Ovládání laboratorního modelu robota Mindstorms

Control of laboratory model Mindstorms robot

Bc. Roman Matušů

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. Roman MATUŠŮ**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Ovládání laboratorního modelu robota Mindstorms NXT (tribot) pomocí PC**

Zásady pro vypracování:

1. **Proveďte literární studii softwarového a hardwarového vybavení robota NXT.**
2. **Proveďte studii programovacího jazyka, možnosti komunikace atd (Java, DotNet).**
3. **Proveďte návrh ovládacího programu (možno i webové aplikace).**
4. **Proveďte návrhy laboratorních úloh a zpracujte pro ně dokumentaci.**

Rozsah práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **PERDUE, David J. THE UNOFFICIAL LEGO MINDSTORMS NXT INVENTOR S GUIDE. Megan Dunchak; Christina Samuell. 1st edition. San Francisco : No Starch Press, Inc, c2008. 296 s. ISBN 978-1-59327-154-1.**
2. **ASTOLFO, Dave, FERRARI, Mario, FERRARI, Giulio. BUILDING ROBOTS WITH LEGO MINDSTORMS NXT. Audrey Doyle. 1st edition. Burlington : Syngress Publishing, Inc, c2007. 447 s. ISBN 978-1-59749-152-5.**
3. **SCHOLZ, Matthias Paul. Advanced NXT: The Da Vinci Inventions Book. Jennifer Whipple. Apress, c2007. 369 s. ISBN 978-1-59059-843-6.**
4. **GASPERI, Michael, HURBAIN, Philippe, HURBAIN, Isabelle. Extreme NXT : Extending the LEGO MINDSTORMS NXT to the Next Level. Susannah Davidson Pfalzer. [s.l.] : Apress, c2007. 286 s. ISBN 978-1-59059-818-4.**
5. **BOOGAARTS, Martijn, et al. THE LEGO MINDSTORMS NXT IDEA BOOK : design, invent, and built. Nancy Sixsmith, Megan Dunchak. 1st edition. San Francisco : No Starch Press, Inc, c2007. 344 s. ISBN 978-1-59327-150-3.**
6. **BAGNALL, Brian. Maximum LEGO NXT : Building Robots with Java Brains. Editied by Sylvia Philipps. 1st edition. Canada : Variant Press, c2007. 505 s. ISBN 978-0-9738649-1-5.**

Vedoucí diplomové práce: **Ing. Roman Šenkeřík**
Ústav aplikované informatiky

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

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

Ve Zlíně dne 20. února 2008

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

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

ABSTRAKT

Práce pojednává o hardware a software robota NXT MINDSTORMS. V teoretické části se čtenář seznámí s hardwarovým vybavením robota, použitou technologii a způsobu komunikace řídící jednotky s okolím, ať už pomocí kabelu nebo bezdrátové technologie Bluetooth. Dále se dozví o programových možnostech řídící jednotky a postup její aktualizace. Teoretická sekce také zahrnuje existující možnosti vytváření programů v různých vývojových prostředích. Praktická část práce je rozdělena do dvou částí. V první části se zaobírá vývojem Windows aplikace, která ovládá robota pomocí technologie Bluetooth s využitím jazyka NXT#. V druhé části jsou rozpracovány a popsány dvě laboratorní úlohy s využitím robota NXT Tribot.

Klíčová slova: NXT, MINDSTORMS, NXT-G, NXT#, robot, BricxCC, RobotC, LeJOS, senzor, Bluetooth

ABSTRACT

This master thesis deals with hardware and software of NXT MINDSTORMS robot. Theoretical part is focused on the hardware and software equipments of robot, used technology and methods of communication between central unit and PC either by cable or via Bluetooth. Next it presents programmatic possibilities and how to update the central unit. The theoretical part also includes existing possibilities of creating the control program in the different development applications. The practical part is separated into two parts. The first part is focused on the development of Windows application, for the purpose of controlling of the robot via Bluetooth technology by means of NXT# language. The second part contents defining and description of two laboratory tasks with used robot NXT Tribot.

Keywords: NXT, MINDSTORMS, NXT-G, NXT#, robot, BricxCC, RobotC, LeJOS, sensor, Bluetooth

Touto cestou bych chtěl poděkovat mému vedoucímu diplomové práce, kterým byl Ing. Roman Šenkeřík za odborné vedení, poskytování rad a pomoci během mé práce. Dále bych chtěl poděkovat mému kamarádovi Tomáši Valovému a jeho otci za umožnění využívání jejich grafického studia. Poděkování zaslouží i pracovníci kopírovacího pracoviště společnosti CENTROPROJEKT a.s., kteří mi umožnili laborování s barevným podkladem pro světelný senzor. V neposlední řadě bych chtěl také poděkovat svým rodičům za pomoc a morální podporu během celého studia.

Prohlašuji, že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků, je-li to uvolněno na základě licenční smlouvy, budu uveden jako spoluautor.

Ve Zlíně ……

Podpis diplomanta

OBSAH

ÚVOD

Existence lidí jde ruku v ruce s neustálým vývojem. Člověk se vždycky snažil o to, aby si ulehčil práci. S vývojem techniky nabral pokrok velkou rychlost. Za poměrně krátké období dospěla technika takového rozmachu, že se její produkty staly každodenním společníkem téměř každého člověka. Dalo by se říct, že už máme všechno, jen nemáme a asi ani nikdy nebudeme mít dost. Vývoj neustále pokračuje. Rodí se děti, které už mají dnešní pokrok v krvi. Malé děti dokážou ovládnout všechny funkce mobilního telefonu během několika minut, přičemž nepotřebují ani návod. To se vůbec nedá říct o spoustě lidí středního věku, kteří jsou většinou rádi, když si umí používat alespoň základní funkce jako je volaní a psaní SMS.

