Řízení robota MINDSTORMS NXT pomocí PC

PC Controlling of MINDSTORMS NXT robot

Bc. Michal Procházka

Diplomová práce
2008

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2007/2008

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Michal PROCHÁZKA**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Řízení robota MINDSTORMS NXT pomocí PC**

Zásady pro vypracování:

Práce se zabývá tvorbou vhodného sw rozhraní pro řízení robota MINDSTORMS NXT. Ovládací sw by měl být schopen programovat a ovládat robora jak přes kabel USB, tak přes rozhraní bluetooth. Programovací práce budou prováděny ve vhodném programovacím prostředí.

1. **Vypracujte literární rešerši na téma využití NXT robotů a možnostech jejich programování.**
2. **Analyzujte současný stav programování NXT robotů.**
3. **Navrhněte způsob tvorby nových knihoven pro NXT robota.**
4. **Zvolený způsob realizujte a aplikujte na vybranou konfiguraci NXT robota.**
5. **Demonstrujte a vypracujte závěr.**

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **FERRARI, Mario, FERRARI, Guilio. Building Robots with LEGO Mindstorms NXT. David Astolfo. [s.l.] : Syngress, 2007. 448 s. ISBN 1597491527.**
2. **HANSEN, John C. LEGO Mindstorms NXT Power Programming : Robotics in C. [s.l.] : Variant Press, 2007. 560 s. ISBN 0973864923.**
3. **BAGNALL, Brian. Maximum Lego NXT : Building Robots with Java Brains. [s.l.] : Variant Press, 2007. 524 s. ISBN 0973864915.**
4. **BOOGAARTS, Martijn, et al. The LEGO MINDSTORMS NXT Idea Book : Design, Invent, and Build. [s.l.] : No Starch Press, 2007. 350 s. ISBN 1593271506.**

Vedoucí diplomové práce: **doc. Ing. Ivan Zelinka, Ph.D.**
Ústav aplikované informatiky

Datum zadání diplomové práce: **20. února 2008**

Termín odevzdání diplomové práce: **19. května 2008**

Ve Zlíně dne 20. února 2008

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Abstrakt česky

Práce se zabývá tvorbou vhodného SW rozhraní pro řízení robota MINDSTORMS NXT. Ovládací SW by měl být schopen programovat a ovládat robota jak přes kabel USB, tak přes rozhraní Bluetooth. Programovací práce budou prováděny ve vhodném programovacím prostředí.

Klíčová slova: robot, MINDSTORMS NXT, ovládání, programování, USB, Bluetooth, BricxCC, C#, Mathematica.

ABSTRACT

Dissertation is dealing with building of the suitable software interface for controlling of robot MINDSTORMS NXT. Control software should be able to program and control robot through USB cable and through Bluetooth Interface. Programming works will be done in suitable programming environment.

Keywords: robot, MINDSTORMS NXT, control, programming, USB, Bluetooth, BricxCC, C#, Mathematica.

Poděkování:

Členům rodiny děkuji za to, že investovali peníze do mého vzdělání, vedoucímu práce doc. Ing. Ivanu Zelinkovi Ph.D., kolegům především za jejich připomínky a programátorskou pomoc při řešení problému a Mgr. Němcové za gramatickou korekturu.

Motto:

Není podstatné jak dlouho žijete, ale jak!

Souhlasím s tím, že s výsledky této diplomové práce může být naloženo dle uvážení vedoucího diplomové práce. V případě publikace budu uveden jako spoluautor.

Na celé diplomové práci jsem pracoval samostatně a veškerou doporučenou literaturu jsem citoval.

Ve Zlíně dne 15. 05. 2008

.....

OBSAH

ÚVOD

Roboti ve všech svých podobách a tvarech musí být nějak řízeni, ať už lidmi nebo programem, který je vytvořen tak, aby robot plnil účel, pro který byl vytvořen. V posledních letech automatizace dosáhla velkých pokroků ve všech různých prostředích, ať už se jedná o automatizované výrobní linky ve výrobních halách těžkého průmyslu nebo o jednotlivé stroje na výrobu léků či jen parkovací automaty. Proto se již několik let zabývají jak jednotlivci tak velké firmy využitím umělé inteligence, mnohdy se jedná o jednoduché programy umožňující robotovi pohyb v definovaném prostoru. Dnes již jde především o vývoj inteligence s implementací různých algoritmů pro různé typy prostředí, předdefinovanými modelovými situacemi a simulací chování.

Tato práce se zabývá popsáním možností programování a řízení robota MINDSTORMS NXT. První část je zaměřena na popsání HW možností robota a shrnutí doposud používaných jazyků a prostředí pro programování s ukázkou prostředí BricxCC a Microsoft Visual C# 2008 Express Edition. V druhé části jsou uvedeny možné způsoby tvorby knihoven se zaměřením na jejich obsah s realizací dvou uvedených příkladů rozšíření stávajícího řešení a propojení prostředí Mathematica s knihovnou jazyka C# sloužící k řízení robota NXT. Obsahem jedné podkapitoly je také popis změn v konstrukci řídícího systému, dále jsou popsány problémy, které byly objeveny při tvoření této práce a postup jakým byly řešeny. Předposlední kapitola obsahuje finální řešení, které byly při této práci vyvinuty včetně popisu jednotlivých funkcí a metod s uvedením zdrojových kódů. V závěru jsou shrnuty nejdůležitější poznatky z této práce s krátkým nastíněním možného dalšího vývoje.

I. TEORETICKÁ ČÁST

1 VYUŽITÍ A MOŽNOSTI PROGRAMOVÁNÍ ROBOTŮ NXT

1.1 Možnosti NXT robotů

Roboti NXT jsou vyráběni jako stavebnice, která, tvoří dostupný prvek pro realizaci technicky zdatných jedinců toužících po náročnějších způsobech realizace, se kterými mohou rozvinout svoji kreativitu. Tyto stavebnice jsou také v mnoha případech využívány pro výuku a v zájmových kroužcích. Na rozdíl od konkurenčních výrobců, kteří vyrábějí roboty pevné konstrukce jsou roboti NXT plně modifikovatelní a ze stavebnice lze vytvořit téměř cokoli, co tvůrce napadne. Spolu se stavebnicí je dodáván jednoduchý návod pro sestavení prvního robota a originální software pro programování. Roboti jsou schopni plnit jednoduché i více složitější úlohy, vždy záleží na konstrukci a schopnosti naprogramovat odpovídající algoritmus. Nejčastější konstrukcí jsou různá vozidla s různým počtem kol či v kombinaci s pásy, humanoidi a napodobeniny strojů. Ve všech těchto robotech je využíváno čidel a motorů dodaných se stavebnicí nebo čidel vlastní výroby či koupených od jiných výrobců než je výrobce stavebnice.

Ke stavebnici dodávaný software umožňuje programování pomocí grafického prostředí s předpřipravenými funkčními bloky. Tyto bloky jsou rozděleny do tří palet a na každé paletě se nacházejí ve skupinách. Pomocí těchto bloků se vytváří program, který je schopen řídit tři motory a vyhodnocovat stavy čtyř senzorů. Závisí samozřejmě na člověku jaký typ robota a za jakým účelem postaví, každý typ má své využití, které je v některých případech i modifikovatelné.

Programování robota je v dodaném softwaru velmi jednoduché metodou „Drag and Drop“ vytvoříte z připravených bloků schéma které odpovídá požadované funkci programu. Jednotlivé bloky jsou samozřejmě modifikovatelné, čímž dosáhnete požadované funkce motoru, senzoru či řídící funkce. Tato práce se však programováním v jazyku NXT-G zabývat nebude, ale zaměří se spíše na textové orientované jazyky a větší práce bude také věnována řízení robota z prostředí Mathematica. Z dostupných jazyků, jejichž zkrácený výpis následuje, byly pro ukázku vybrány jazyk NXC v prostředí BricxCommandCenter a C# v prostředí Microsoft Studio C# 2008 Express Edition. Jazykem C# se budeme zabývat i později při rozšíření nalezené knihovny a řešení problémů vzniklých řízením robota z prostředí Mathematica.

Programovací prostředí:

- BricxCC
- OnBrick
- ICommand
- Microsoft Robotics Studio
- LabVIEW

Některé z těchto jazyků vyžadují také nahrazení originálního firmwaru, který je vytvořen pro programování pomocí dodaného SW od výrobce, firmwary jsou většinou od stejných výrobců jako dané prostředí nebo jsou ke stažení na stránkách výrobce NXT. Mezi dnes dostupnými jazyky nalezneme i jazyky speciálně vytvořené pro programování robotů NXT založené na některém z široce rozšířených jazyků jako je například C.

Jazyky:

- NBC (Next Byte Code)
- NXC (Not eXactly C)
- NXT
- NXT-G
- RobotC
- RUBY
- C#
- Java
- Python

Jak je vidět v seznamu se nachází klasické komerčně vyvíjené jazyky stejně jako OS jazyky vyvíjené komunitami lidí v jednotlivých projektech.

1.2 Hardwarové možnosti dané konstrukcí NXT

Všechny součástky stavebnice je kompatibilní s kostkami LEGO a také je lze kombinovat se stavebnicemi LEGO Technics. Všechny součástky nebo alespoň jejich povrch je tvořen plastem nebo gumou kromě standardních kostek jsou zde ozubená kola, kloubové spoje a osy. Nejpodstatnější součástí je však hlavní řídící jednotka („brick“ kostka) spolu s motory a čidly tvořící aktivní část každého robota. Komunikace mezi jednotlivými prvky a řídící jednotkou je realizována pomocí 6-ti žilových kabelů podobajících se standardním telefonním s tím rozdílem, že na koncovkách je posunut jistící kolík na stranu. Možnosti propojení s počítačem nebo jiným zařízením jsou dvě a to pomocí dodaného USB kabelu nebo pomocí Bluetooth. Ke komunikaci přes Bluetooth je na stránkách výrobce doporučeno několik zařízení různých typů a výrobců, které dnes již ale nejsou k dostání.

1.2.1 Technické specifikace

32-bit ARM7 microcontroller (Atmel, AT91SAM256, 50 Mhz)

256 Kbytes FLASH, 64 Kbytes RAM

8-bit AVR microcontroller

4 Kbytes FLASH, 512 Byte RAM

Bluetooth (Bluetooth Class II V2.0)

USB (12 Mbit/s)

Display 100 x 64 pixel LCD.

Reproduktor - 8 kHz.

Zvukový kanál s rozlišením 8-bitů a vzorkováním 2-16 KHz.

Napájení 6 AA baterií.

6-ti žilové kabely (bílá, černá, červená, zelená, žlutá, modrá)

1.2.2 Porty

4 vstupy, (jeden port IEC 61158 Type 4/EN 50 170 pro budoucí rozšíření)

3 výstupní

1.2.3 I/O zařízení

3x Motor (9V)

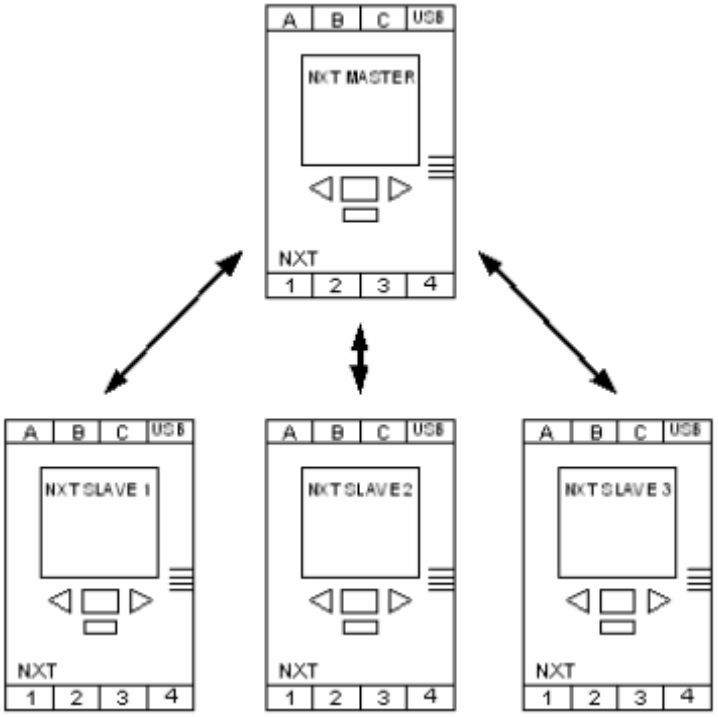
1x Dotykový senzor

1x Zvukový senzor

1x Světelný senzor

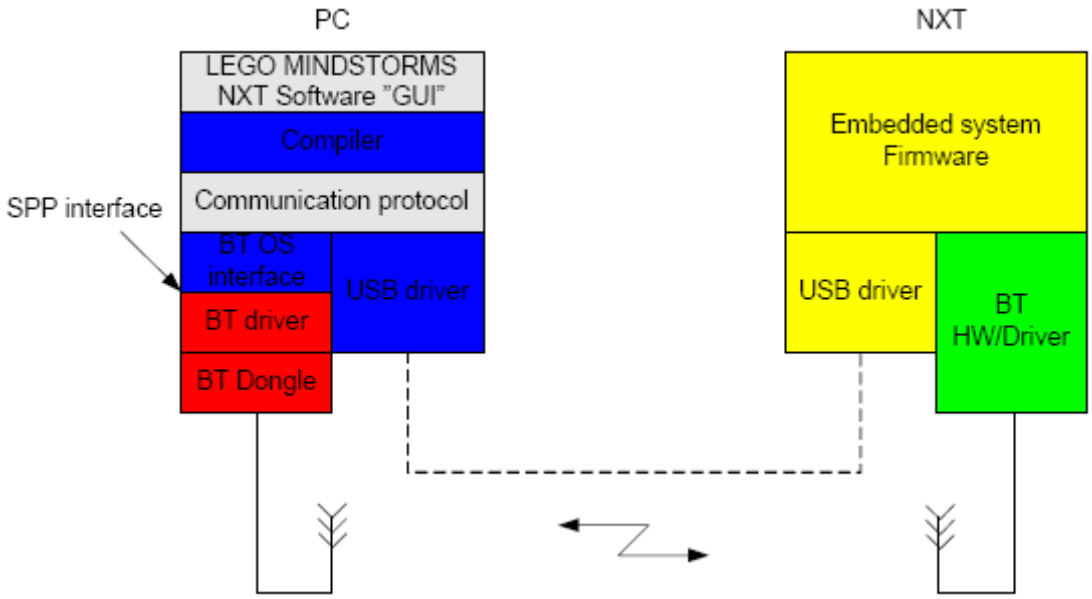
1x Ultrazvukový senzor (0 - 255 cm s přesností+/- 3 cm)

Další možností komunikace přes Bluetooth je možnost připojení až na tři zařízení současně v jednom okamžiku samozřejmě komunikují pouze dvě zařízení mezi sebou.



