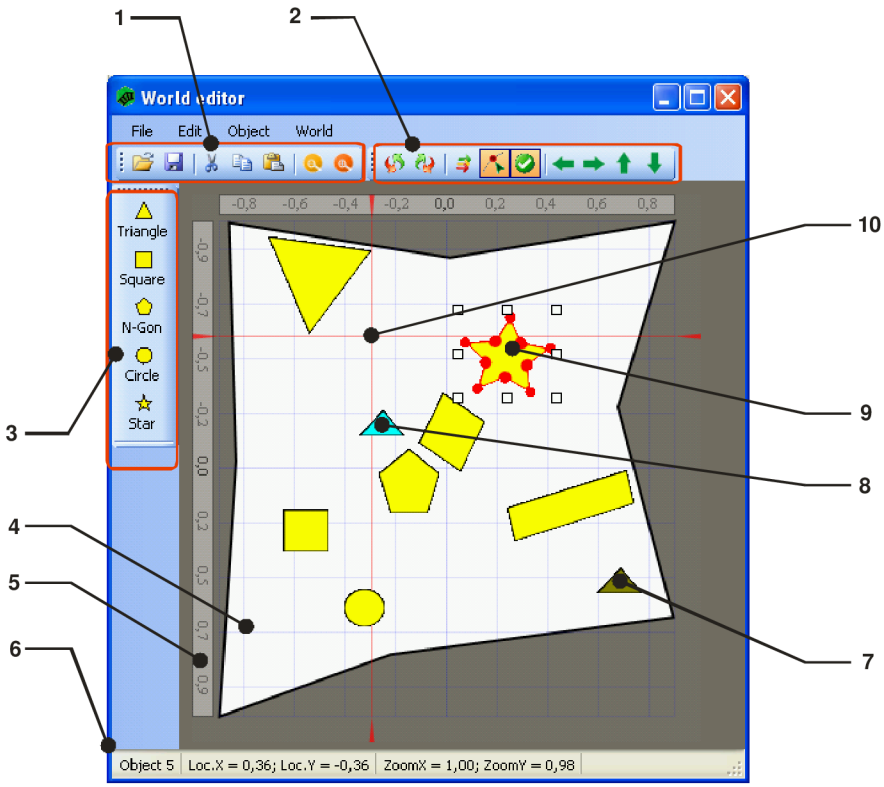
Naplánovanou trajektorii lze uložit do XML souboru, která může být dále zpracována. Ukázka XML souboru je uvedena v příloze [A] kapitola 12.1.

4.4.5 Editor světa

Editor světa (*World Editor*, viz Obr. 29) slouží ke snadné editaci zdi a překážek. Editovat lze na úrovni celých objektů (posun, rotace a zvětšení), nebo na úrovni bodu, kdy lze měnit pozici jednotlivých bodů polygonu, stejně tak lze i body přidávat a odebírat. Rozměry všech objektů se ukazují v reálných souřadnicích, aby vznikla snadná představa o poloze a rozměru objektu. K lepší orientaci při návrhu je okno doplněno o pravítka a nitkový kříž.

Další důležitou vlastností editace je stanovení zjednodušeného tvaru robotu a nastavení počáteční a cílové pozice ve světě. Počáteční a cílová pozice musí ležet uvnitř polygonu stěny (ohraničené simulační prostředí).

Součásti editoru je také možnost vložení nových překážek ve tvaru jednoduchých polygonů (trojúhelník, čtverec, pravidelný n-úhelník, kruh a hvězda). Program umí počítat vedle konvexních polygonů i polygony nekonvexní, což je velmi příjemné při návrhu, kdy se objekty nemusí skládat k sobě.