V roce 1949 vznikly první kostičky, které umožňovaly pevné, ale rozebíratelné stavby. Tento vynález se nevídaně ujal ve světě hračkářství. Teprve však v roce 1958 expandoval tento výrobek do téměř celého světa pod názvem LEGO. Dobře si pamatuji, když jsem já jako dítě dostal první kyblík těchto zázračných kostiček. Od té chvíle jsem nedělal nic jiného, než stavil různé autíčka a vymýšlel spoustu konstrukcí, abych docílil největší univerzálnosti. I když byla stavebnice velmi variabilní, brzy vyčerpala všechny nápady a stala se tak pro dětskou fantazii nedostatečná. Ovšem tuto skutečnost si dobře uvědomovali i samotní tvůrci a tak přišli s pokročilejší verzí LEGO Technik. Tahle stavebnice mne opět dokázala zabavit na poměrně dlouhou dobu. Co víc si mohl malý kluk přát víc, než postavení si vlastního autíčka, kde se přední kola otáčela pomocí volantu jako u opravdového auta. Autíčku už chyběl jen vlastní motor, který jsem získal z rozebraného policejního auta na tužkové baterie a pomocí provázku jsem jeho točivý moment převáděl na osu zadních kol.

To byla doba, kdy většina dětí neměla přístup k počítačům. Dnes už je všechno jinak a tomu se přizpůsobila i stavebnice LEGO svými produkty. Aktuálně je to produkt MINDSTORMS NXT, kde už se využívá moderní technologie v plném rozsahu. Jako centrum inteligence slouží řídící jednotka, která z tohoto produktu vytvořila hračku nejen pro děti, ale i pro dospělé. Díky Open Source licenci se dosáhlo velkého množství využití pro tento produkt. Studenti se zde můžou naučit základům robotiky nebo automatizace celkově. Stačí jen znovu rozvinout fantazii jako v dětství.

I. TEORETICKÁ ČÁST

1 HARDWAROVÉ A SOFTWAROVÉ VYBAVENÍ ROBOTA NXT

Veškeré vybavení je možno zakoupit ve stavebnici LEGO MINDSTORMS, která obsahuje součástky potřebné k postavení libovolného robota. Obsahem je také CD s uživatelským programem, ve kterém je možno vytvářet jednoduché programy pro robota.

Obrázek 1: Lego mindstorms NXT kid

Stavebnice dále nabízí jednoduchou trasu vykreslenou na tvrdém papíru velikosti A1. Trasu představuje černá linie ve tvaru oválu. Sledování této linie je možné naučit například Tribota, který má pro takový úkol správně umístěný potřebný senzor.

Obrázek 2: Model tribot

Lego tímto produktem otevírá cestu novým nápadům a rozvíjí lidskou fantazii ve spojení s technikou. Důkazem je například ztvárnění zvířecího robota škorpiona.

Obrázek 3: Model scorpio

Protože pod názvem robot si většina lidí představí napodobeninu člověka, kterou ovládá spoustu motorků a různých svíticích senzorů, nesmíme zapomenout na další oblíbený model, kterým je humanoid.

Obrázek 4: Model humanoida

1.1 Hardwarové vybavení robota NXT

Začátek práce s jakýmkoli technickým zařízením si nejprve vyžaduje důkladné seznámení s jeho částmi a rovněž s funkcemi jednotlivých částí.

Systém LEGO Mindstorms umožňuje sestavovat a taky programovat roboty postavené z hodně známých LEGO kostek. Vývoj robotů od LEGA prošel od počátku svého vývoje velkými změnami. Oproti předchozímu modelu RCX který byl spíše určený ke spojení se systémem LEGO CITY, je model NXT určen ke spojení se systémem LEGO TECHNIC.

Obrázek 5: Starší model RCX

I když oba zmíněné systémy je možno za pomoci speciálních přechodových kostek kombinovat. Nový model NXT (Obrázek 6) nabízí spoustu nových funkcí a obsahuje úplně nové druhy senzorů jako je zvukový (3) a ultrazvukový senzor (5). Dále disponuje 3 servomotory (6), tlakovým senzorem (2) a světelným senzorem (4). Všechny tyto komponenty sdružuje srdce celé technologie, kterou je řídící jednotka (1).

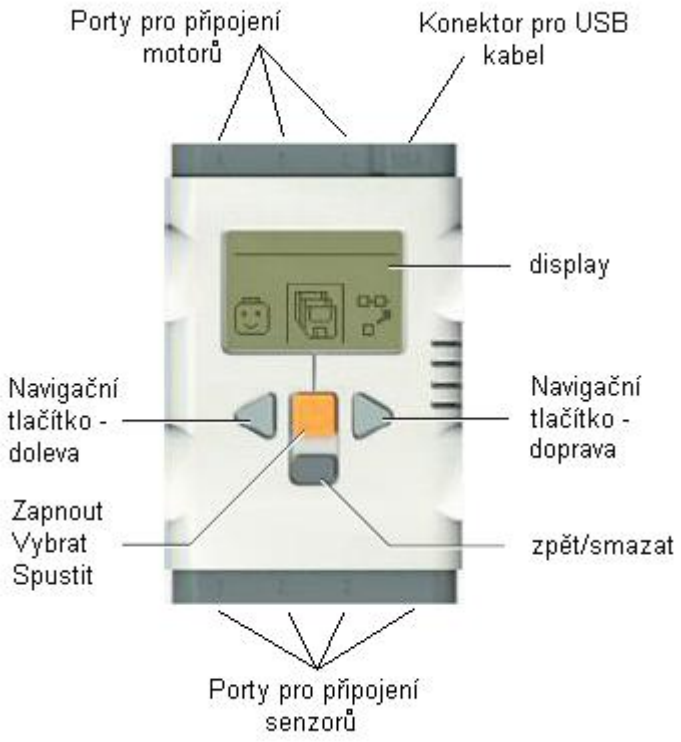
Obrázek 6: Technologie NXT

Bohužel postrádá teplotní senzor, který se neprodává se ani jako příslušenství, jako tomu bylo u starého systému. Oproti předchozímu modelu RCX už zde nenajdete ani samostatný senzor otáček, který už není potřebný, neboť každý ze tří servomotorů ho má už zabudovaný a umožňuje tak přesnější řízení robota. Také způsob komunikace s PC zase o něco dospěl. Na rozdíl od starého systému, který používal ke komunikaci IR věž, umožňuje NXT využití technologie Bluetooth nebo kabelu s konektorem USB, který je součástí stavebnice.