Obr. 1 - Schéma možností řízení přes Bluetooth

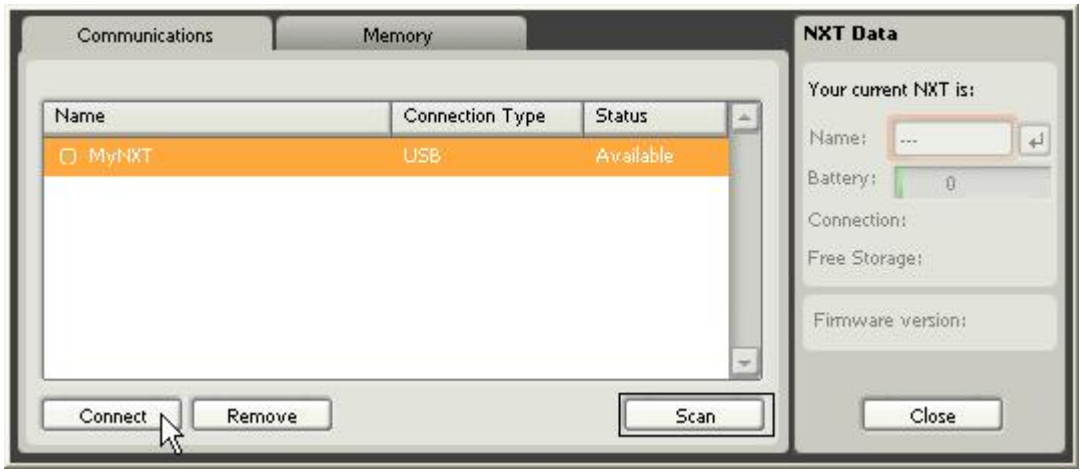
Znázornění komunikačních vrstev je podílejících se na komunikaci mezi zařízeními. Při použití jiného programovacího prostředí dojde v obrázku samozřejmě ke změně ale jednotlivé vrstvy zůstanou zachovány.



Obr. 2 - Vrstvy komunikace s NXT

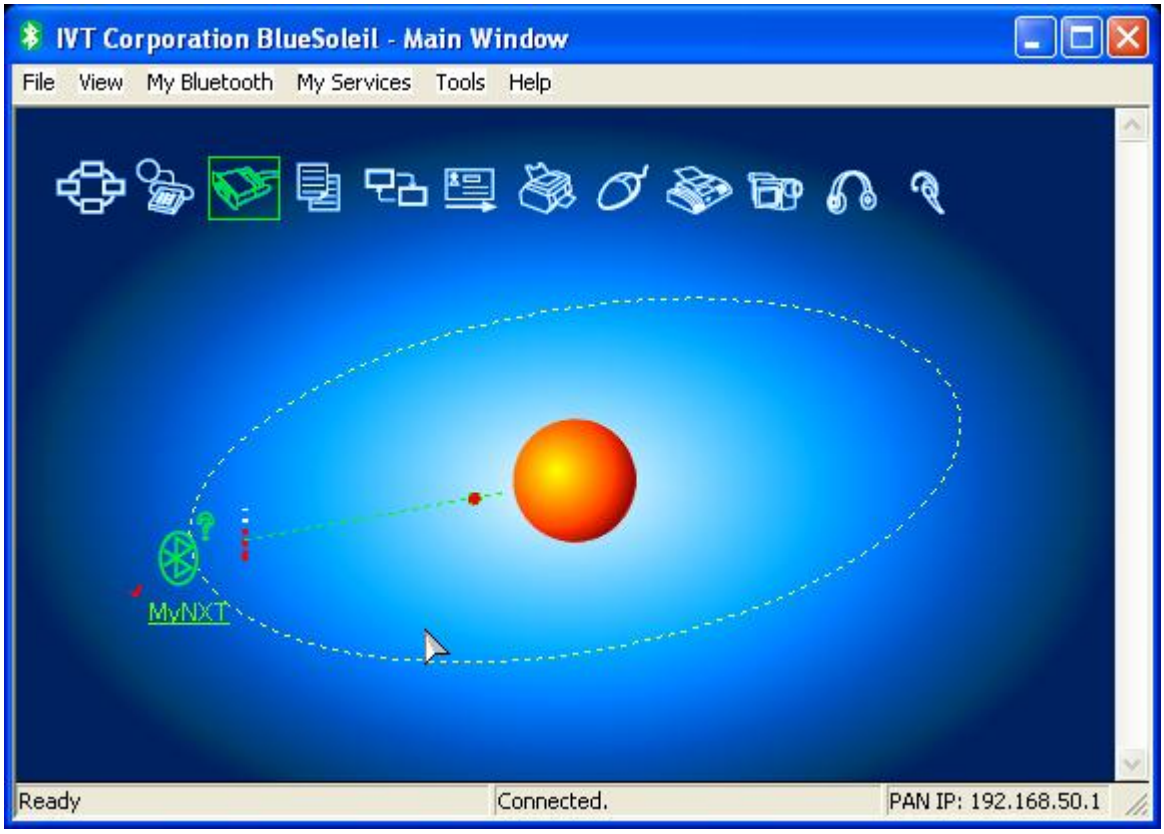
1.3 Komunikace NXT s PC

Komunikace pomocí dodávaného USB kabelu byla bezproblémová. Zařízení v systému sice nejde vidět, jako například klasický flash disk, ale po nainstalování programovacího prostředí a vyhledání zařízení viz Obr. 3 se lze na zařízení připojit. Obdobným způsobem se lze na robota připojit i z prostředí BricxCC, které nabízí dialog pro připojení při každém spuštěním prostředí jako první. Programovat lze samozřejmě i v režimu offline a po dokončení program na robota nahrát až ve finální verzi.



Obr. 3 - Ukázka dialogu pro nalezení a připojení NXT pomocí USB kabelu

Komunikace přes Bluetooth však byla z počátku velmi problémová a často vypadávala. Po neúspěšné snaze zajistit bluetooth zařízení, které bylo na stránkách výrobce označeno jako kompatibilní byl zakoupeno USB Bluetooth od firmy MSI, který byl nástupcem podporovaného zařízení. Po nainstalování SW který byl součástí balení a vyhledání dostupných zařízení viz Obr. 4 bylo provedeno spárování přístrojů za použití standardního klíče NXT a navázáno spojení.



Obr. 4 - Spojení NXT a PC

1.4 Možnosti programování NXT robotů

Možnosti programování se vždy odvíjejí od znalostí programátora a od zadání každého problému. Nejjednodušším způsobem je použití dodaného SW a pomocí grafické prostředí a několika bloků vytvořit celý program, který posléze nahrajeme na robota a spustíme pomocí ovládacích prvků na kostce. SW můžeme vytvořit programy různých složitostí, tímto programovacím prostředím se v této práci však zabývat nebudeme.

Jak je vidět na obrázku Obr. 5 robot kromě firmwaru, který není vidět jako soubor obsahuje i celou řadu modulů, které obsahují rutiny ovládacích či zobrazovacích prvků nebo zvukové stopy připravené k využití v programech vytvořených uživatelem.

Všechny prostředí které umožňují vývoj programů pro NXT musí po kompilaci vytvořit kompatibilní spustitelné programy a data v podporovaných formátech.

1.4.1 Podporované soubory pro řídící jednotku NXT

Firmware „soubor“.rfw

Uživatelem vytvořený program „soubor“.rxe

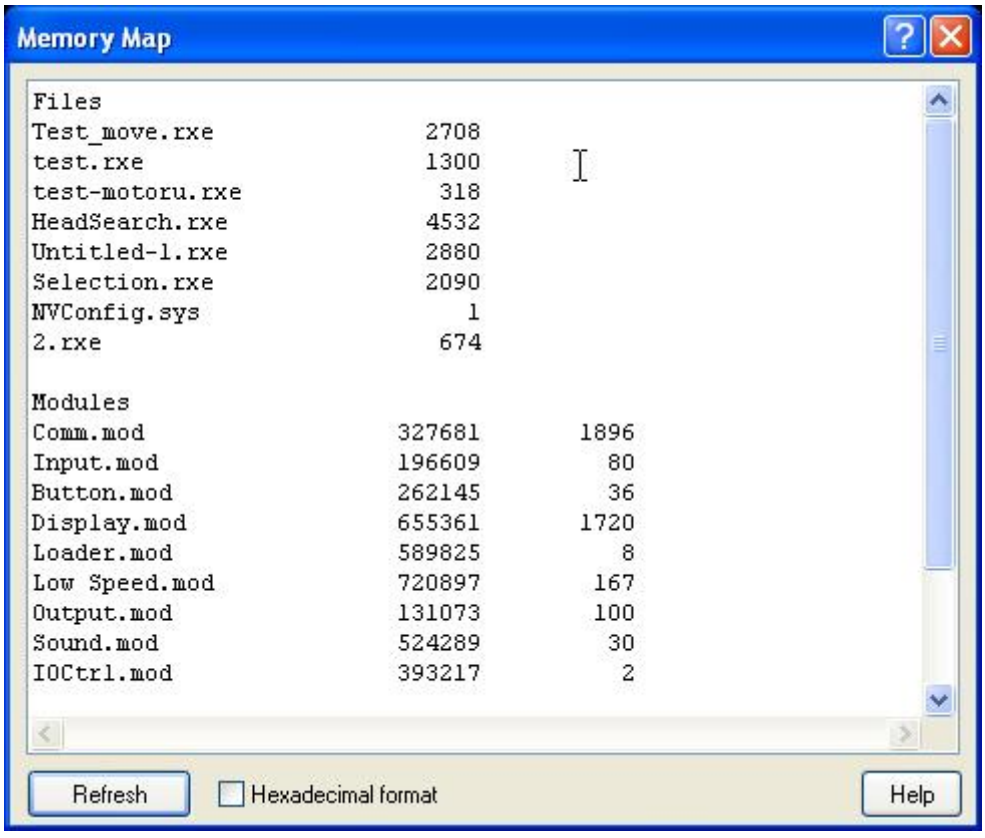
OnBrick program „soubor“.rpg

Předprogramované programy (Try-Me programs) „soubor“.rtm

Zvuky „soubor“.rso

Grafika „soubor“.ric

Datové log soubory „soubor“.rdt



Obr. 5 - Výpis paměti NXT

1.5 Způsoby řízení NXT

Z obecného pohledu existují dva způsoby řízení NXT robotů. Některé prostředí dovolují oba, jiné pouze jeden. Nezáleží však jen na zvoleném prostředí ale také na programovacím jazyku.

1.5.1 On-Board řízení

Po napsání programu a uložení do paměti robota je program připraven k použití. Ke spuštění dochází pomocí ovládacích prvků na těle robota. Můžete samozřejmě spustit jakýkoliv z dříve vytvořených programů, je však nutné dbát, aby se všechny spouštěné programy vztahovaly k momentální konstrukci robota a nedošlo tak k mechanickému poškození. Veškeré řízení v tomto případě probíhá v těle robota. Do této kategorie lze zařadit i kooperaci více robotů kteří si předávají informace, ale každý z nich má svůj řídící program.

1.5.2 Remote řízení

Tento způsob řízení je prováděn pomocí přímých řídících příkazů. Řídící jednotka je řízena jiným zařízením, které s ní komunikuje pomocí Bluetooth. Jako řídící zařízení zde může vystupovat téměř jakékoliv zařízení vybavené technologií Bluetooth jako jsou mobilní telefony, PDA, notebooky, PC a jiné. Více k tomuto typu řízení naleznete v praktické části této práce.

2 ANALÝZA SOUČASNÉHO STAVU PROGRAMOVÁNÍ NXT ROBOTŮ

Největší komunita zabývající se programováním robotů NXT se nachází na portálu [13] prezentují se zde nové věci, ať už se jedná o SW nebo HW. Je zde také několik odkazů na projekty a weby jednotlivců zabývající se roboty NXT. Za nejpřínosnější lze považovat fórum, na kterém lze najít řešené problémy, diskuze či výsledky testů různých uživatelů. Mezi členy patří jak začátečníci tak i profesionální programátoři stejně jako i autoři mnohých publikací o těchto robotech. Celý web je samozřejmě v angličtině, i když účastníci diskuzí jsou z celého světa.

2.1 Vývoj programování NXT

Programování NXT již není omezeno téměř ničím. Každým dnem vznikají nové projekty, kterých se účastní jak děti tak dospělí. Ať už se jedná třeba o nejznámější FLL nebo jen projekt několika nadšenců, vždy se jedná o napodobení nějakého známého mechanizmu či stroje, nebo spojení s některou z dnešních technologií jako je například GPS. Jako příklad používaných prostředí a jazyků jsou zde uvedeny dvě prostředí a dva jazyky vycházející z jazyka C pro úplnost je zde také uveden krátký seznam několika dostupných prostředí a jazyků.

2.1.1 Existující programovací prostředí a dostupné jazyky

Některé z těchto jazyků i prostředí byly vytvořeny za účelem programování robotů jiné byly pouze rozšířeny o další možnosti.

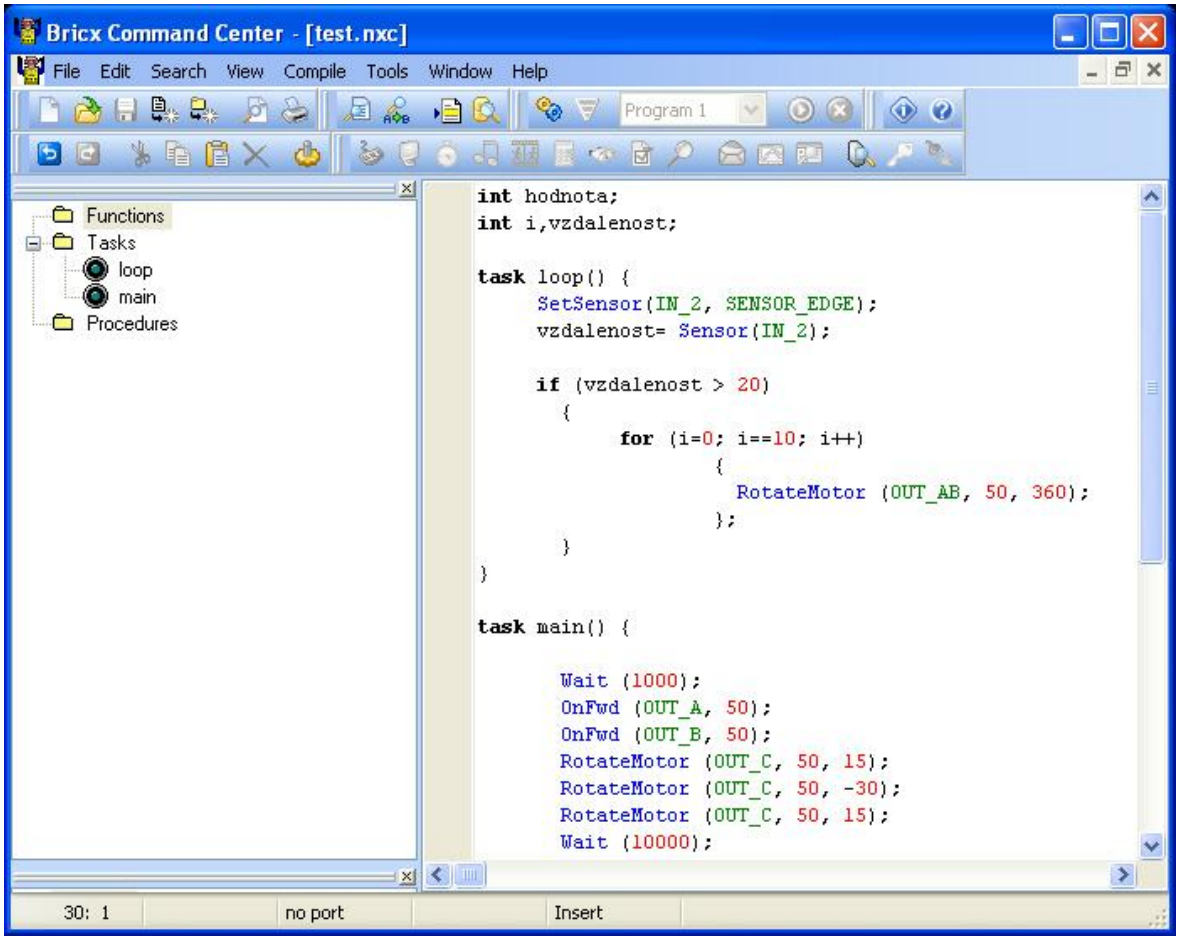
Jazyky: NBC (Next Byte Code), NXC (Not eXactly C), Robolab, Pblua, NXT-G, RobotC, RUBY, C#, LEJOS (Java).