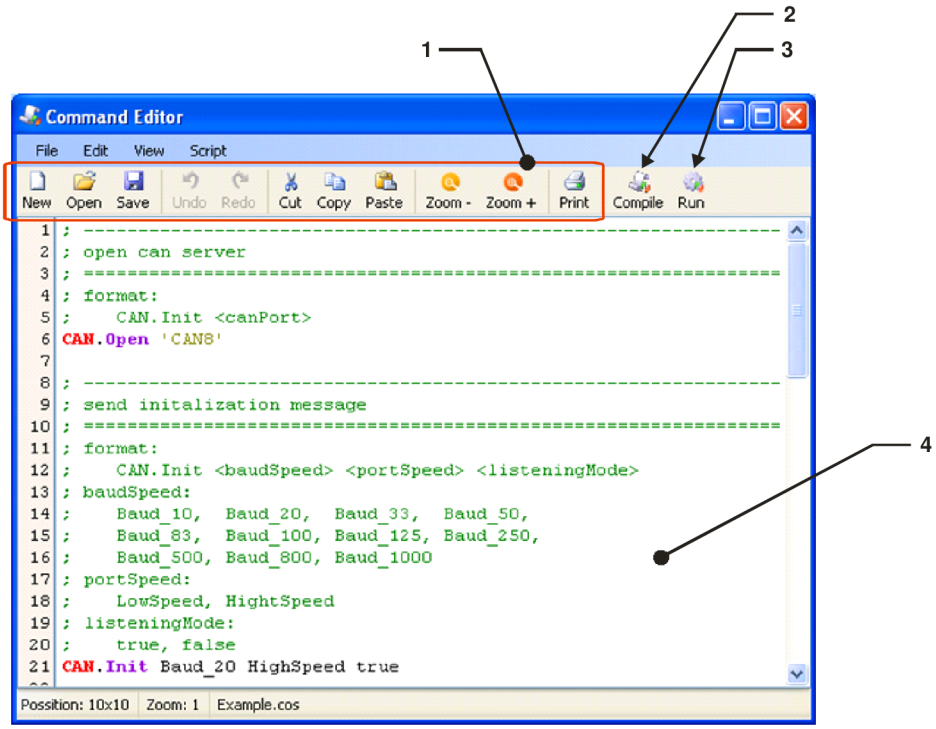


Obr 29. Popis okna *World editor*
1 – hlavní nástroje
2 – transformace objektu
3 – vložení nového tvaru
4 – simulační oblast
5 – pravítko
6 – informace o objektu
7 – cílová pozice robotu
8 – počáteční pozice robotu
9 – označený objekt - možno editovat tvar, upravovat body
10 – nitkový kříž

Po dokončení editace je možno úpravy uložit. Aby byly úpravy aktuální i pro 3D scénu v hlavním okně je zapotřebí, buď znovu otevřít projekt, nebo zvolit příkaz pro obnovení 3D scény.

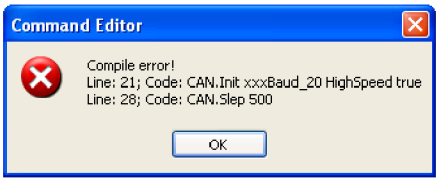
4.4.6 Editor příkazů

Editor příkazu (*Command Editor*) slouží zejména k editaci skriptu (*CANopen script*), ve kterém je zvýrazňována syntaxe příkazů (viz Obr. 30). Dále je možné provést kompilaci a spuštění celého skriptu. Více o procesu kompilace a spuštění skriptu pojednává kapitola 4.3.2. V příloze [A] kapitola 10 je uvedena ukázka skriptu s popisem možných příkazů a jejich správná syntaxe.



Obr 30. Editor příkazů (*Command editor*)
1 - standardní tlačítka pro úpravu
2 – zkompiluje otevřený skript (*Compile*)
3 – spustí otevřený skript (*Run*)
4 – editační okno

Spuštěním kompilace (*Compile*) je ověřena syntaxe skriptu. Pokud proběhne kompilace bez chyby, je vytvořena kolekce příkazů, které jsou připraveny ke zpracování CAN serverem. Při nalezení chyb editor o této skutečnosti náležitě informuje.