Obrázek 7: Porovnání jednotky RCX a NXT

1.1.1 Řídící jednotka NXT

Řídící jednotka je mozkem celého MINDSTORMS robota. Díky ní sestavený robot ožívá a může vykonávat různé operace.



Obrázek 8: Popis řídící jednotky

Na první podhled uvidíme nenápadnou bílou krabičku, na jejímž povrchu je displej a čtyři funkční tlačítka. Největší z nich je oranžové tlačítko, které slouží k zapnutí robota, potvrzování výběru v menu a spouštění jednotlivých programů. Dvě trojúhelníkové tlačítka umožňují pohyb v grafickém menu, které se zobrazuje na displeji. Poslední obdelníkové šedé tlačítko umožňuje akci zpět během pohybu v menu a taky slouží pro vypnutí jednotky. Uvnitř tajemné krabičky se skrývá vlastní elektronika (Obrázek 9) a ze spod je umístěno 6 tužkových baterií typu AA nebo lithiová dobíjecí baterie.

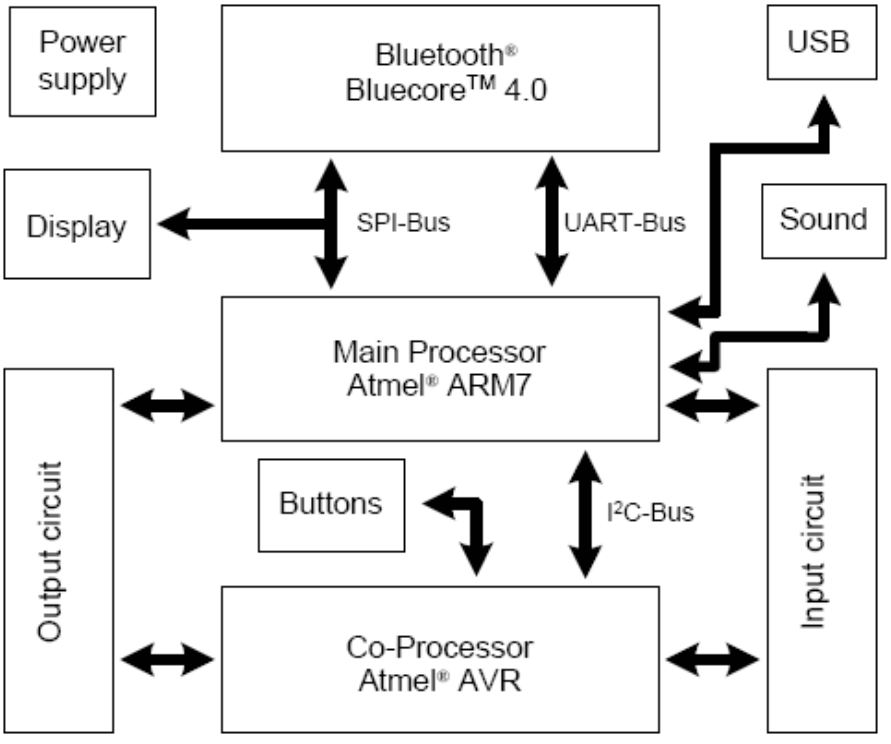
Obrázek 9: Elektronika řídící jednotky

Na bocích dvou užších stran (podle Obrázek 8) krabičky jsou rozmístěny porty. Na horní straně jsou to porty A, B a C pro připojení servo-motorů a taky konektor pro USB kabel. Na spodní straně jsou pak porty 1, 2, 3 a 4 pro připojení senzorů robota. Jeden z těchto portů zahrnuje normu IEC61158, typ 4 / EN50170, což nám do budoucna umožní připojení senzorů, vyžadujících vysokou přenosovou rychlost. Všechny kabely jsou šesti-vodičové a jsou zakončeny podobným konektorem, jaký známe z telefonních kabelů, s tím rozdílem, že má šest drážek a pérko po zajištění ve zdířce je umístěno na okraji konektoru.

Technická specifikace:

- 32-bitový mikroprocesor ARM7
    - 48 MHz
    - 256 KB flash paměť
    - 64 KB paměti RAM
- 8-bitový co-procesor AVR
    - 4 MHz
    - 4 KB flash paměť
    - 512 B paměti RAM

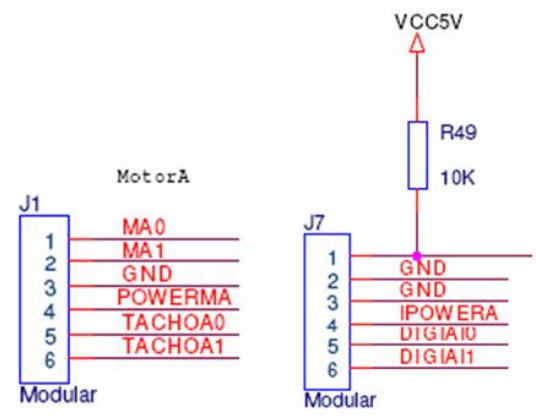
- 4 vstupní porty a 3 výstupní porty využívající 6-ti vodičovou digitální platformou
- LCD maticový displej o velikosti 100×64 pixel
- Vestavěný reproduktor
    - 8 kHz kvalita zvuku
    - 8-bitový zvukový kanál
    - Frekvenční rozsah 2-16 kHz
- Bluetooth třídy II v2.0 CSR BlueCoreTM 4 + EDR (Enhanced Data Rate)
    - Podporuje komunikaci po sériovém portu (SSP)
    - 47 KB vnitřní paměti RAM
    - 8Mbit vnější paměti FLASH
    - 26 MHz
- USB 2.0 port (přenos 12Mbit/s)
- Elektrické zdroje: nabíjecí lithiová baterie nebo 6AA článků
- Vývojové nástroje pro známé platformy (Windows, MAC OS, Linux)



Obrázek 10: Blokové schéma řídící jednotky

1.1.2 Kabely spojující jednotku s periferiemi

Stavebnice MINDSTORMS NXT obsahuje 7 kabelů. Každý kabel je složen z 6 vodičů a je na obou stranách zakončen koncovkou, která je podobná známému telefonnímu konektoru RJ12. Kabely jsou k dispozici ve 3 délkách. Kterýkoliv kabelem můžeme spojit libovolné zařízeni s řídící jednotkou. Komunikace s jednotkou je umožněna dvojím způsobem a to buď analogově, nebo digitálně.