Prostředí: OnBrick, ICommand, Microsoft Robotics Studio, LabVIEW

Podrobnější přehled prostředí a jejich vlastností naleznete v příloze P I.

2.1.2 Jazyk NXC v prostředí BricxCC

Jedná se o textový jazyk odvozený od jazyka C. Každý program je rozdělen na jednotlivé „tasky“ úkoly. Každý program musí obsahovat main task, který je vykonáván vždy po spuštění programu a dají se z něj volat jednotlivé další úkoly.

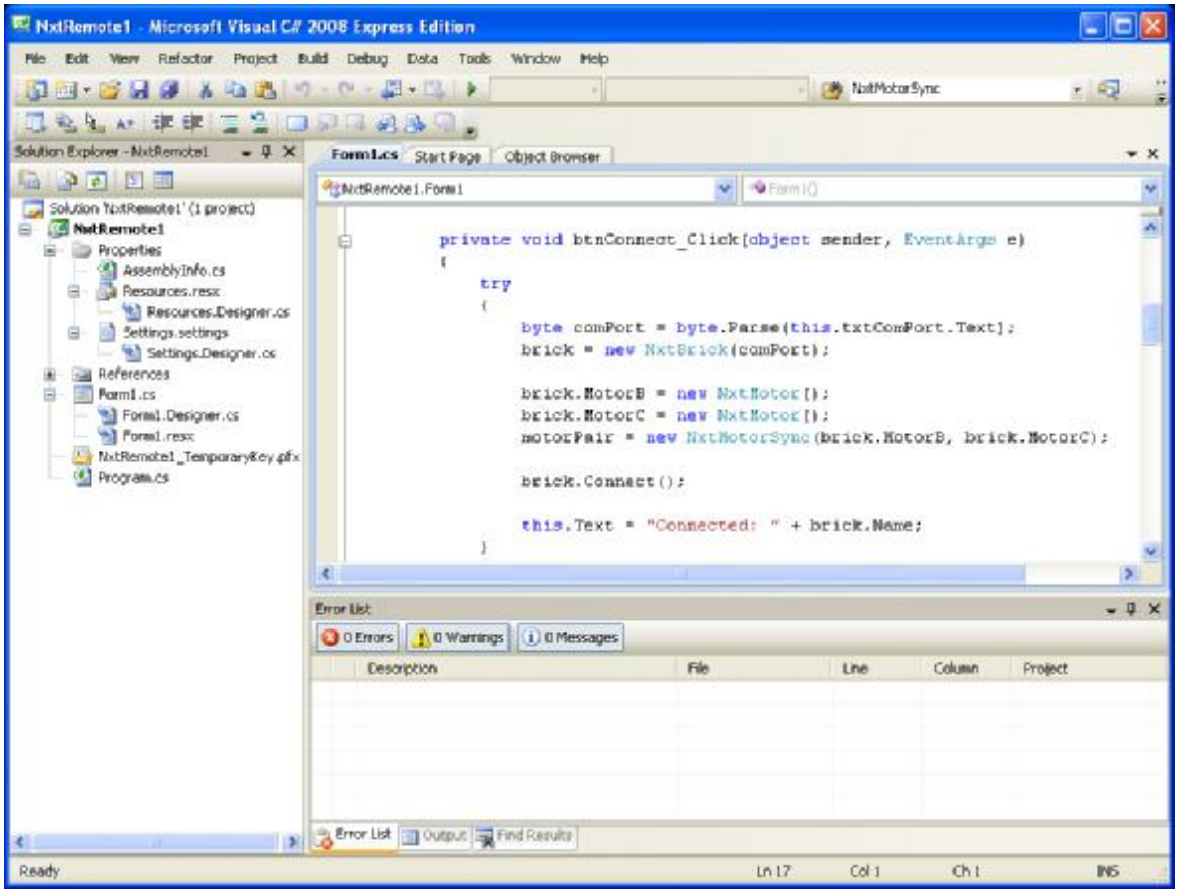


Obr. 6 - Programovací prostředí BricxCC

Toto prostředí je uživatelsky velmi přívětivé. Umožňuje připojení pomocí USB kabelu stejně jako Bluetooth obsahuje mnoho nástrojů jako například přímé ovládání, nástroje na konverzi zvuků, sledování proměnných a jiné. Také obsahuje seznam příkazů a funkcí spolu s funkcí IDE, která nabízí uživateli možnosti při psaní zdrojového kódu.

2.1.3 Jazyk C# v prostředí Microsoft Visual C# 2008 Express Edition

Roboti NXT se dají programovat s použitím některé z dostupných knihoven i v jazyku C# z dvou testovaných knihoven se zdála NKH.MindSqualls.dll kompletnější a oproti Bram.Utilities.dll byla dostupná i samostatná nápověda ke všem funkcím. Programování už není tak jednoduché jako u jazyka NXT, ale uživatel se základy programování a aspoň základní znalostí prostředí určitě ocení ještě lépe fungující IntelliSence a možnosti vytváření především remote aplikací. Na obrázku Obr. 7 je vidět kód, který se vykonává po stisknutí tlačítka „Connect“ vytvoření dvou objektů typu motor jejich spárování a připojení k robotu které je infikováno změnou v titulku aplikace.



Obr. 7 - Programovací prostředí Microsoft Visual C# 2008 Express Edition

Při programování lze samozřejmě využívat veškeré možnosti jazyka C#, které v kombinací s funkcemi knihovny vytvoří výslednou aplikaci pro řízení robota.

2.1.4 Přehled vlastností jednotlivých jazyků

Tabulka 1 Přehled některých prostředí a jejich vlastností

Prostředí		**NXT-G**	**ROBOTC**	**NXC**	**NXJ**
Jazyk		Graphical	C	C-like	JAVA
Platformy		Windows, MAC OSX	Windows	Windows, MAC OSX,LINUX	Windows
Schopnost uživatele		Nováček, středně pokročilý	Nováček, pokročilý	Středně pokročilý, pokročilý	Pokročilý, zkušený
Použitelnost (1 - 10, 10 nejlepší)		10	8	6	4
Relativní rychlost programu		1X	130X	25X	n/a
IDE		ANO	ANO	ANO	s Eclipse
Vývoj v reálném čase		NE	ANO	částečně	NE
Možnosti přehrávání zvuku		Tones + WAV	Tones + WAV	Tones + WAV	Tones + WAV
MOŽNOSTI JAZYKA	Proměnné	?	ANO	ANO	ANO
	Použití „stringů“	NE	ANO	NE	ANO
	Trigonometrické funkce	NE	ANO	NE	?
	Používání ‘switch’	ANO	ANO	?	ANO
	Použití polí	NE	ANO	ANO	ANO
BLUETOOTH	Podporovaná zařízení	PC, NXT	PC, NXT, GPS, jiné	PC, NXT	NXT, jiné
	Fantom protokol	ANO	ANO	ANO	NE
	Rychlost (duplex)	HALF	FULL	HALF	FULL

II. PRAKTICKÁ ČÁST

3 NÁVRH TVORBY NOVÝCH KNIHOVEN PRO NXT ROBOTA

Vzhledem k podpoře různých programovacích jazyků a existenci několika knihoven (C++, C#) popisujících funkce, které robot umožňuje, je tvorba nových knihoven na místě zejména při vzniku nových rozšiřujících HW součástí. Jako další možnost se nabízí rozšíření stávajících řešení o nové funkce nebo tvorba knihoven pro další prostředí, která zatím nebyla použita pro řízení robota. Oba tyto způsoby budou popsány na následujících stránkách. Knihovny by se také daly použít pro sofistikovaný způsob využívání skupin algoritmů, které by v nich byly připraveny k použití v programech. Robot NXT nemá však výpočetní ani paměťovou kapacitu, která by byla využitelná ke složitějším algoritmům, proto jeho úloha při řešení problémů spočívá především ve sběru dat a odesílání hodnot na řídící PC.

3.1 Rozšíření stávajících dostupných řešení

Tento přístup přináší velkou časovou úsporu, která vychází z rozdílné časové náročnosti na vytvoření kompletního řešení obsahujícího veškerou dostupnou funkčnost zařízení v tomto případě robota, a nalezení dostupného řešení, které již existuje. Při hledání existujících řešení je třeba nejprve vytvořit hodnotící kritéria, podle kterých se bude samotný výběr řídit. Kritéria by samozřejmě měla korespondovat se zaměřením vlastního úkolu, který se má v daném případě realizovat.

V tomto případě byla zvolena kritéria výběru:

- základní znalost programovacího jazyka
- dostupnost programovacího prostředí
- dostupnost dokumentace ke stávajícímu řešení
- možnost komunikace a řízení pomocí Bluetooth
- možnost propojení s prostředím Mathematica

Podle těchto kritérií byla zvolena knihovna NKH.MindSqualls.dll, která je vytvořena v jazyku C# a splňuje všechny výše uvedené podmínky.

3.2 Řízení robota NXT v prostředí Mathematica s DLL knihovnou

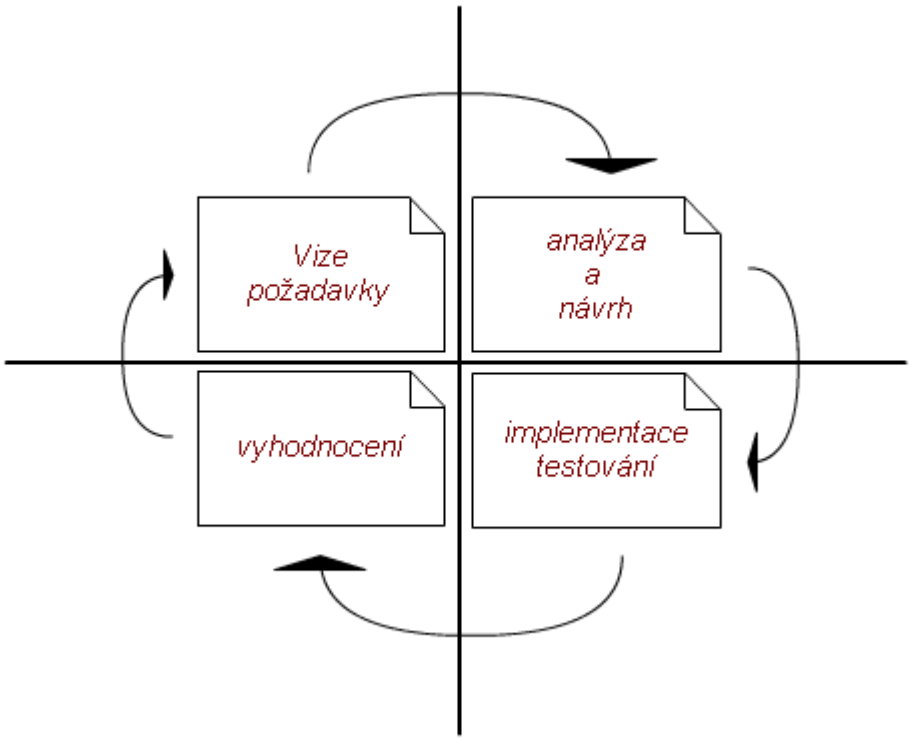
Nejprve byla brána v úvahu varianta vytvoření vlastní knihovny, která by byla vytvořena v prostředí Mathematica za účelem řízení a komunikace s robotem NXT. Tento přístup byl po zvážení všech možných aspektů, jako jsou například časová náročnost a kvalita výsledku, přehodnocen. Většina moderních programovacích prostředí a jazyků podporuje volání knihoven z jiných prostředí nebo volání celých programů. Prostředí Mathematica umožňuje komunikaci s knihovnami jiných programovacích jazyků způsobem přilinkování knihoven a voláním jejich funkcí. K tomuto řešení byla stejně jako v předcházejícím bodě použita knihovna NKH.MindSqualls.dll, která byla rozšířena o další část jak je popsáno v části 4.1 z důvodů, které byly objeveny při testování řízení a popsány v bodě 4.2.5.

3.3 Tvorba knihoven obsahující skupiny algoritmů

Původně měla při této práci vzniknout i knihovna algoritmů, které by bylo možno použít k řízení robota, ale po bližším seznámení s možnostmi robota byla práce na této části zastavena. Původní myšlenka byla využití známých a používaných skupin algoritmů při řízení robota jako jsou například evoluční a genetické algoritmy. Tvorba knihovny byla zastavena z důvodu technického vybavení robota viz. 1.2.1 a také zejména z důvodu, že u remote aplikací je k dispozici výpočetní kapacita PC, která je samozřejmě výkonově nesrovnatelná s výpočetním výkonem robota. Další myšlenkou, kterou přispěl vedoucí práce bylo testování kooperativních algoritmů například v „robotickém“ fotbale. V tomto případě by šlo o vytvoření kooperativních algoritmů řídících celý tým, jehož společným cílem by bylo samozřejmě vstřelení gólu do protivníkovy branky. V tomto případě by bylo třeba sledovat pozici míče, spoluhráčů a soupeřovy branky, což by s dostupným vybavením bylo dosti problematické. Týmy by musely být složeny z maximálně čtyř robotů, zejména kvůli komunikačnímu omezení, které je u NXT dáno 3 zařízeními. Na každém robotu by nepřetržitě běžel program, který by vyhodnocoval stav senzorů a posílal je ostatním spoluhráčům, kteří by dělali to samé. Každý z robotů by musel mít vlastní program upravený podle postu, jenž by na hřišti zastával u všech členů týmu by musela být implementována komunikační část s velmi kvalitní synchronizační smyčkou, která by se starala o předávání informací a vyhodnocování stavu na hřišti a další strategie.

4 REALIZACE NAVRŽENÉHO ZPŮSOBU

K realizaci navržených řešení uvedených v části 3.1 a 3.2 bylo použito dvou prostředí Microsoft Visual C# 2008 Express Edition a Mathematica 5.2. V obou následně uvedených řešeních byla použita knihovna NKH.MindSqualls.dll, která byla objevena při literární rešerši a později při testování vybrána jako vhodné řešení, na které v této práci bylo navázáno. Řešení jsou uvedena v logické posloupnosti, ve skutečnosti však vývoj probíhal souběžně, dal by se zakreslit do obecně známého spirálového modelu viz Obr. 8.