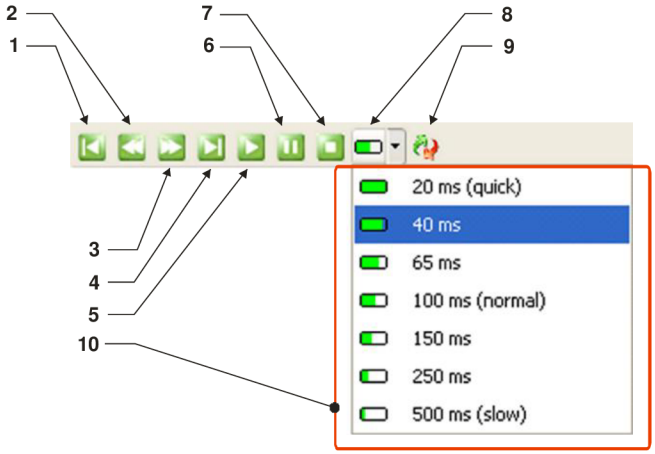


Obr 31. Chyby vzniklé při kompilaci, chyba syntaxe na řádku 21 a 28

4.4.7 Simulace

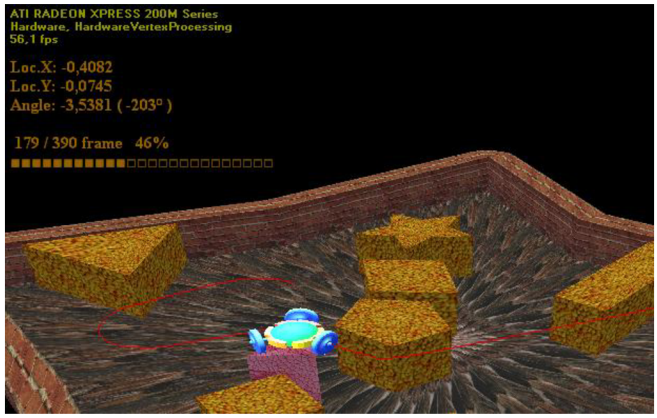
Simulaci je možné provést kdykoliv po nalezení cesty. Simulaci je možné provést buď v okně s plánováním trajektorie (*Build RRT*) anebo v hlavním okně aplikace, kde je zobrazena simulace pohybu mobilního robotu ve 3D projekci.

Důležitým prvkem simulace je ovládací panel (*SimulationControl*, viz Obr. 32), který dokáže simulaci spustit, zastavit a popřípadě i krokovat.



Obr 32. Ovládání simulace (*SimulationControl*)
1 – první snímek (*first*)
2 – předchozí snímek (*previous*)
3 – další snímek (*next*)
4 – poslední snímek (*last*)
5 – spuštění simulace (*run*)
6 – pozastavení simulace (*pause*)
7 – zastavení simulace (*stop*)
8 – volba rychlosti přehrání simulace (*speed*)
9 – opakování simulace (*repeat*)
10 – výběr rychlosti simulace (hodnota udává pauzu mezi snímky)

Při spuštění simulace v hlavním okně je zobrazena trajektorie mobilního robotu (červená křivka) a robot je uveden do pohybu dle nastavené rychlosti. Aktuální pozice a natočení robotu je zobrazováno v levém horním rohu 3D scény. Dále je zde také vidět postup simulace uvedený ve snímcích a procentech.