Obrázek 11: Schéma zapojení kabelů

1.1.2.1 Redukční kabel pro NXT

Protože senzory dodávané k starší jednotce RCX pracují na téměř stejném principu jako nové senzory pro jednotku NXT, tak je můžeme využívat a nahradíme tak absenci třeba tepelného senzoru. Ovšem původní senzory k jednotce RCX používají jiné konektory, bylo nutné vytvořit redukční kabel, který je umožní připojit i k nové řídící jednotce NXT

Obrázek 12: Redukční kabel pro senzory RCX

Zařízení	Kompatibilita
NXT jednotka + všechny NXT senzory	ANO
NXT jednotka + všechny RCX senzory	ANO
RCX jednotka + tlakový a svěrelný senzor	ANO
RCX jednotka + zvukový a ultrazvukový senzor	NE
RCX jednotka + NXT jednotka	NE

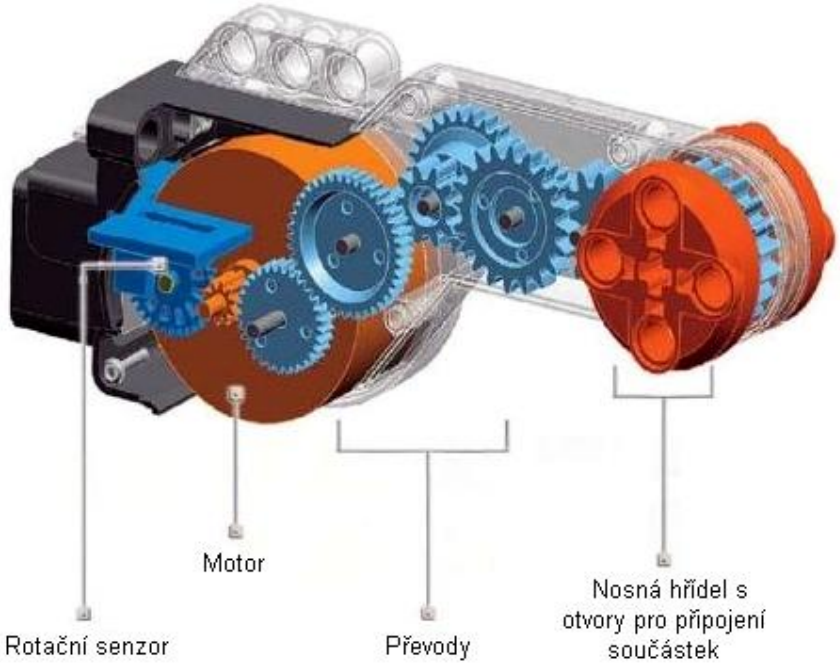
Tabulka 1: Kompatibilita RCX a NXT zařízení

1.1.3 Výstupní zařízení

V tomto případě jsou výstupní zařízení 3 motory. Jedná se prakticky o servomotory, které lze pomocí řídící jednotky jednoduše ovládat.

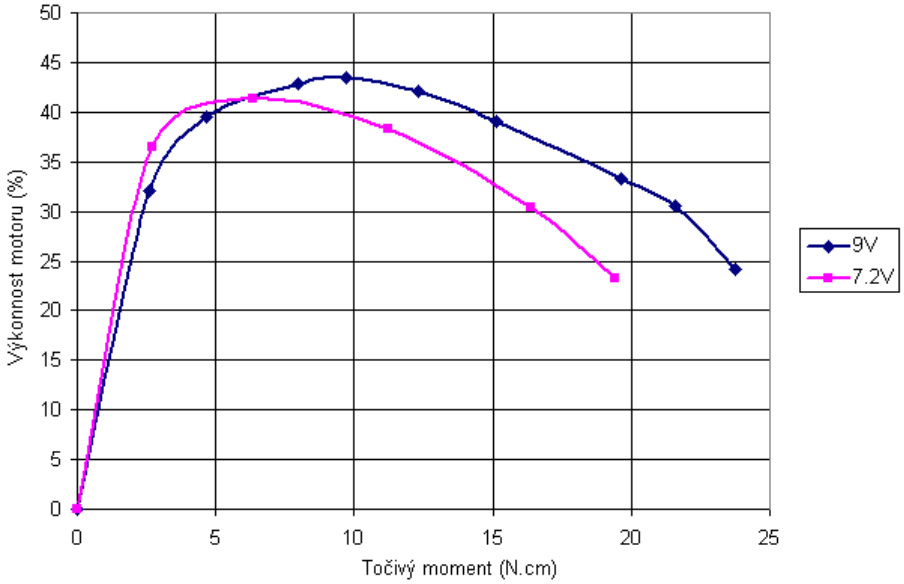
Obrázek 13: Servomotor NXT

Servomotor je podstatně větší oproti předchozímu motoru používaném s jednotkou RCX. Toto zvětšení způsobilo použití převodového ústrojí a vestavění rotačního senzoru. Díky těmto změnám se docílilo zvýšení síly motoru a taky zvýšení spolehlivosti. Servomotor dosahuje rychlosti až 200 otáček za minutu. Další velkou předností tohoto servomotoru je možnost řízení otáčení s přesností až 1°. Rotační senzor taky umožní snadnější synchronizaci dvou motorů nebo lze jeden motor použít jako ovladač jiného motoru.