Obr. 8 - Obecný nákres jedné iterace spirálového modelu

Práce také probíhaly ve více prostředích a to nejen v době testování nalezených řešení, ale také při vlastním vývoji. Pokud se při testování objevil problém nejprve byla snaha vyřešit problém podrobnějším nastudováním možných příčin a poté byla hledána alternativa v podobě například jiného příkazu. Pokud se problém stále nepodařilo vyřešit znovu se hledaly problémy týkající se podobné problematiky, ze kterých by mohlo vzejít řešení i pro náš problém. Mezitím mnohdy také došlo ke změně prostředí. Proto třeba problém 4.3.1, který se projevil při řízení z prostředí Mathematica byl vyřešen pomocí jazyka C#.

4.1 Rozšíření stávajících dostupných řešení

Jak již bylo uvedeni výše tento přístup přináší velkou časovou úsporu po nalezení vhodného řešení v tomto případě NKH.MindSqualls.dll se můžete zaměřit na řešení vlastního úkolu.

K rozšíření již kompletního řešení bylo přistoupeno zejména z důvodu existence problémů, které byly objeveny při testování řízení z prostředí Mathematica, které jsou podrobněji popsány v části 4.2.5. Nejprve byla snaha řešit problémy v prostředí Mathematica. Avšak řešení se tímto způsobem nepodařilo nalézt, a proto musely být problémy řešeny v jazyku C#. Ze dvou možných řešení bylo vybráno rozšíření knihovny respektive vytvoření nové knihovny využívající funkce původního řešení. Řešení, jehož obsahem bylo vytvoření programu který by příkazy z prostředí Mathematica převáděl a korektně pak řídil NXT bylo zavrženo z důvodu jeho nízké univerzálnosti a efektivnosti. Knihovnu vlastních funkcí lze jednoduše upravovat a rozšiřovat. 100% univerzálnosti nejde však dosáhnout ani zde a to zejména z toho důvodu, že uživatel může změnit konstrukci robota. Jednotlivé funkce byly převzaty z původního řešení, ke kterému nebyly dostupné zdrojové soubory pouze zkompilované řešení, které bylo v této práci rozšířeno například jako zde v ukázce.

```
bool run = true;
while (run)
{
      run = false;
      NxtGetOutputStateReply? reply =
      _brick.CommLink.GetOutputState(NxtMotorPort.PortB);
      if (reply.HasValue)
      run = (reply.Value.runState ==
      NxtMotorRunState.MOTOR_RUN_STATE_RUNNING);
      System.Threading.Thread.Sleep(100);
}
```

Jedná se o smyčku, která se dotazuje na stav motoru a pokud není jeho běh ukončen vyvolá uspání vlákna, což zabrání v běhu dalším příkazům. Ukázka finální funkce ošetřující tento problém je uvedena v části 4.3.1 jež se zabývá problémem synchronizovaného řízení.

4.1.1 Použití knihovny NxtRemoreLibrary.dll

Knihovna je vytvořena jako samostatný projekt obsahující několik základních funkcí a metod. Pro řízení robota NXT pro použití je třeba dodržet konstrukci robota Tag-bot s konstrukční úpravou popsanou v části 4.3.3. Nebo upravit knihovnu především nastavení portů a senzorů, které jsou na ně umístěny a také funkcí, které byly vytvořeny pro tuto konstrukci.

```
public void Connect(int port)
```

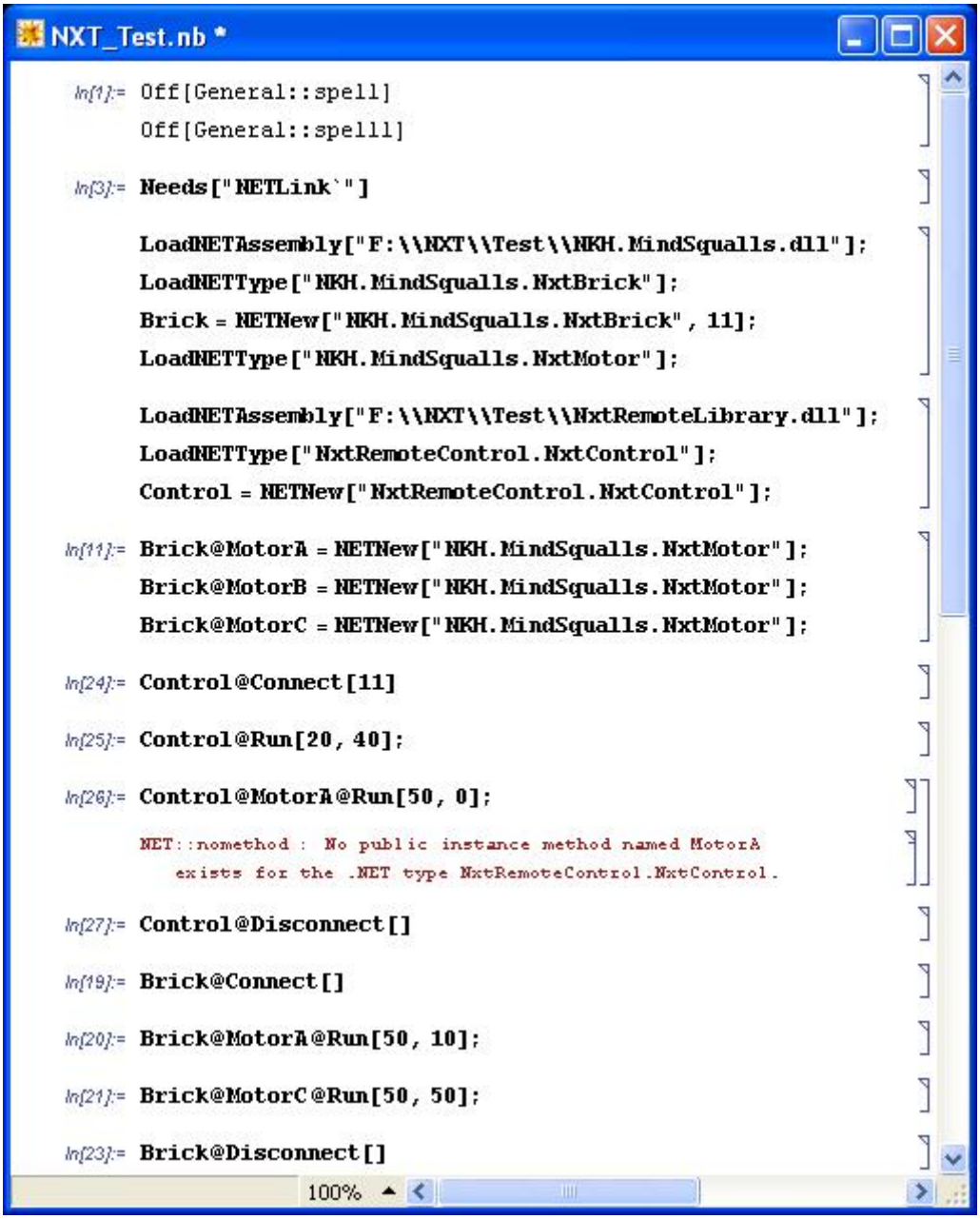
```
        {
            NXT = new NxtBrick((byte)port);
            NXT.MotorA = new NxtMotor();
            NXT.MotorB = new NxtMotor();
            NXT.MotorC = new NxtMotor();
            NXT.Sensor1 = new NxtLightSensor();
            NXT.Sensor2 = new NxtUltrasonicSensor();
            NXT.Sensor3 = new NxtTouchSensor();
            NXT.Sensor4 = new NxtSoundSensor ();
            SyncMotors = new NxtMotorSync(NXT.MotorA, NXT.MotorC);

            NXT.Connect();
        }
```

V metodě ***Connect*** vidíme vytvoření NXT na zadaném portu, motory, typy senzorů přiřazené na jednotlivé porty a synchronizaci motoru A s motorem C. Vytvoření a přiřazení zařízení je vždy třeba provést před samotným připojením k NXT. U senzorů je ještě možnost nastavit periodu, se kterou bude měřící senzor dotazován na měřenou hodnotu.

```
static void Main(string[] args)
        {
            // Vytvoření NXT na portu COM40.
            NxtBrick brick = new NxtBrick(40);

            // Vytvoření dotykového senzoru.
            NxtTouchSensor touchSensor = new NxtTouchSensor();

            // Přiřazení senzoru na port 1.
            brick.Sensor1 = touchSensor;

            // Dotazování nastaveno na 50 milisekund.
            touchSensor.PollInterval = 50;

            // Vytvoření „handleru“ události senzoru.
            touchSensor.OnPolled += new Polled(touchSensor_OnPolled);

            // Připojení k NXT.
            brick.Connect();

            // Vypsání řádky na display a čekání na stisk
            Console.WriteLine("Press any key to stop.");
            Console.ReadKey();

            // Odpojení od NXT.
            brick.Disconnect();
        }

            // Metoda obsluhující událost senzoru touchSensor.OnPolled.
static void touchSensor_OnPolled(NxtPollable polledItem)
        {
            NxtTouchSensor touchSensor = (NxtTouchSensor)polledItem;
            if (touchSensor.IsPressed)
                Console.WriteLine("Pressed");
            else
                Console.WriteLine("Please press the sensor!");
        }
```

Ukázka použití metody dotazování senzorů s přeloženými komentáři převzatá z [5].

Pro použití knihovny NxtRemoreLibrary.dll je třeba mít v notebooku vytvořeno spojení i s původní knihovnou NKH.MindSqualls.dll jak je vidět na Obr. 9.



Obr. 9 - Ukázka chybného použití funkce

Na obrázku je také vidět použití funkce ***Run***, která náleží původní knihovně. Při pokusu o její zavolání za použití chybné knihovny vrátí prostředí Mathematica chybu, že daná metoda není v knihovně. Pokud by knihovna tuto metodu obsahovala a obsahovala také funkci, příkaz by se korektně provedl. V opačném případě by chybové hlášení obsahovalo informaci o neexistenci funkce.

4.2 Propojení prostředí Mathematica s DLL knihovnou

Většina moderních programovacích prostředí a jazyků podporuje volání knihoven z jiných prostředí nebo volání celých programů. Prostředí Mathematica umožňuje komunikaci s knihovnami jiných programovacích jazyků způsobem přilinkování knihoven a voláním jejich funkcí. V této části bude prezentováno použití knihovny jazyka C# pomocí modulu NETLink nejprve je však třeba provést několik kroků pro jeho správné fungování.

4.2.1 Přilinkování knihovny jazyka C#

Pro úspěšné propojení je nejprve nutné zjistit verze Framework-u, které máte nainstalovány na vašem počítači informaci o instalovaných verzích naleznete standardně v adresáři ***C:\WINDOWS\Microsoft.NET\Framework***. Poté je třeba upravit konfigurační soubor ***InstallableNET.exe.config*** prostředí Mathematica, který se při standardně nalézá ve složce

C:\Program Files\Mathematica5.2\AddOns\NETLink a změnit hodnotu podporované verze Framework-u `<supportedRuntime version="v2.0.50727" />`. Samozřejmě na verzi, kterou máte nainstalovanou, předchozí nebo následující řádky můžete smazat a zanechat v souboru pouze Vámi editovaný řádek při editaci je také třeba zachovat formátování souboru.