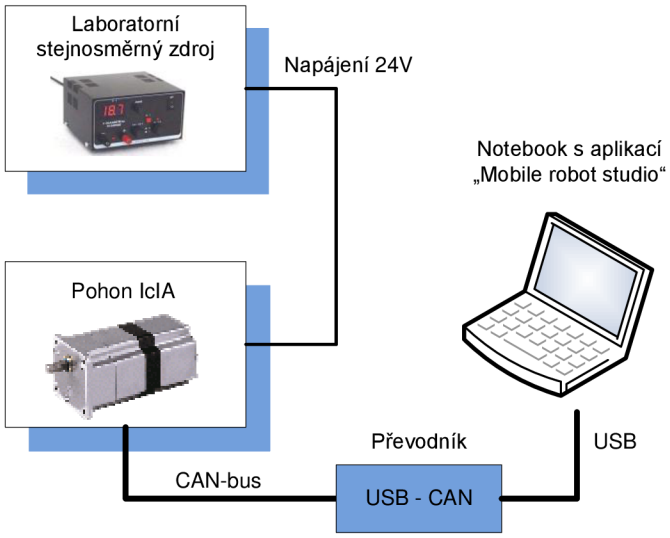


Obr 33. Průběh simulace pohybu mobilního robotu ve 3D pohledu

5 Test aplikace

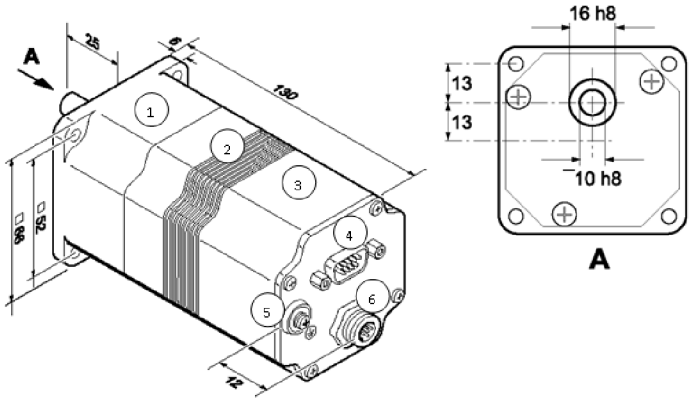
Zkušební zapojení je realizováno pouze s jedním pohonem (regulační pohon IclA), protože v době dokončení diplomové práce ještě nebyl zkušební mobilní robot zcela dokončen.

	Zařízení nebo jiná součástka	Typ
1	Regulační pohon IclA	POSITEC IclA D065
2	Digitální altimetr	Typ M890G
3	Laboratorní zdroj	DIAMETRAL R124R50E
4	Notebook	ASUS A6R
5	USB – CAN převodník	FTDI FT245RL
7	Datový kabel CAN-bus	



Obr 34. Schéma zapojení pohonu IclA s notebookem

Pohon IclA je napájen ze zdroje napětím 24V a je propojen s notebookem, přes převodník USB – CAN. Simulační software „Mobile robot studio“ a ukázkový projekt je dostupný v příloze [B].



Obr 35. Schéma regulačního pohonu IclA
1 - čelní převodovka
2 - stejnosměrný motor
3 - snímač polohy, výkonová a řídící elektronika
4 - konektor pro napájecí napětí, manuální operace a nouzové vypnutí
5 - konektor pro ochranný vodič (zem)
6 – CAN-bus konektor

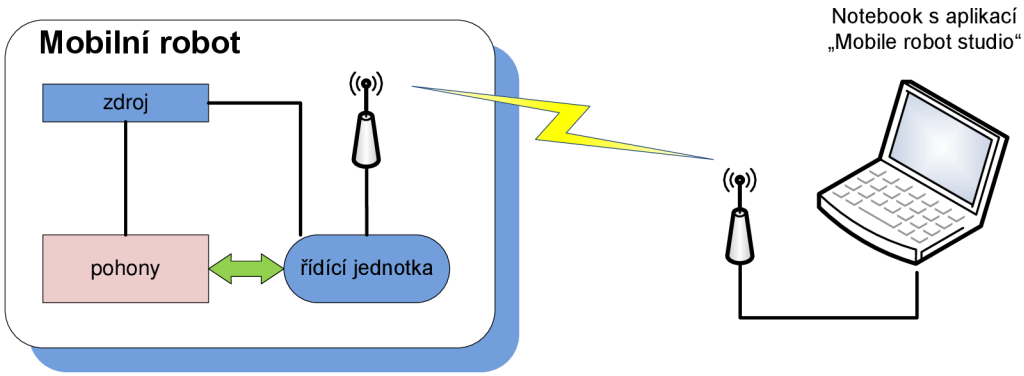
Po propojení regulačního pohonu IclA s notebookem, byla na notebooku spuštěna aplikace „Mobile robot studio“ a byl proveden test komunikace. Ukázka skriptu a výpis logu linky CAN je uveden v příloze [A] kapitola 11.