Obrázek 14: Ústrojí servomotoru

Spolehlivé využívání servomotorů hodně závisí na stavu baterie. Slabá baterie ovlivní jednak rychlost otáčení, ale hlavně přesnost při požadavku na otočení o daný úhel.



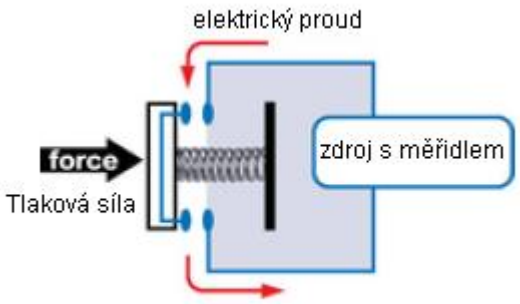
Obrázek 15: Závislost výkonnosti motoru na točivém momentu při změně napětí

1.1.4 Vstupní zařízení

Vstupními zařízeními se v tomto případě míní různé senzory, které poskytují jednotce informace z okolí. Ve stavebnici jsou k dispozici 4 druhy senzorů a to zvukový senzor, světelný senzor, ultrazvukový senzor a tlakový senzor.

1.1.4.1 Tlakový senzor

Jedná se v podstatě pouze o jednoduché tlačítko, které vrací hodnotu pravda nebo nepravda v závislosti na stavu tlačítka – stlačeno / nestlačeno. Princip je v rozpojeném elektrickém obvodu, který se po stisku dotykové plochy uzavře.



Obrázek 16: Princip tlakového senzoru

Tlakový senzor neprošel oproti jeho předchůdci žádným velkým rozdílem. Jediný rozdíl spočívá k přizpůsobení pro LEGO Technic.

Obrázek 17: Tlakový senzor

1.1.4.2 Světelný senzor

Jedná se o základní zařízení umožňující robotu vidění, i když jen v omezené míře. Rozeznává pouze rozdíl mezi tmavým a světlým podkladem. V senzoru není zabudováno rozeznávání barev jako je červená, modrá, apd., ale vrací pouze intenzitu odraženého světla. Návratová hodnota ze senzoru je v rozmezí 0 (tmavý) až 100 (velmi světlý). Tato hodnota je značně relativní, protože závisí na intenzitě osvětlení. Z tohoto důvodu může senzor pracovat ve dvojím režimu a to buď v aktivním, nebo pasivním. Aktivní režim spustí přisvětlení diodou, která emituje červené světlo a tím se docílí zesílení odrazivosti tmavých předmětů.

Obrázek 18: Světelný senzor

1.1.4.3 Zvukový senzor

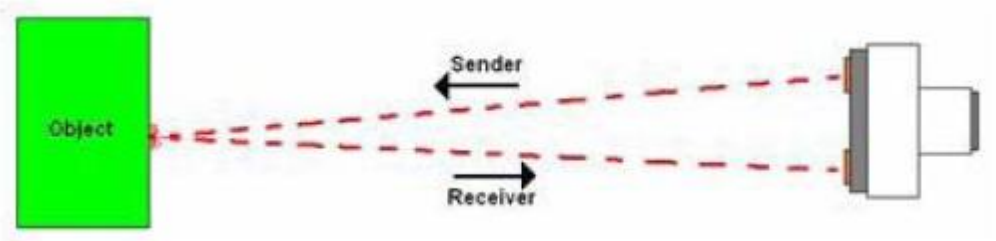
Zvukový senzor nahrazuje robotovi sluch. Umí pracovat ve dvou režimech. První režim se nazývá „adjusted decibel (dbA)“. V tomto režimu senzor zachytává frekvenční rozsah zvuku pro člověka slyšitelného. V druhém režimu, „standard decibel (db)“, zachytává všechny zvuky, i ty, které člověk neslyší.

Obrázek 19: Zvukový senzor

Zvukový senzor se nejčastěji používá pro spouštění různých příkazů, kdy můžeme například tlesknutím ruky spustit funkci motoru a pak ji dvojím tlesknutím zastavit. Stejně tak může reagovat na tichý a hlasitý povel. Senzor vrací procentuální hodnotu hlasitosti zvuku do 90dB (100%). Pro představu při běžné řeči bude senzor vracet hodnotu od 10% do 30%.

1.1.4.4 Ultrazvukový senzor

Ultrazvukový senzor je novým senzorem ve stavebnici MINDSTORMS. Pracuje na principu odražení vyslaného vysokofrekvenčního signálu od překážky a dopadu zpět na senzor. Doba od vyslání signálu do jeho přijetí je úměrná vzdálenosti překážky.



Obrázek 20: Princip ultrazvukového senzoru

Senzor tak robotu umožňuje prostorové vidění. Dokáže detekovat překážky do vzdálenosti cca 255 cm. Lze jej použít rovněž k rozeznávání pohybu před senzorem. Bohužel dva a více robotů operujících v blízkém okolí se navzájem ovlivňují.

Obrázek 21: Ultrazvukový senzor

1.1.4.5 Doplňkové senzory

Robota je možno doplnit i senzory dodávanými firmami třetích stran. Jednou s nejvýznamnějších firem zabývajících se vývojem senzorů pro NXT je firma HITECHNIC.

1.1.4.5.1 NXT Compass sensor

Rozšíří robota o schopnost navigace podle zemského magnetického pole. Jedná se o digitální kompas, který vrací aktuální pozici od 1° do 359 stupňů od severního pólu. Pracuje ve dvou režimech. Ve čtecím režimu vrací polohu a v kalibračním režimu načítá magnetické pole kolem sebe a snaží se anulovat ovlivnění čidla magnetickým polem motorů nebo baterií.