4.2.2 Obsah samotné knihovny

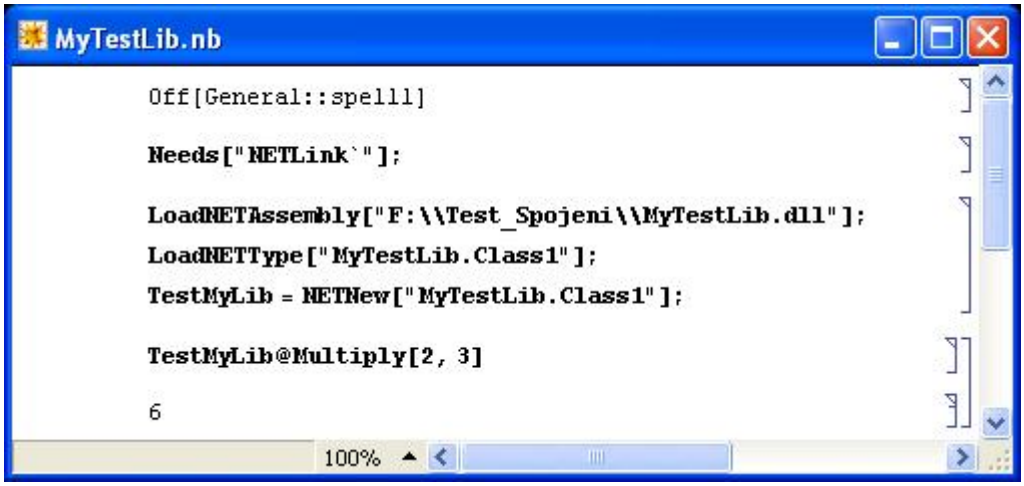
Pro otestování funkce přilinkování knihovny stačí vytvořit nový projekt typu knihovna a v něm jednu třídu obsahující pro názornost elementární funkci realizující násobení dvou vstupních prvků vracející výslednou hodnotu.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace MyTestLib
{
    public class Class1
    {
        public int Multiply(int x, int y)
        {
            return (x * y);
        }
    }
}
```

Tento příklad je pouze demonstrativní a byl zvolen pro jeho jednoduchost a názornost.

4.2.3 Obsah notebooku Mathematica

První řádek je pouze vypnutím upozornění prostředí Mathematica na podobné názvy proměnných tento příkaz je samozřejmě nepovinný. Druhý příkaz je volání standardního modulu Mathematica který slouží pro volání a spolupráci s prostředím Microsoft .NET Framework.

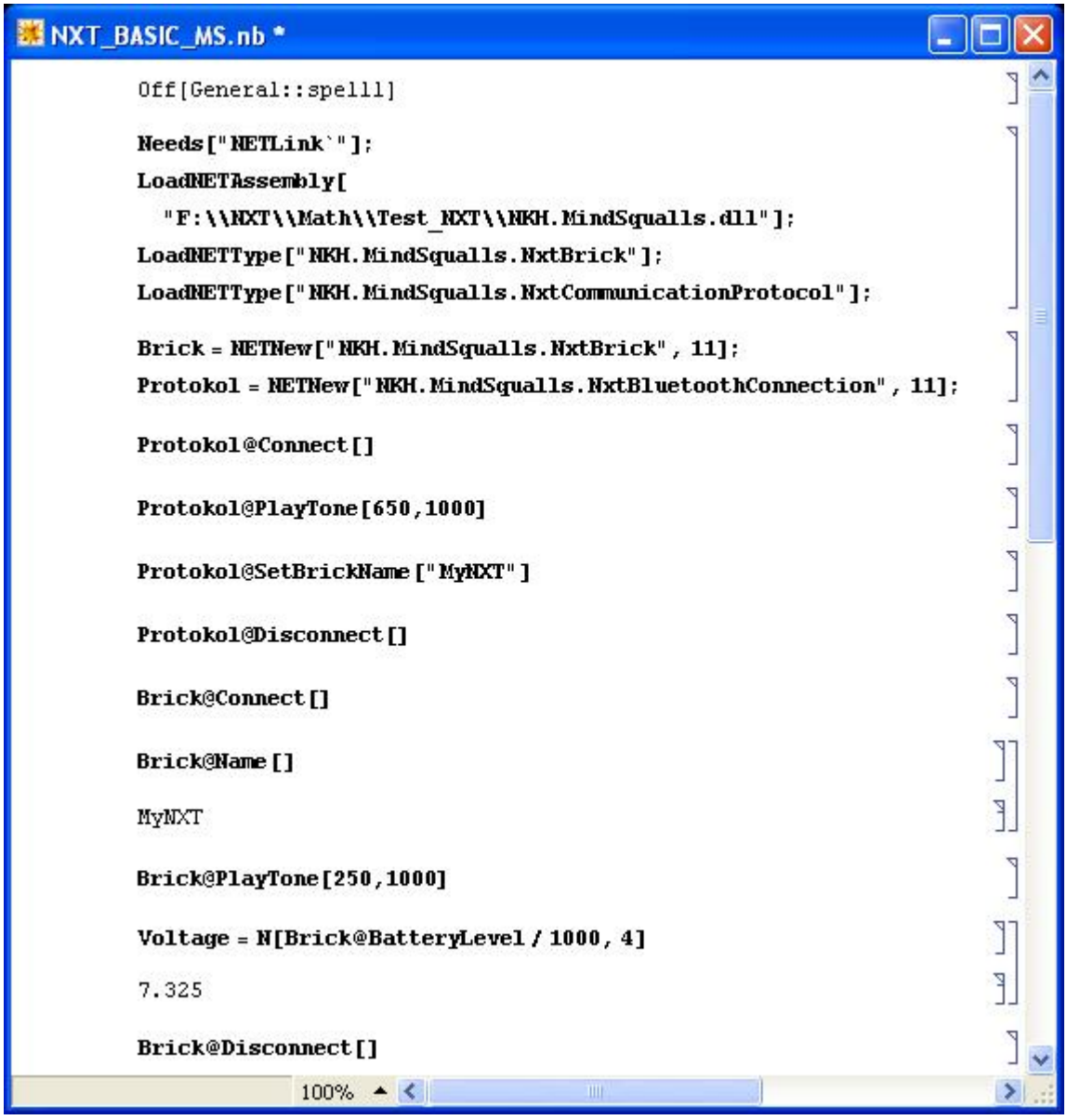


Obr. 10 - Příklad testovacího notebooku

Třetí příkaz slouží k načtení knihovny z uvedeného umístnění zápis cesty je klasický pouze s dvojitými lomítky. Příkaz ***LoadNETType*** zajišťuje načtení třídy z příslušné knihovny poslední příkaz třetího bloku slouží k vytvoření objektu třídy pro použití v prostředí Mathematica. Nyní je vše připraveno pro použití Vámi vytvořených funkcí syntaxe odpovídá poslednímu vstupu ukázkového příkladu nejprve zadáte název objektu jehož funkce chcete volat a pak již zadáte originální název funkce s příslušným počtem vstupních parametrů. Spojovacím operátorem je v tomto případě zavináč. Je samozřejmé, že nejde používat funkce objektu které nejsou v prostředí Mathematica vytvořeny nebo je obsahují jiné objekty. V další části 4.2.4 je ukázka bezparametrových funkcí a také používání více tříd stejné knihovny. Obě tyto třídy komunikují s NXT, proto již při vytváření mají jako vstupní parametr 11, což odpovídá číslu portu pro bluetooth spojení. Funkce ***Connect*** slouží pro navázání spojení se zařízení, které se snaží realizovat přes port který byl zadán při vytvoření objektu čili COM 11. Funkce nevrací žádnou hodnotu navázání spojení je indikováno zelenou ikonou Bluetooth v try oblasti Windows pokud nedojde ke spojení dojde k zahlášení chyby. ***Disconnect*** slouží k ukončení spojení. Opět se jedná o funkci nevracející žádné informace.

4.2.4 Ukázka testovacího notebooku

Základní příkazy již byly popsány v předcházející části 4.2.3. Tato jednoduchá ukázka slouží k ukázce možností používání stejných funkcí jiných tříd. V ukázce jsou také použity funkce ***SetBrickName*** a ***Name***, které slouží k nastavení a zjištění jména NXT tyto funkce jsou však obsaženy v různých třídách.



Obr. 11 - Testování základních příkladů v prostředí Mathematica

4.2.5 Seznam chyb

Při použití prostředí Mathematica k řízení NXT se mohou vyskytnout chyby, které vznikly bud chybou v komunikaci nebo chybným zadáním parametrů. Nejčastějšími chybami jsou chyby vyskytující se po použití příkazu ***Connect***. Pokud byly již přístroje jednou spárovány prostředí Mathematica je schopno navázat spojení i bez použití aplikace pro Bluetooth připojení. Někdy se při ladění programu nezbývá než vypnout NXT nebo ukončit kernel prostředí Mathematica po novém spuštění metody ***Connect*** se mohou vyskytnout tyto chyby:

NET::netexcptn: A .NET exception occurred: System.UnauthorizedAccessException: Access to the port 'COM11' is denied. …

NET::netexcptn: A .NET exception occurred: NxtCommunicationProtocolException - Command: LsGetStatus; Error: SpecifiedChannelOrConnectionNotConfiguredOrBusy; …

Řešení je několik: ukončení prostředí Mathematica, ukončení spojení s NXT. Při ztrátě spojení PC a NXT během vykonávání programu se objeví zde uvedená chyba nebo hlášení „`Failed`“.

LinkObject::linkd: LinkObject[C:\Program Files\Mathematica5.2\ AddOns\NETLink\ InstallableNET.exe,2,2] is closed; the connection is dead. …

V případě předání špatných parametrů, jejich počtu, či tvaru volané funkci nebo metodě se můžete setkat s těmito dvěma chybami.

NET::methodargs: Improper arguments supplied for method named Stop.

NET::methargc: Wrong number of arguments supplied for method Rotate.

Další přiklad chyby je zachycen na Obr. 9 kde za zavolána metoda knihovny, která v ní není obsažena.

4.3 Problémy řešené při realizaci

Po zprovoznění Bluetooth komunikace a úspěšném propojení prostředí Mathematica s vytvořenou knihovnou jazyka C# bylo testování přesunuto na zvolenou knihovnu MindSqualls.

4.3.1 Synchronizované řízení

Při testování řízení v prostředí Mathematica se ukázalo, že déle trvající příkazy nebo spíše činnost která je jimi vykonávána a trvá delší časovou dobu není dokončena. Tento problém se nejvíce projevil při otáčení řídícího systému, což vedlo k chování robota, které neodpovídalo naprogramovaným činnostem. Zde uvedený kód ukazuje finální řešení tohoto problému funkcí které jsou jako parametry zadány port motoru, který vykonává činnost. Druhý parametr je řídící hodnota pro ošetření uváznutí motoru, na který se v případě vypršení časového limitu ***hodnota*** x 100 milisekund již dále nečeká.

```
private bool WaitForCommand(NxtMotorPort motorPort, int retryLimit)
{
      bool running = true;
      int retryCount = 0;

      while (running)
            {
                  running = false;
                  NxtGetOutputStateReply? reply =
                  _brick.CommLink.GetOutputState(motorPort);
                  if (reply.HasValue)
                   running = (reply.Value.runState ==
                   NxtMotorRunState.MOTOR_RUN_STATE_RUNNING);
                  System.Threading.Thread.Sleep(100);

                  if (retryLimit != -1)
                   if (++retryCount >= retryLimit)
                   return false;
            }
            return true;
}
```

4.3.2 Krokové řízení

Stejně tak jako při testování jednotlivých příkazů tak i při tvoření smyčky realizující automatický běh robota se objevily problémy. V tomto případě u krokového řízení motoru ovládajícího řídící systém. Při řešení tohoto problému byly testovány různé metody řízení, které vycházely z možností implementovaných v knihovně MindSqualls. Nejprve byl motor stejně jako hnací motory řízen pomocí funkce ***Run***, která po zadání dvou parametrů

realizuje otočení motoru silou 0 – 100% směr pohybu se řídí znaménkem před prvním parametrem. Druhý parametr udává otočení ve stupních. Nula je brána jako nekonečno a motor pokračuje v běhu i po skončení příkazu.

```
public void Run
      (
            sbyte power,
            uint tachoLimit
      )
```

Pro ukončení pohybu lze použít několik příkazů: **Brake** zastaví motor, **Coast** nechá motor a **Idle** uvede motor do stavu nečinnosti.

```
public void Brake ()
public void Coast ()
public void Idle ()
```

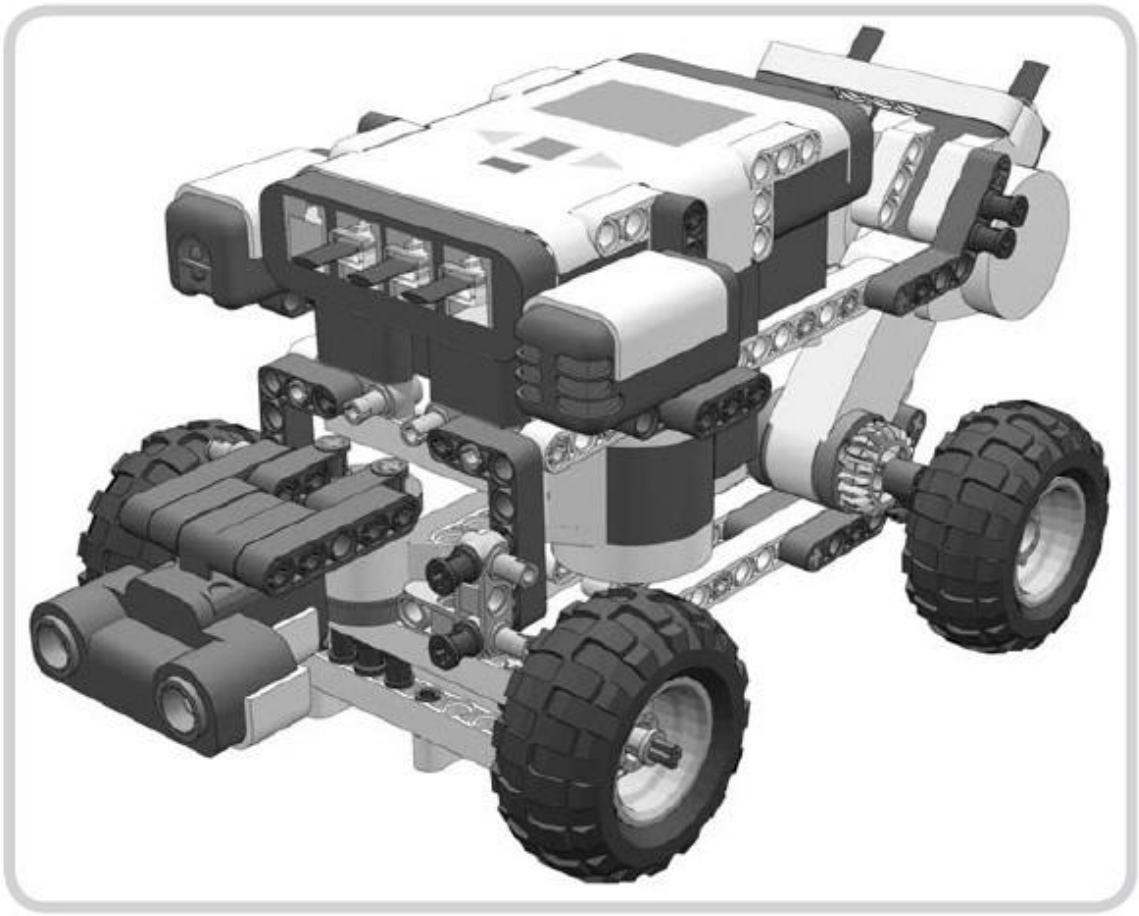
Při použití příkazů řízení fungovalo, ale někdy motor při použití malé síly a při velkém odporu povrchu nemohl dokončit pohyb a vydával akustický signál. Naopak při použití příliš velké síly motor požadovaný úhel překročil a pak se do žádané hodnoty vracel již menší silou. Zde se opět objevil problém s dosáhnutím přesné polohy. Tento způsob řízení je použit i v původním SW. Někdy došlo také k tomu, že motor stále běžel a kmital kolem požadované hodnoty, které se snažil neustálými úpravami polohy dosáhnout.