Obr 36. Propojení regulačního pohonu IclA s notebookem, na kterém je spuštěna aplikace „Mobile robot studio“

6 Budoucnost využití aplikace pro řízení mobilního robotu

V aplikaci Mobile robot studio se naplánuje optimální cesta, která bude bezdrátově přenesena do řídící jednotky mobilního robotu. Řídící jednotka bude ovládat jednotlivé pohony. Mobilní robot se může pomocí řídící jednotky pohybovat autonomně.

Dalším vylepšením bude přidání senzoriky, která bude tvořit zpětnou vazbu do aplikace. Tím bude možné reagovat na změny prostředí při plánování nové trajektorie pro pohyb mobilního robotu.



Obr 37. Schéma využití aplikace *Mobile robot studio* v řízení mobilního robotu

Obr 38. Vývojová fáze všesměrového mobilního robota

7 Závěr

Robotika je rozsáhlý vědní obor, který pravděpodobně nejvíce zasahuje do průmyslového, výzkumného a armádního prostředí. V prostředí, kde se uplatňují mobilní roboty, nemusí figurovat osoby.

V průmyslu se mobilní roboti nasazují ve výrobních halách, např. k dopravě materiálu, přímé výrobě, automatickému skladování apod. Častým důvodem nasazení robotů je prostředí nebezpečné nebo nevhodné pro člověka. V armádním prostředí je rovněž mobilních robotů hojně využíváno, ať už to jsou bezpilotní letouny, vozidla, která nemusí nikdo řídit, nebo třeba i roboti prozkoumávající minové pole. Díky mobilním robotům je velice podpořena i výzkumná činnost, protože díky nim je možno prozkoumat svět, který byl pro člověka běžně nedosažitelný. (příklad)

Protože mobilní robot není jenom nějaká sestava hardwaru, ale má i nějakou řídicí jednotku, která obsahuje řídící program, je zapotřebí tento program někde navrhnout. Tím se dostáváme k cíli této diplomové práce, která má vytvořit právě takový software, který bude plánovat trajektorii pro mobilního robota.

Navržený simulační software „Mobile robot studio“ je postaven na několika modulech, které lze využít i samostatně. Je to modul „CANopen“, který zajišťuje komunikaci s hardwarem. Dále je to modul „LinearAlgebra“, který obsahuje datové struktury a výpočty z oblasti lineární algebry. Jádrem celého softwaru je modul „Core“, který zajišťuje všechny potřebné výpočty pro plánování trajektorie. Plánování trajektorie je založeno na RRT algoritmu a RRPC optimalizaci. Simulační software je podrobněji popsán v kapitole 4.

V závěru této práce bych ještě zmínil, že navržený software je přednostně určen pro plánování trajektorie tříosého všesměrového mobilního robotu, který je umístěn na odboru Aplikované informatiky, ÚAI. Tento software bude dále rozvíjen a využíván pro potřeby řešitelů výzkumného záměru Inteligentní systémy v automatizaci, řešeného tímto pracovištěm.

8 Literatura

[1] LAVALLE, M. STEVEN: *Planning Algorithms*. Cambridge UP, 2006. 1007 s. ISBN 0-521-86205-1.

[2] TROELSEN, ANDREW: *C# a .NET 2.0 profesionálně*. ZONER Press, 2006. 1200 s. ISBN 80-86815-42-0.