Obrázek 22: Digitální kompas

1.1.4.5.2 NXT Color sensor

Jedná se o vylepšení světelného senzoru od společnosti LEGO. Detekuje daný rozsah barev a pak vrací jméno barvy. Pokud je intenzita mimo definovaný rozsah, vrátí číselnou hodnotu intenzity.

Obrázek 23: Color sensor

1.1.4.5.3 NXT IRSeeker

Jedná se o senzor složený z pole infračervených detektorů umožňující sledování pohybujícího se předmětu v rozsahu 135°. Senzor vrací směr pohybu a relativní sílu signálu z vyzařujícího předmětu. Ve spojení s koulí vyzařující IR záření je možno naučit robota například hrát fotbal.

Obrázek 24: Detektro IR záření

Obrázek 25: Koule vyzařující IR záření

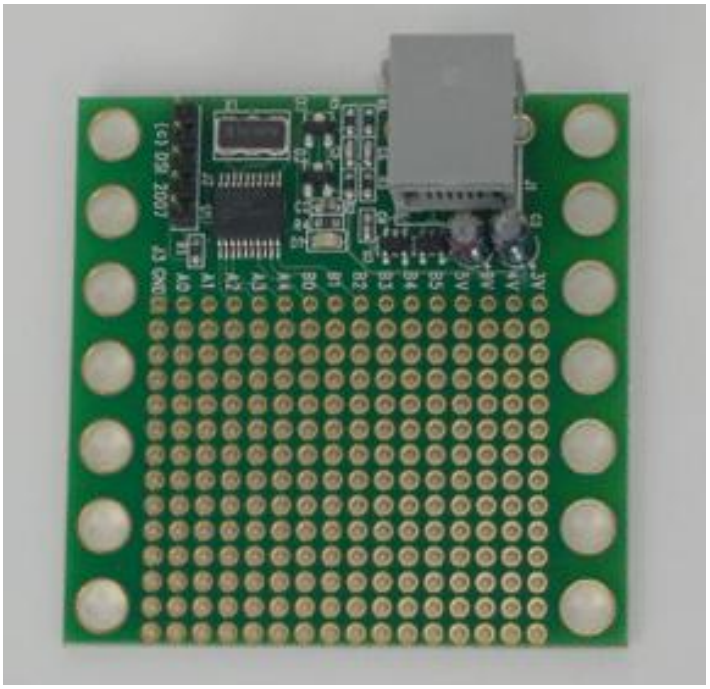
1.1.4.5.4 NXT Gyro

Je senzor pracující na bázi gyroskopu. Na základě pohybu robota v prostoru vrací stupně rotace.

1.1.4.5.5 NXT Prototype board

Firma HITECHNIC nabízí i moduly, které umožňují vytváření vlastních senzorů a jejich testování. Vyrábí se ve dvou provedeních a to pájecí a konektorové. Výhodou konektorových je možnost opětovného využívání. Pájecí jsou vhodné spíše pro konečné zhotovení senzoru.

Obrázek 26: Konektorová deska pro tvorbu vlastních senzorů



Obrázek 27: Pájecí deska pro výrobu vlastního senzoru

1.1.5 Komunikace s PC

Komunikace s PC je umožněna dvěma způsoby a to buď prostřednicím kabelu USB nebo bezdrátové technologie Bluetooth.

1.2 Softwarové vybavení robota NXT

Aby robot ožil, potřebuje k tomu nějaký program, který ho naučí využívat své schopnosti. Jako u každého programovatelného zařízení, i u NXT záleží na dobře vymyšleném softwaru. Bez něj by byl robot NXT jen kouskem plastu nebo chcete-li modelovou skládačkou vystavenou ve výkladní skříni.

K tomu, aby robot ožil, potřebujeme dvě základní věci. První je operační systém, kterým je v tomto případě firmware. Druhou podmínkou je nějaký systém ukládání dat, abychom do robota mohli vkládat různé programy. K tomuto účelu nám poslouží NXT file system. No a třetí, už ne tak důležitou věcí, jsou různé administrativní nástroje. Základním nástrojem u standardního NXT LEGO firmwaru je nástroj „Try Me“, který nám umožní vyzkoušení připojených senzorů a motorů. Dalším nástrojem je využívání nějakého programovacího softwaru, který nám umožní vytvářet vlastní ovládací programy.

1.2.1 Firmware

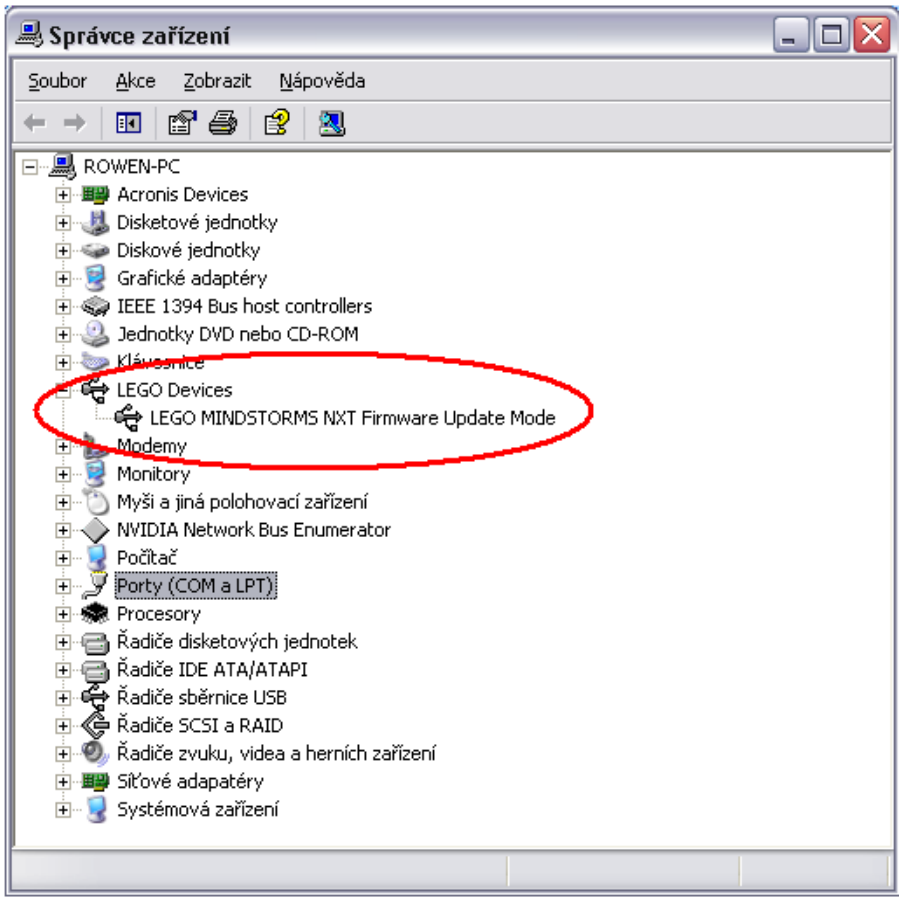
Řídící jednotka v sobě obsahuje ovládací program, který se nazývá firmware. Můžeme říct, že je to jakýsi operační systém robota. Je uložený v paměti typu flash, která se po vypnutí jednotky ani po vytažení baterií nesmaže. Firmware je dodáván už přímo od výroby. Podléhá ovšem neustálému vývoji a tak je možné si jej stáhnout z webových stránek výrobce a nahrát do řídící jednotky. Přehrávání firmwaru můžeme provádět opakovaně až na hranici, kterou vydrží paměť. U flash paměti se počet přehrání uvádí kolem 70000, poté už je paměť nepoužitelná a je nutné ji vyměnit přepájením.