```
public void SetOutputState
      (
            NxtMotorPort motorPort,
            sbyte power,
            NxtMotorMode mode,
            NxtMotorRegulationMode regulationMode,
            sbyte turnRatio,
            NxtMotorRunState runState,
            uint tachoLimit
      )
```

Druhým testovaným příkazem byl ***SetOutputState*** řídících parametrů je zde sedm. Nejpodstatnější je opět síla a otočení motoru, ke kterým je přidána volba portu, mód motoru. Typ regulace a stav motoru při běhu všechny funkce a možnosti jejich nastavení jsou popsány v originální dokumentaci MindSqualls_v1_2_1.chm, která je vzhledem k rozsahu přiložena pouze v elektronické formě.

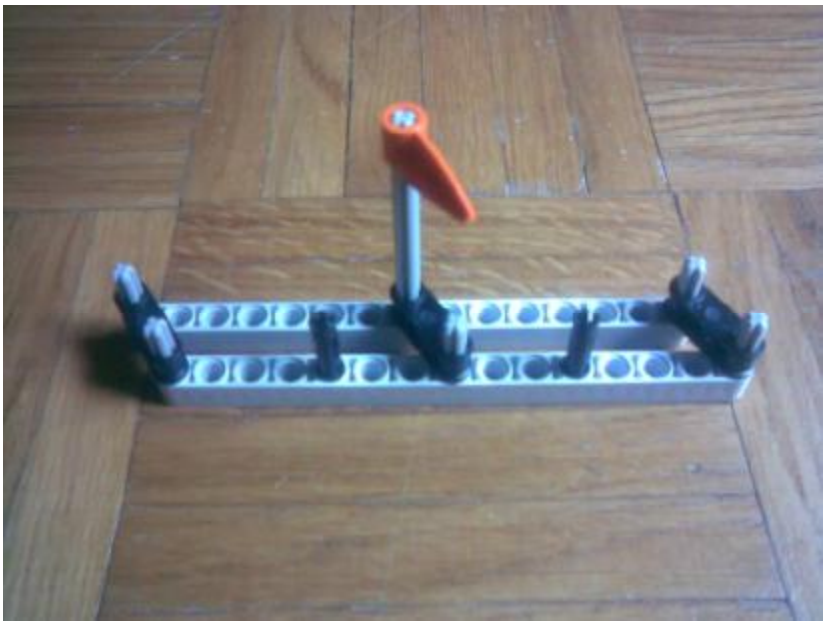
4.3.3 Úpravy konstrukce řídícího systému

Na Obr. 12 můžete vidět původní konstrukci a také jediný materiál, podle kterého byl testovací robot postaven, posléze byla na internetu nalezena část knihy, ve které byly uvedeny potřebné součástky a několik základních kroků. Robot je konstrukčně řešen jako vozidlo s náhonem na zadní dvojici kol, které pohání dva motory, které je možno synchronizovat jak konstrukčně tak softwarově.Třetí motor umístěný v rámu pod řídící jednotkou zajišťuje pohon řídícího systému a senzoru, který měří vzdálenost od překážek.



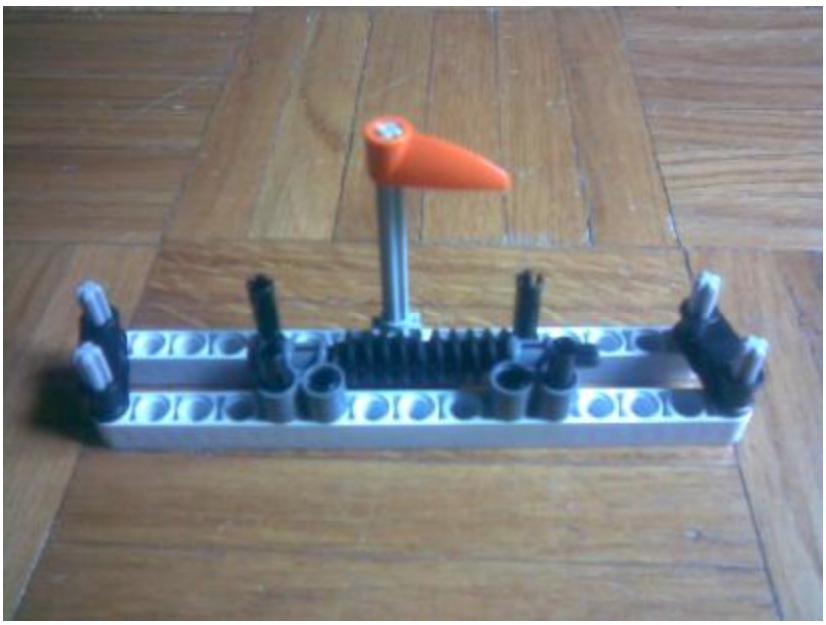
Obr. 12 - Originální konstrukce Tag-bot

Na dalším Obr. 13 je vidět originální systém řízení, delší osa umístěná ve středu přední nápravy znázorňuje umístnění osy, kterou pohybuje motor. Oranžová součástka na ose je pouze pomocná a slouží k odhadnutí úhlu potřebného otočení. Kola v tomto případě byla uložena nezávisle na krátkých osách. Úhel otočení řídícího motoru se pohyboval v rozmezí -60º + 60º, což muselo být z důvodu délky nápravy a narážení kol do konstrukce programově ošetřeno na -40º + 40º.



Obr. 13 - Původní systém řízení

Dalším testovaným řešením vlastní konstrukce byl přenos síly pomocí šroubového systému. Osa motoru byla opět umístěna ve středu prvního nosníku přední nápravy na druhém nosníku byla umístěna šroubovice, která převáděla pohyb motoru na pohyb kol. Tento systém byl také funkční, ale v krajních polohách docházelo k nadměrnému namáhání soustavy realizující přenos sil, úhel otočení motoru byl téměř -90º + 90º.

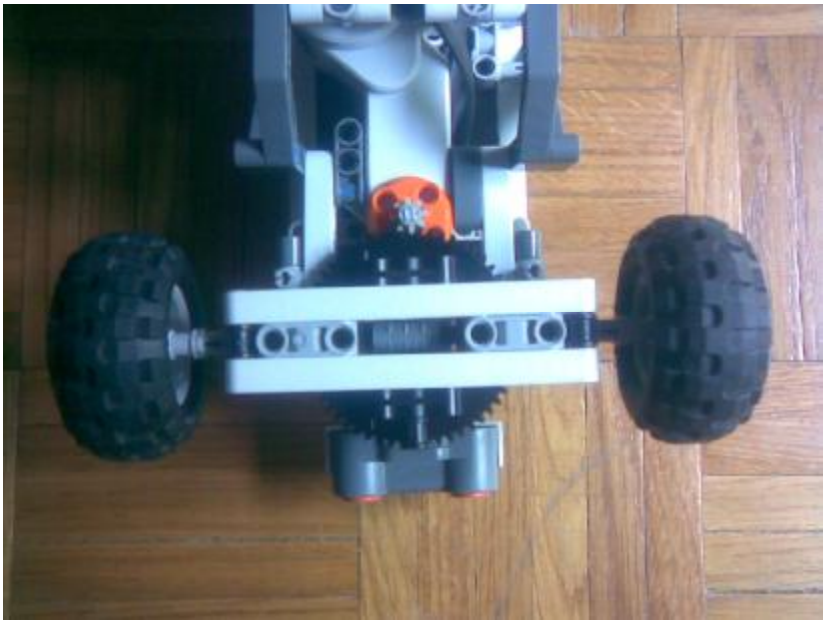


Obr. 14 - Řešení se šroubovým převodem

Při konstrukci třetího řešení v řadě Obr. 15 bylo zasaženo do originální konstrukce vysunutím motoru více dopředu zejména z důvodu potřeby vytvoření místa pro řídící systém. Také byla zkrácena a otočena přední náprava čímž se zmenšil rozchod kol, které byly také umístněny na společnou osu hybná osa motoru byla tentokrát umístněna přímo do středu nápravy takže je na Obr. 15 krytá spojovací součástí osy kol. Síla motoru byla v tomto případě přenášena pomocí ozubených kol s převodovým poměrem 1:1.

Obr. 15 - Systém řízení po konstrukčních úpravách

Pro konečnou konstrukci, jejímž základem byla konstrukce předchozí, byla upravena především pozice motoru, který byl posunut dozadu jak jen to bylo možné. Nejpodstatnější změnou však byla úprava řídícího převodu, která konečně vyřešila problém s krokovým řízením.



Obr. 16 - Řídící systém finální řešení převod 8 : 54

Jak je vidět na Obr. 17 - Finální konstrukceObr. 17 původní konstrukce byla upravena. Přídavné konstrukční prvky po stranách jsou umístněny tak, aby vytvářely přítlak ozubeného kola řízení na ozubené kola na hřídeli motoru. Obě součásti na sebe doléhají i za normálních okolností, ale z důvodu že pohyb motorů není měřen a spoléháme se pouze na přesnost řízení motoru je třeba omezit možnosti vzniku chyb. Pod zadní dva motory bylo ještě umístěno dotykové čidlo které při couvání kontroluje, zda nedošlo k nárazu.

Obr. 17 - Finální konstrukce

5 DEMONSTRACE ŘEŠENÍ

V této části se nalézají finální řešení vzniklé při této práci.

5.1 Knihovna NxtRemoteLibrary.dll

Knihovna obsahuje metody a funkce pro připojení k NXT, řízení robota a snímaní hodnot čidel.

5.1.1 Metody pro komunikaci

Funkce komunikaci jsou pouze dvě. ***Connect*** sloužící pro připojení k NXT, která obsahuje vytvoření všech použitelných periférií a jejich přiřazení k jednotlivým portům spolu se synchronizací dvou motorů poháněcích robota. Každému senzoru je také nastavena perioda, se kterou se ho řídící jednotka dotazuje na měřenou veličinu.

```
public void Connect(int port) //Metoda pro vytvoření spojení.
      {
            NXT = new NxtBrick((byte)port);
            NXT.MotorA = new NxtMotor();
            NXT.MotorB = new NxtMotor();
            NXT.MotorC = new NxtMotor();
            NXT.Sensor1 = new NxtLightSensor();
            NXT.Sensor1.PollInterval = 50;
            NXT.Sensor2 = new NxtUltrasonicSensor();
            NXT.Sensor2.PollInterval = 50;
            NXT.Sensor3 = new NxtTouchSensor();
            NXT.Sensor3.PollInterval = 50;
            NXT.Sensor4 = new NxtSoundSensor ();
            NXT.Sensor4.PollInterval = 50;
            SyncMotors = new NxtMotorSync(NXT.MotorA, NXT.MotorC);

            NXT.Connect();
      }
```

Funkce ***Disconnect*** obsahuje pouze kontrolu existence spojení s NXT a samotné

```
public void Disconnect()      //Metoda pro ukončení spojení.
      {
            if (NXT != null && NXT.IsConnected)
                NXT.Disconnect();
      }
```

5.1.2 Metody pro motory

Metoda ***Rotate*** umožňuje řídit motor na vybraném portu, požadovanou rychlostí odpovídající procentuálnímu vyjádření výkonu a otáčkách, nebo úhlu otočení. Obsahem samotné metody je triviální switch konstrukce se třemi možnými variantami portů doplněná o příkaz ***Run*** z původní knihovny a metodu zajišťující čekání na vykonání pohybu.

```
//Metoda pro řízení motoru.
public void Rotate(string Mport, int power, int angle)
      {
            NxtMotor nxtMotor = null;
            NxtMotorPort nxtMotorPort = NxtMotorPort.All;

            switch (Mport)
            {
                case "A":
                    nxtMotor = NXT.MotorA;
                    nxtMotorPort = NxtMotorPort.PortA;
                    break;
                case "B":
                    nxtMotor = NXT.MotorB;
                    nxtMotorPort = NxtMotorPort.PortB;
                    break;
                case "C":
                    nxtMotor = NXT.MotorC;
                    nxtMotorPort = NxtMotorPort.PortC;
                    break;
            }
            nxtMotor.Run((sbyte)power, (uint)angle);
            WaitForCommand(nxtMotorPort, 10);
      }
```

Metoda ***Run*** je opet použití puvodní funkce k pohonu synchronizovaných motorů A a C k pohybu robota vpřed i vzad.

```
//Metoda pro řízení synchronizovaných motorů.
public void Run(int power, int angle)
      {
            SyncMotors.Run((sbyte)power, (char)angle, 0);
            WaitForCommand(NxtMotorPort.PortC, 10);
      }

//Metoda ukončující běh motoru.
public void Stop(string device)
      {
            switch (device)
                  {
                        case "PortA":
                              NXT.MotorA.Brake();
                              break;
                        case "PortB":
                              NXT.MotorB.Brake();
                              break;
                        case "PortC":
                              NXT.MotorC.Brake();
                              break;
                        case "All":
                              NXT.MotorA.Brake();
                              NXT.MotorB.Brake();
                              NXT.MotorC.Brake();
                              break;
                        default:
                       throw new NotImplementedException("Unknown device.");
                  }
      }
```

Metoda ***Stop*** slouží k zastavení některého nebo všech motorů použita je opět konstrukce pomocí switch a původní metoda ***Brake***. Posledními dvěma metodami jsou ***TurnR*** a ***TurnL*** slouží k odbočení robota o 90º v pravo a vlevo. Po spuštění nelze tuty metody zastavit, metody jsou identické pouze se záměnou proměných ***rPower*** a ***lPower***, které jsou nastaveny na 40 a -40 u těchto metod se zadává pouze rychlost pro obdbočení.

```
public void TurnR(int power)     //Metoda pro odbočení vlevo.
        {
            NXT.MotorB.Run((sbyte)rPower, 120);      //Otočení řízení.
            WaitForCommand(NxtMotorPort.PortB, 10);
            SyncMotors.Run((sbyte)power, 1800, 0);
            WaitForCommand(NxtMotorPort.PortA, 10);
            SyncMotors.Brake();                  //Zastavení motorů.
            NXT.MotorB.Run((sbyte)lPower, 120); //Srovní kol po odbočení.
            WaitForCommand(NxtMotorPort.PortB, 10);
        }
```

Nejdůležitější funkcí je ***WaitForCommand*** tato funkce byla vytvořena pro zajištění synchronního řízení a je použita téměř ve všech metodách i funkcích jak v knihovně pro řízení z prostředí Mathematica tak i v aplikaci „NXTRemoteControl“.

```
//Funkce pro kontrolu běhu motoru.
private bool WaitForCommand(NxtMotorPort motorPort, int retryLimit)
//limit X*100ms
      {
            bool running = true;
            int retryCount = 0;

            while (running)
            {
                running = false;
                NxtGetOutputStateReply? reply =
                NXT.CommLink.GetOutputState(motorPort);

                if (reply.HasValue)
                    running = (reply.Value.runState ==
                    NxtMotorRunState.MOTOR_RUN_STATE_RUNNING);
                    System.Threading.Thread.Sleep(100);

                if (retryLimit != -1)
                    if (++retryCount >= retryLimit)
                        return false;
            }

            return true;
      }
```

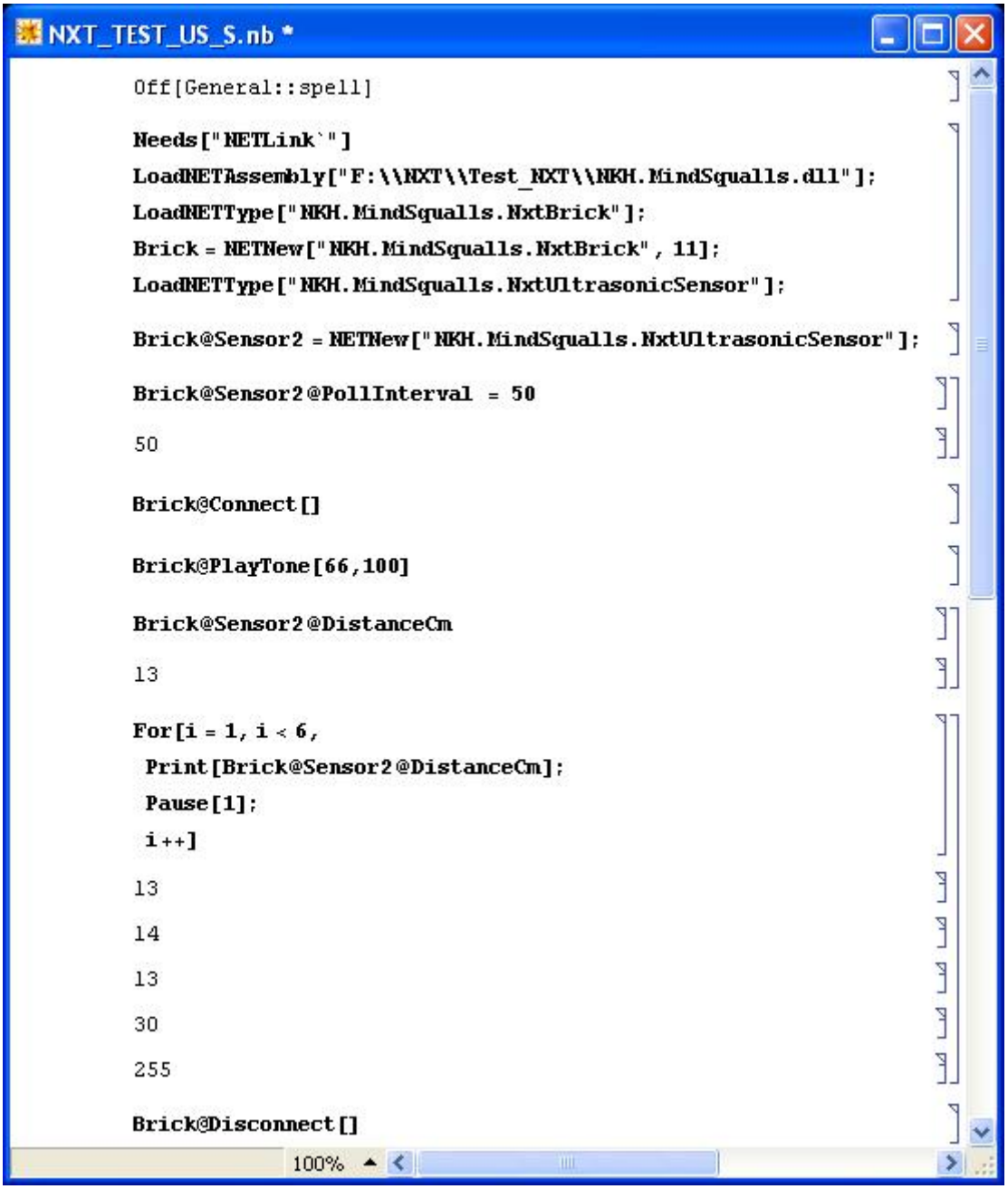
5.1.3 Senzory

Funkce snímání hodnot jednotlivých senzorů jsou velmi jednoduché. Vytvoření senzorů, přiřazení jednotlivých portů a nastavení periody snímání již bylo provedeno v metodě ***Connect***. Samotné funkce obsahují jen příkaz a navrácení změřené hodnoty.