[3] LAVALLE, M. STEVEN: *Planning Algorithms / Motion Planning*. [HTML dokument]. Dostupné z: http://planning.cs.uiuc.edu/

[4] LAVALLE, M. STEVEN: *The Rapidly-Exploring Random Tree (RRT) Page*. [HTML dokument]. Dostupné z: http://msl.cs.uiuc.edu/rrt/

[5] SIG POSITEC AUTOMATION, Lahr.: *IclA Field bus manual*.

[6] SIG POSITEC AUTOMATION, Lahr.: *IclA Device manual*.

[7] ROBOTIKA.CZ: *Bug algoritmy*. [HTML dokument]. Dostupné z http://robotika.cz/guide/bug-alg/cs

[8] KUFFNER, JAMES: *Rapidly-Exploring Random Trees (RRTs) Algorithm*. [HTML dokument]. Dostupné z: http://www.kuffner.org/james/plan/algorithm.php

[9] KIM, JONGWOO; ESPOSITO, M. JOEL: *Adaptive Sample Bias for Rapidly-exploring, Random Trees with Applications to Test, Generation*. [PDF dokument]. Dostupné z: http://www.usna.edu/Users/weapsys/esposito/auxFiles/KE_ACC05.pdf

[10] KIM, JONGWOO; ESPOSITO, M. JOEL; KUMAR, VIJAY: *An RRT-Based Algorithm for Testing and Validating Multi-Robot Controllers*. [PDF dokument]. Dostupné z: http://www.roboticsproceedings.org/rss01/p33.pdf

[11] STANEK, JASON; MARKS, MATTHEW: T*he Stanek Marks Anticipating Rapidly Exploring Random Tree (The SMARRT Algorithm)*. [HTML dokument]. Dostupné z: http://msl.cs.uiuc.edu/~lavalle/cs576_2000/StanekMarks/final.html.

[12] IMFSOFT: *USB-CAN Adapter (high speed)*. [HTML dokument]. Dostupné z: http://www.imfsoft.com/hardware/produkty/usb-can-adapter-high.asp.

[13] IMFSOFT: *USB-CAN Adapter TRIPLE drivers V4.5*. [PDF dokument]. Dostupné z: http://www.imfsoft.com/hardware/produkty/usb-can-adapter/download/cz/usb-can-adapter.pdf.

[14] VOJÁČEK, ANTONÍN: *Co je CANopen a jak na něj*. [HTML dokument]. Dostupné z: http://automatizace.hw.cz/mereni-a-regulace/ART231-co-je-canopen-a-jak-na-nej.html

[15] POKORNÝ, JAN: *Detekce kolizí v DirectX (collision detection)*. [HTML dokument]. Dostupné z: http://programovani.net-mag.cz/?action=art&num=459

[16] LIN, C. MING: *Ecient Collision Detection for Animation and Robotics* . [PDF dokument]. Dostupné z: ftp://ftp.cs.unc.edu/pub/users/manocha/PAPERS/COLLISION/thesis.pdf

[17] PELIKÁN, JOSEF: *Detekce kolizí v 3D*. [PowerPoint prezentace]. Dostupné z: http://cgg.ms.mff.cuni.cz/~pepca/lectures/pdf/collisions.pdf

[18] CODERSOURCE.NET: *DirectX Programming in C#*. [HTML dokument]. Dostupné z: http://www.codersource.net/published/view/328/directx_programming_in.aspx

[19] STÅLBERG, MAGNUS; STRÖMBLAD, HENRIK: U*sing Managed DirectX for Visualizing Process Data in 3D*. [PDF dokument]. Dostupné z: http://www.idt.mdh.se/utbildning/exjobb/files/TR0319.pdf

[20] SUNDAY, DAN: *Intersection of a Segment with a Convex Polygon or Polyhedron*. [HTML dokument]. Dostupné z: http://geometryalgorithms.com/Archive/algorithm_0111/algorithm_0111.htm

9 Seznam příloh

[A] KOCH, ZDENĚK: *Objektový model a struktura souborů projektu Mobile Robot*, Brno 2008. 34 s.

[B] KOCH, ZDENĚK: *Koch_DP_2008* [CD-ROM]. Brno 2008.