K dispozici jsou i firmwary třetích stran, které se dají uplatnit při vývoji složitějších aplikací. Tyto firmwary mají většinou jednu velkou výhodu, kterou je jejich velikost zabírající v paměti řídící jednotky. Zpravidla jsou tyto firmwary podstatně menší, než originální firmware, čímž vývojáři nabízí možnost využití větší paměti pro své aplikace.

Pokud dojde k chybě při přeflashování firmwaru, disponuje řídící jednotka nouzovým režimem, který se spustí podržením tlačítka ze spod jednotky po dobu 5 vteřin. Tlačítko je schováno v útrobě první upevňovací díry hned pod USB konektorem.

Obrázek 28: Tlačítko pro obnovu firmware

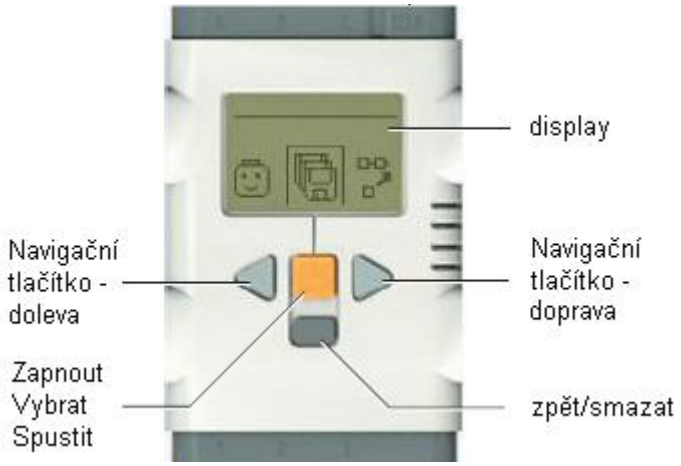
Při správně provedené operaci se řídící jednotka připraví na nahrání originálního firmware pomocí softwaru LEGO MINDSTORMS NXT. Skutečnost, že je jednotka připojena k PC poznáme podle správce zařízení ve svém operačním systému.



Obrázek 29: Nouzový režim pro opětovné nahrání FW

1.2.1.1 Standardní LEGO NXT firmware

Tento druh firmwaru je dodávaný přímo se stavebnicí LEGO MINDSTORMS. Uživateli nabízí přehledné grafické zobrazení základních funkcí firmwaru s intuitivním ovládáním, které je podobné běžným mobilním telefonům.



Obrázek 30: Popis řídící jednotky

Aktuální verze je k dispozici na stránkách http://mindstorms.lego.com a v současné době je to verze 1.05 vydaná v září 2007.

Obrázek 31: Hlavní ikony přístupového menu

- Settings – V této položce je možno měnit hlasitost vestavěného reproduktoru, nastavovat vypínání jednotky při nečinnosti a mazat soubory. Ve většině případů práce s jednotkou tuto položku nevyužijeme.
- Try Me – Obsahuje sérii jednoduchých programů, které poslouží při otestování každého senzoru. Velmi dobře poslouží také při zjišťování stavu senzoru v dané situaci.
- My Files – Zde je možné nahlédnout k uloženým programům a ostatním souborům jako jsou zvuky apd.
- NXT Program – Pod touto položkou nalezneme jednoduchý nástroj pro vytváření programů pomocí příkazových bloků. Takže nemusíme ani využívat externí programovací software.

- View – Umožní náhled na hodnoty, které senzor vrací řídící jednotce. Velmi užitečný nástroj při programování vlastních aplikací. Pomocí něj lze zjistit například o kolik stupňů se musí otočit kolo, aby došlo k otočení robota o 180°C nebo jakou intenzitu vrací která barva.
- Bluetooth – Je to nástroj pro nastavení bluetooth. Tedy slouží k zapnutí bluetooth, vyhledání okolních zařízení a připojení k nim.

1.2.1.2 RobotC firmware

Tento firmware nabízí skupina vývojářů, která se nazývá RobotC. Instalaci firmware můžeme provést až po nainstalování vývojového prostředí. Firmware je součásti vývojového prostředí, jedinou podmínkou k instalaci je update ovladačů jednotky NXT pro USB port. Na první pohled nezaznamenáme žádný větší rozdíl. Rozšiřuje nastavení bluetooth zařízení a přidává animaci ikon menu.