```
public byte LightIntensity()  //Funkce pro zjištění intenzity světla.
      {
            byte? intensity = ((NxtLightSensor)NXT.Sensor1).Intensity;
            return intensity.Value;
      }

public byte GetDistance()     //Funkce pro zjištění vzdálenosti překážky.
      {
            byte? distance =
            ((NxtUltrasonicSensor)NXT.Sensor2).DistanceCm;
            return distance.Value;
      }

public bool IsPressed() //Funkce pro zjištění stavu dotykového senzoru.
      {
            bool? touch = ((NxtTouchSensor)NXT.Sensor3).IsPressed;
            return touch.Value;
      }

public byte SoundLevel()      //Funkce pro zjištění intenzity zvuku.
      {
            byte? loud = ((NxtSoundSensor)NXT.Sensor4).SoundLevel;
            return loud.Value;
      }
```

5.2 Řízení v prostředí Mathematica

Řízení za použití prostředí Mathematica může být realizováno pomocí původní knihovny nebo s použitím obou knihoven záleží na uživateli jaké funkce a metody chce využívat řízení NXT. Při použití původní knihovny Obr. 18 je přiřazení senzorů stejně jako nastavení dotazovacího intervalu provedeno až v prostředí Mathematica. Při používání funkcí NXTRemoteControl.dll je třeba přilinkovat k právě spuštěnému notebooku i původní knihovnu NKH.MindSqualls.dll. Na uvedeném Obr. 9 je vidět že před použitím funkcí jiné knihovny je nejprve třeba ukončit předchozí spojení a navázat nové metodou dané knihovny. K použití jsou všechny metody a funkce původního řešení spolu s nově vytvořenými v knihovně NXTRemoteControl.dll jejichž popis je v předcházející podkapitole 5.1. Funkce a metody lze samozřejmě použít spolu s příkazy prostředí Mathematica jak je vidět na Obr. 18 kde je funkce pro zjištění vzdálenosti vložena do jednoduchého ***For*** cyklu. Této kombinace bylo také využito při tvoření automatického módu robota popsaného v následující kapitole 5.3.

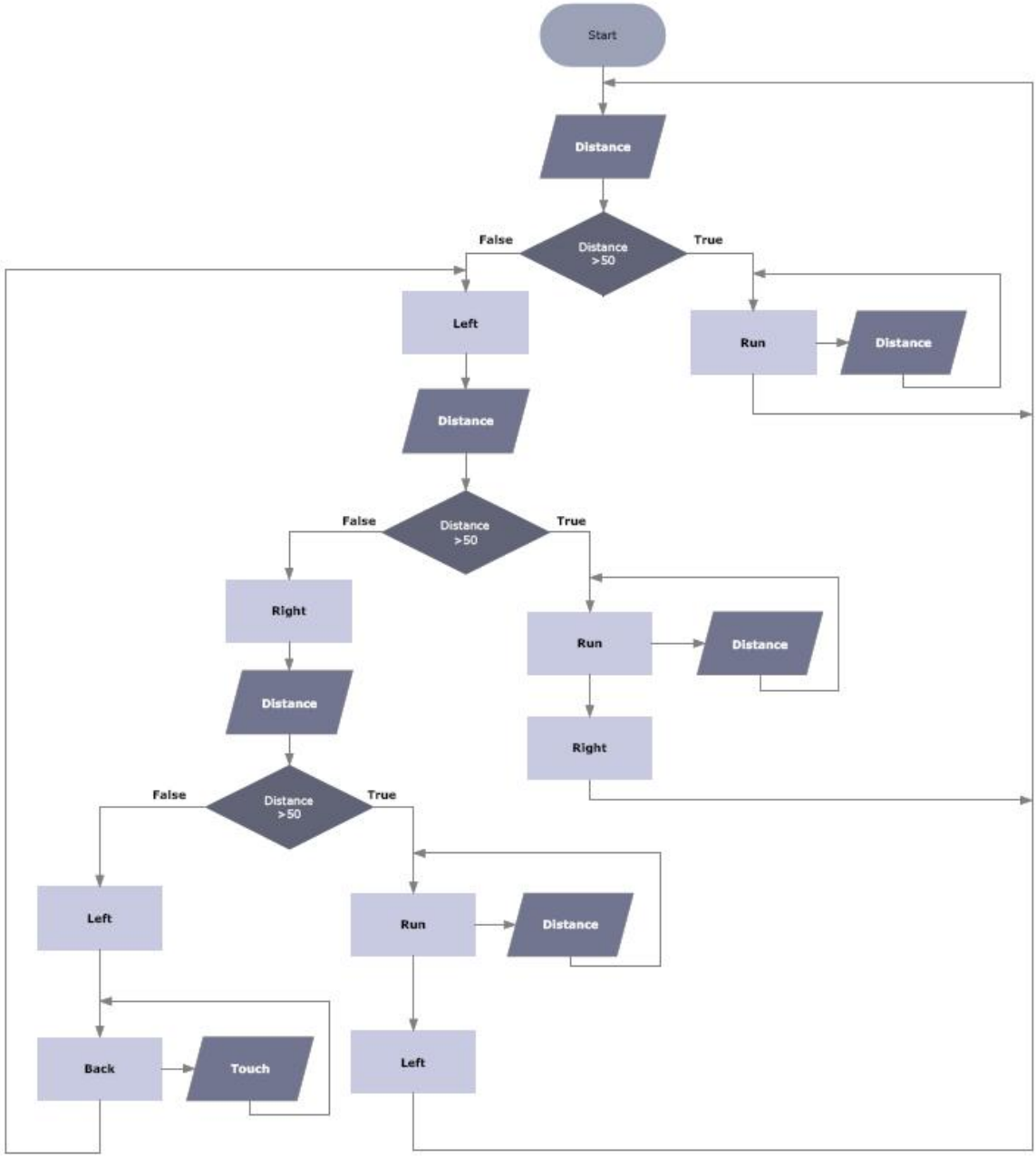


Obr. 18 - Ukázka použití původní knihovny

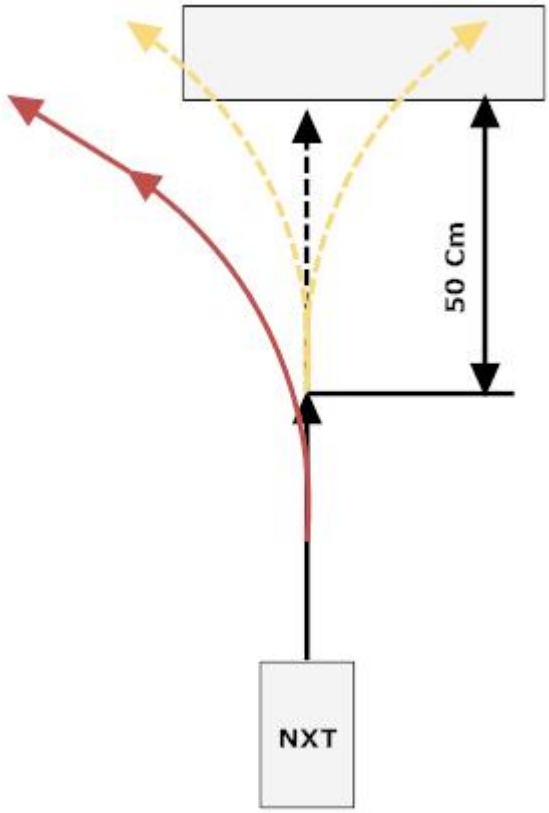
5.3 Program v prostředí Mathematica pro automatický chod robota

Robota lze řídit jednotlivými příkazy a nebo také vytvořit program, který bude řídit robota sám bez dalších zásahů. Jedná se o nekonečnou smyčku, která se neustále opakuje Obr. 19. Logická náročnost smyčky není nijak složitá a samotné její napsání zvládne každý se základní znalostí programování a algoritmizace. Přesto právě při tvoření tohoto programu zajišťujícího automatický mód robota byl objeven problém se synchronním řízením popsaný v bodě 4.3.1. Po spuštění dojde ke změření vzdálenosti k překážce po vyhodnocení robot

bud pokračuje v jízdě a po ujetí dané vzdálenosti pokračuje znovu od začátku nebo se snaží zjistit zda vlevo není více místa. Pokud je vlevo více místa pokračuje v pohybu tímto směrem po ujetí dané vzdálenosti srovná kola a vrátí se na začátek. V případě, že ani vlevo není více místa změří vzdálenost ještě napravo v případě pozitivního výsledku pokračuje tímto směrem. Pokud ani vpravo není dostatek místa srovná se robot do původní polohy a začne couvat dokud nenarazí do překážky nebo neujede požadovanou vzdálenost. V obou případech se pak vrací na kontrolu vzdálenosti vlevo.



Obr. 19 - Schéma smyčky automatického řízení



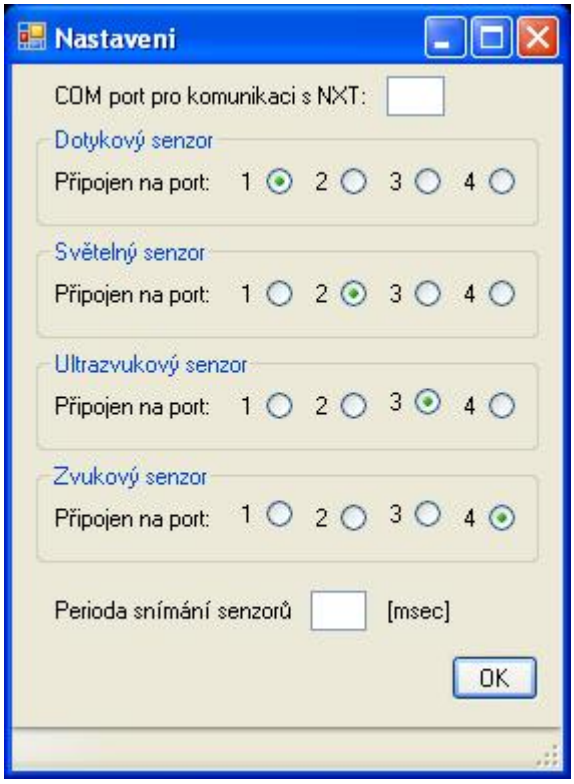
Obr. 20 - Modelová situace s překážkou

Na Obr. 20 je vidět modelová situace původní směr pohybu je vyznačen černě ve vzdálenosti 50 cm je dále vyznačen čárkovaně. V tomto místě začínají také obě žluté čáry značící možný pohyb pokud by byla překážka menších rozměrů a robot by nalezl v některém ze směrů více místa. Po měření se robot srovná do původní polohy a začne couvat poté se otočí doleva a začne vykonávat pohyb. Tímto způsobem by se měl vypořádat s různými typy překážek větších rozměrů, které zachytí čidlem problém nastává u hran objektů a při velkém úhlu natočení vůči překážce. Pro ošetření tohoto problému bylo původně plánováno využití funkce ***TachoCount***, která však nefungovala korektně a tudíž v řešení automatického chodu tato část není implementována. Stejná funkce měla být použita i v remote aplikaci pro sledování polohy řídícího motoru za účelem ošetření možnosti kolize s konstrukcí. Problém s naražením do překážky by byl také řešitelný za použití dotykového čidla které však již bylo použito k ošetření couvání.

5.4 Aplikace sloužící k řízení a monitorování v jazyku C#

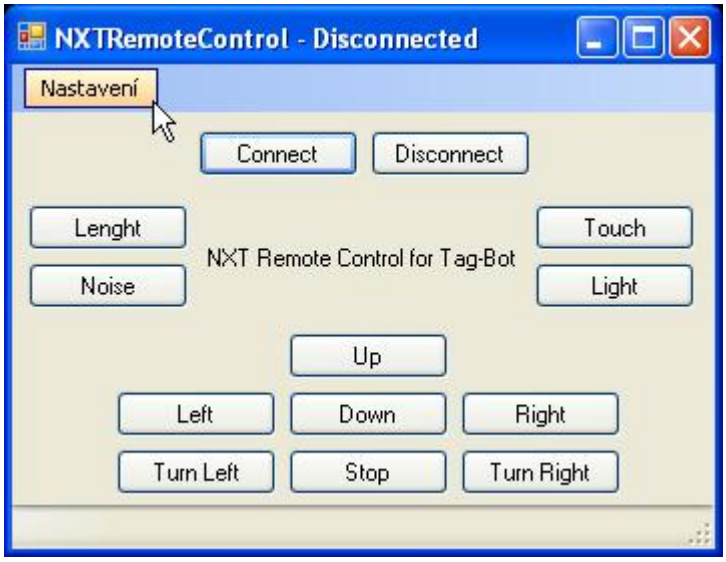
Jedná se o klasickou aplikaci, spustitelnou ve Windows, která slouží k ovládání NXT robota konstrukce Tag bot s upravenou konstrukcí, jak již bylo popsáno výše v této práci. Ke spuštění je třeba mít ve stejném adresáři jak aplikaci NxtRemoteControl.exe tak původní knihovnu NKH.MindSqualls.dll. Pro ovládání robota je nezbytné fungující Bluetooth připojení a některý z řady operačních systémů Windows. Testování i vývoj samotný probíhal na Windows XP. Aplikaci stačí spustit pomocí exe souboru po spuštění se objevý prostředí zachycené na Obr. 22, které slouží řízení nejprve je však třeba pomocí menu **Nastavení** v pravém horním rohu provést několik kroků.

Nejprve je třeba nastavit port pro komunikaci s NXT, ten zjistíte při spárování obou zařízení. Poté následuje nastavení senzorů na jednotlivé porty. Je logické, že při možnosti připojení jednoho senzoru na jeden port se při obsazení portu senzorem port zablokuje pro ostatní senzory. Poslední volbou celého formuláře s nastavením je nastavení periody snímání senzorů. Zadává se číselně v milisekundách celé nastavení je ještě třeba potvrdit tlačítkem ***OK***.



Obr. 21 - Formulář sloužící k nastavení jednotlivých zařízení

Ovládací formulář obsahuje v horní části tři tlačítka: ***Connect***, sloužící pro připojení k NXT, ***Disconnect*** mající opačnou funkci. Po stranách se nalézají tlačítka pomocí kterých můžete získat hodnoty připojených senzorů, jako je vzdálenost překážky ***Length***, hluk okolí ***Noise*** a intenzita osvětlení ***Light*** v místě pohybu robota. Poslední senzor slouží k detekci zda je za robotem nějaká překážka ***Touch*** všechny hodnoty jsou infikovány ve status baru. Dále jsou zde tlačítka pro ovlivňování rychlosti ***Up***, ***Down*** pomocí nichž dochází k pohybu vpřed či vzad, další tlačítka ***Left*** a ***Right*** slouží ke změně směru pohybu. Navíc jsou zde přidána tlačítka pro otočení robota o 90º doprava ***TurnR*** i doleva ***TurnL*** průběh těchto dvou akcí se nedá zastavit. Tyto funkce je však napsány pro zvolenou konstrukci, s její se změnou je třeba upravit i parametry funkce , poslední ovládací tlačítko ***Stop*** slouží k zastavení motorů.



Obr. 22 - Interface sloužící k řízení robota

5.4.1 Kód pro řízení v jazyku C#

Aplikace obsahuje stejně jako knihovna funkce a metody které jsou obsaženy v „handlerech“ obsluhujících jednotlivá tlačítka. Na počátku je vytvořeno několik privátních proměnných spolu s proměnnými ovlivňujícími funkce jednotlivých metod a funkcí.

```
private NxtBrick brick = null;
private NxtBluetoothConnection BtCon;
private NxtMotorSync motorPair;


SByte fast = 0;      //Počáteční rychlost motorů.
int position = 0;    //Pozice otočení přední nápravy.
```

```
SByte lPower = 40;   //Síla pohybu otočení doleva.
SByte rPower = -40;  //Síla pohybu otočení doprava.
SByte power = 50;    //Síla pohybu otočení doprava.
uint angle = 25;     //Úhel otočení.
```

V konstruktoru formuláře je volána metoda zamykající ovladací prvky tak aby uživatel musel nejprve použít položku Nastavení viz Obr. 22.

```
public MainForm()   //Konstruktor formuláře.
      {
            InitializeComponent();

            this.Text = "NXTRemoteControl - Disconnected";
            LockButtons(true);
        }

//Metoda pro vytvoření spojení.
private void btnConnect_Click(object sender, EventArgs e)
      {
            try
            {
                brick.MotorA = new NxtMotor();
                brick.MotorB = new NxtMotor();
                brick.MotorC = new NxtMotor();
                motorPair = new NxtMotorSync(brick.MotorA, brick.MotorC);

                brick.Connect();

                this.Text = "NXTRemoteControl - Connected: " +
                brick.Name;
            }
            catch
            {
                Disconnect();
            }
      }
```

Metoda pro vytvoření spojení stejná jako u kninovny NXTRemoteControl.dll je však bez vytvoření sennzorů a přiřazení jednotlivým portům. Metody jsou si velmi podobné zejména u tlačítek řídících pohyb robota a měření pomocí senzorů proto jsou zde uvedeny pouze příklady.

```
//Metoda pro odbočení vlevo.
private void btnTurnLeft_Click(object sender, EventArgs e)
      {
            brick.MotorB.Run((sbyte)lPower, 120);    //Otočení řízení.
            WaitForCommand(NxtMotorPort.PortB, 10);
            statusLabel.Text = "Aktuální úhel: -140";
            motorPair.Run((sbyte)power, 1800, 0);
            WaitForCommand(NxtMotorPort.PortA, 10);
            motorPair.Brake();                       //Zastavení motorů.
            brick.MotorB.Run((sbyte)rPower, 120);    //Srovní kol.
            WaitForCommand(NxtMotorPort.PortB, 10);
      }
```

Obslužná metoda pro zvyšování rychlosti vpřed obsahuje podmínku ***If***, která zajištuje posílání příkazů v povoleném intemalu 0 - 100% každým kliknutím se zvýší rychlost o 20% uživatel je informovám o rychlosti pomocí textu v status baru. Handler pro snížení rychlosti obsahuje stejný kod pouze se zmenou znaménka v podmínce a inkrementace je nahrazena dektementací.

```
//Metoda obsluhující tlačítko UP.
private void btnUp_Click(object sender, EventArgs e)
        {
            if (fast < 100)
            {
                fast += 20;
                motorPair.Run(fast, 0, 0);
                statusLabel.Text = "Aktuální tychlost: " + fast + "%";
            }
            else
            {
                motorPair.Run(fast, 0, 0);
                statusLabel.Text = "Aktuální tychlost: " + fast + "%";
            }
        }
```

Pohyb vlevo nebo vpravo je proveden pomocí metody řídící se parametry nastavanými pomocí proměnných pozice je opet indikována aktualizací textu ve status baru. Tento údaj je pouze informativní a lze jej považovat za relevantní pouze v případě, že spuštění aplikace proběhlo s řízením ve výchozí poloze. Ošetření pomocí ***TachoCount*** nebylo možné realizov z důvod nefunkčnosti metody.

```
//Metoda obsluhující tlačítko Left.
private void btnLeft_Click(object sender, EventArgs e)
        {
            brick.MotorB.Run(lPower, angle);
            WaitForCommand(NxtMotorPort.PortB, 10);
            position = position - (int)angle;
            statusLabel.Text = "Aktuální úhel: " + position;
        }
```

Tlačítko ***Stop*** a jeho metoda slouží k zastavení motoru A a C při jízdě vpřed nebo couvání.

```
//Metoda obsluhující tlačítko Stop.
private void Stop_Click(object sender, EventArgs e)
      {
            fast = 0;     //Vynulování rychlosti.
            brick.MotorA.Brake();
            brick.MotorC.Brake();
            statusLabel.Text = "Aktuální tychlost: 0";
      }
```

Metody měřících senzorů jsou typově stejné pouze zjištění stavu dotykového čidla je doplněno o podmínku ***If***, která vybírá mezi dvěma texty zobrazovanými ve status baru ovládací aplikace NXTRemotControl.

```
//Metoda pro zjištění stavu dotykového senzoru.
private void btnTouch_Click(object sender, EventArgs e)
      {
            bool? touch =
            ((NxtTouchSensor)GetSensor(typeof(NxtTouchSensor))).IsPressed
            ;
            if (!touch.HasValue)
                return;

            if (touch.Value==true)
                statusLabel.Text = "Touch sensor is pressed.";
            else
                statusLabel.Text = "Touch sensor isnt pressed.";
        }
```

Funkce metody čekání na provedení příkazu je zcela identická s funkcí použitou u knihovny NXTRemoteControl viz 5.1.2.

```
//Metoda obsluhující formulář Nastavení.
private void nastaveníToolStripMenuItem_Click(object sender, EventArgs e)
      {
            Nastaveni nastaveni = new Nastaveni();
            nastaveni.StartPosition = FormStartPosition.CenterScreen;
            if (nastaveni.ShowDialog() == DialogResult.OK)
            {
                  brick = new NxtBrick(nastaveni.ComPort);
                  BtCon = new NxtBluetoothConnection(nastaveni.ComPort);

                  brick.Sensor1 = nastaveni.GetSensorOnPort(1);
                  brick.Sensor2 = nastaveni.GetSensorOnPort(2);
                  brick.Sensor3 = nastaveni.GetSensorOnPort(3);
                  brick.Sensor4 = nastaveni.GetSensorOnPort(4);

                  LockButtons(false);
            }
        }
```

Zamknutí volané v konstruktoru aplikace je realizováno metodou ***LockButtons***, která po spuštění aplice zanechá aktivní pouze položku v menu.

```
//Metoda zamykající ovládací prvky formuláře.
private void LockButtons(bool locked)
      {
            foreach (Control control in Controls)
            if (control is Button)
                    ((Button)control).Enabled = !locked;
      }
```

ZÁVĚR

Problematika řízení robota NXT je dnes realizovatelná téměř ve všech možných dostupných vývojových prostředích s podporou velkého množství programovacích jazyků. Realizace řízení v novém prostředí či jazyku proto není nikterak obtížná, a to především díky dnešním možnostem v programování a také díky jednotlivcům a komunitám, kteří se aktivně podílejí na jeho dalším vývoji. Problémem, jak se ukázalo v této práci, je však spolehlivost a přesnost samotného řízení. Vytvářet veškerou komunikační logiku a jednotlivé funkce opakovaně pro každý jazyk, by však bylo velmi neefektivní, a proto tato práce navázala na již existující řešení. Při výběru byl kladen důraz především na úroveň komplexnosti samotného řešení a jeho dokumentace.

Knihovna NKH.MindSqualls.dll byla použita pro počáteční testování řízení NXT pomocí SW Mathematica a dále pro vytvoření aplikace pro řízení pomocí bluetooth technologie. Výsledkem rozšíření původního řešení je nová knihovna NxtRemoreLibrary.dll, která obsahuje několik funkcí a metod řešící problémy popsané výše. Knihovna je prezentována jako příklad možného rozšíření prostředí použitelných k řízení robotů NXT. Rozšíření je ukázkou možností, které mohou být dále rozvíjeny a dostupnost zdrojových souborů je brána jako samozřejmost. Prostředí Mathematica bylo doposud bráno spíše jako program určený pro různé matematické problémy a algoritmy. Touto prací se však podařilo poukázat na fakt, že za použití prostředníka, v tomto případě jazyka C#, je prostředí Mathematica použitelné i k jiným než početním operacím.

Po zkušenostech získaných s roboty NXT při zpracování této práce vidím jejich využití zejména v praktickém ověření programovacích schopností. Jako příklad řešení automatického modu robota pro pohyb v neznámém prostředí. Při testování a samotné realizaci bylo odhaleno, že některé funkce jako například ***TachoCount*** nefungují správně. Problémy nebyly řešeny jen na úrovni programové, ale i konstrukční. Využití robotů je limitováno především jejich HW možnostmi a použitým programovacím jazykem. Přínosem by určitě byl lepší HW základ a použití platformě nezavilých jazyků .

CONCLUSION

Problems of NXT robot control is nowadays realized nearly in all possible available developing means with support of a many programming languages. That is the reason why realization of controlling in brand new space or language is not difficult. Mainly is that because of contemporary programming possibilities and thanks to individuals and communities who are actively participating on its subsequent development. As this dissertation shows, problem lies in reliability and precision of device. Creating of all communication logic and functions for each language separately would be very ineffective and that is why this dissertation followed in existing solution. By selection accent was putted on stage of solution complexness and its documentation.

Library NKH.MindSqualls.dll was used for initial control testing NXT with support of SW Mathematica and next for creating of an application for controlling via Bluetooth technology. The result of former solution expansion is a new library NxtRemoreLibrary.dll, which contains several functions and methods which solves above mentioned problems. Library is presented as an example of expansion possibilities of controlling NXT robots. Expansion shows options, which can be furthermore developed and availability of source files taken automatically. Mathematica space was until now considered as a program dedicated for several mathematics problems and algorithms. This dissertation shows the fact that when we use mediator (in this care language C#) is Mathematica applicable for other than just numerical operations.

On the base of experiences with robots NXT which I got while working on this dissertation, I see their utilization mainly in practical verification of programming abilities. As example of the solution for robot automatic movement in unknown area. The problems was not solved only in programming level but also in constructional. It was made out, when the robot was tested and SW programmed, that some function e.g. ***TachCount*** are not worked right. The usage of the robots is limited in applied HW and programming language. It would be better use some other HW components and platform independent languages.

SEZNAM POUŽITÉ LITERATURY

[1] FERRARI, Mario, FERRARI, Guilio. Building Robots with LEGO Mindstorms NXT. David Astolfo. [s.l.] : Syngress, 2007. 448 s. ISBN 1597491527.

[2] HANSEN, John C. LEGO Mindstorms NXT Power Programming: Robotics in C. [s.l.] : Variant Press, 2007. 560 s. ISBN 0973864923.

[3] BAGNALL, Brian. Maximum Lego NXT : Building Robots with Java Brains. [s.l.] : Variant Press, 2007. 524 s. ISBN 0973864915.

[4] BOOGAARTS, Martijn, et al. The LEGO MINDSTORMS NXT Idea Book : Design, Invent, and Build. [s.l.] : No Starch Press, 2007. 350 s. ISBN 1593271506.

[5] http://www.mindsqualls.net/

[6] http://www.hitechnic.com/

[7] http://www.teamhassenplug.org/

[8] http://www.ortop.org/

[9] http://www.thenxtstep.blogspot.com/

[10] http://www.nxt-mindstorms.com/

[11] http://www.botmag.com/

[12] http://www.cs.wikipedia.org/

[13] http://www.nxtasy.org/

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

NXT	Význam první zkratky.
SW	Programové vybavení, aplikace, program, data (Software).
HW	Fyzicky existující technické vybavení (Hardware).
PC	Osobní počítač (Personal Computer).
OS	Open Source
PDA	Malý kapesní počítač, osobní digitální pomocník (Personal Digital Assistant).
USB	Univerzální sériová sběrnice (Universal Serial Bus)
NXT-G	Grafický jazyk využívající programování pomocí funkčních bloků od společnosti LEGO® Mindstorms® NXT system.
FLL	First Lego League http://www.firstlegoleague.org/
brick	„kostka“ – řídící jednotka robota
IDE	Funkce programovací prostředí, jež zobrazuje uživateli možnou syntaxi zdrojového kódu v produktech společnosti Microsoft označována IntelliSence.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: TABULKA VLASTNOSTÍ PROGRAMOVACÍCH PROSTŘEDÍ

Features	NXT-G	RoboLab 2.9	NBC	NXC	RobotC	NI LabVIEW Toolkit	leJOS NXJ	pbLua	LEJOS OSEK
Typ jazyka	Grafický	Grafický	Assembly	C-like	C	Grafický	Java	Lua	ANSI C
Firmware	Standard	Standard	Standard	Standard	Standard	Standard	Vlastní	Vlastní	Vlastní
IDE	Ano	Ano	Ano	Ano	Ano	Ne	plugin pro Eclipse a NetBeans.	Ne	Eclipse CDT(GCC +ATMEL)
Windows	Ano	Ano	Ano	Ano	Ano	Ano	Ano	Ano	Ano
Mac OSX	Ano	Ano	Ano	Ano	Ne (zatím)	Ano	Ano	Ano	Ne
Linux	Ne	Ne	Ano	Ano	Ne	Ne	Ano	Ano	Ne (možná)
Events	Ne	Ano	Ne	Ne	Ano	Ne	Standard Java events	?	Ano (OSEK RTOS)
Multithreading	Ano	Ano	Ano	Ano	Ano	Ano	Ano	?	Ano (OSEK RTOS)
Bluetooth Brick to PC	Ano	Ne	Ano	Ano	Ano	Ano	Ano	Ano	Ano
Bluetooth Brick to Brick	Ano	Ne	Ano	Ano	Ano	Ano	Ano	Ano	Ne (zatím)
Bluetooth to Other Device	Ne	Ne	Ne	Ne	Ano	Ne	Ano	Ano	Ne
I2C Support	Nepřímo	Ano	Ano	Ano	Ano	Ano	Ano	Ano	Ano
File System	Ano	Ano	Ano	Ano	Ano	Ano	Ano	Ne (zatím)	není plánován
Floating Point	Ne	Ano	Ne	Ne	Ano	Ne	Ano	pouze částečně	Ano
Datalog	Ne	Ano	Ne	Ne	Ano	Ne	Ano	Ne	Ne (zatím)
WWW	NXT	Educational Version of NXT	BricxCC	BricxCC	RobotC	LabVIEW toolkit	leJOS	pbLua	LEJOS OSEK