Obrázek 32: Logo RobotC

1.2.1.3 LeJOS

LeJOS je další alternativní firmware, který je distribuován spolu s celým vývojovým balíčkem. Je to opět upravený firmware tentokrát pro psaní aplikací pomocí javy.
K nahrání tohoto firmware je nutná instalace Java SE Development Kit a knihoven pro USB komunikaci.

Obrázek 33: Logo leJOS

1.2.1.4 NBC/NXC firmware

Jedná se o firmware k vývojovému prostředí BricxCC. Ve firmware nejsou nainstalované žádné zvuky a není ani funkční položka Try Me. Jeho předností je využívání více rozměrných polí, nativní posun motorů a čekací operace.

Obrázek 34: Logo NBC/NXC firmware

1.2.1.5 Ostatní firmware

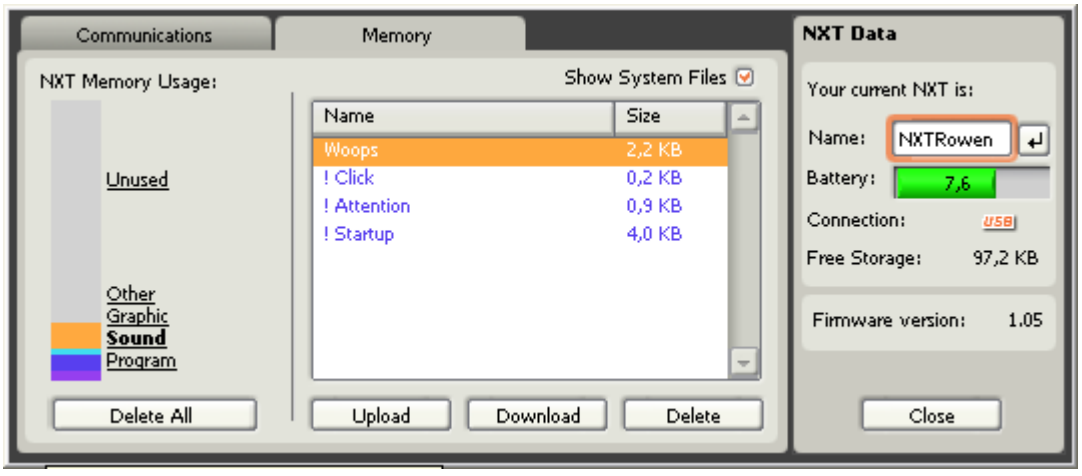
Kdybychom měli probírat všechny možné dostupné firmwary pro řídící jednotku NXT, vydalo by to ještě minimálně na jednu práci. Proto jen zmíním ostatní existující, které jsou dostupné na uvedených webech:

NXTGCC: http://nxtgcc.sf.net/
pbLUA: http://www.hempeldesigngroup.com/lego/pbLua/
TinyOS: http://nxtmote.sf.net/

1.2.2 Souborový systém NXT

Flash paměť jednotky NXT využívá základní souborový systém, který se označuje TOC (Table Of Contents). Využívá se pro ukládání stálých dat, jako jsou programy a datové soubory. Umožňuje spravovat 63 položek v paměti.

Přístup k paměti je umožněn buď pomocí menu, a nebo pomocí utility obsažené v programu LEGO MINDSTORMS NXT.



Obrázek 35: Utilita pro správu paměti jednotky NXT

2 MOŽNOSTI PROGRAMOVACÍHO JAZYKA A KOMUNIKACE

V této kapitole probereme nejpoužívanější vývojové prostředí pro tvoření programů ovládajících řídící jednotku a tím nám umožní, aby nás robot poslouchal a dělal, co po něm požadujeme.

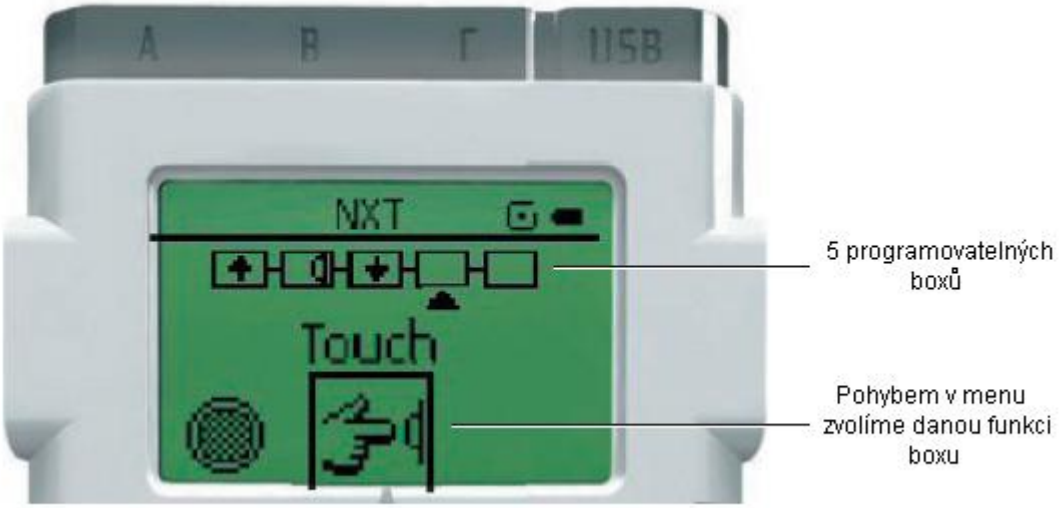
Trochu zabrouzdáme i do způsobu a technologie komunikace s robotem.

2.1 Vývojové prostředí

Protože společnost Lego uvolnila zdrojový kód, vytvořilo se spoustu OpenSource projektů a komunit, které se zabývají vývojem robotiky, a nebo alespoň možnostmi programování jednotek NXT.

2.1.1 NXT Program

Jedná se o jednoduché vytvoření programu přímo pomocí menu originálního firmwaru. Programování je založeno na definování akce 5 boxů.

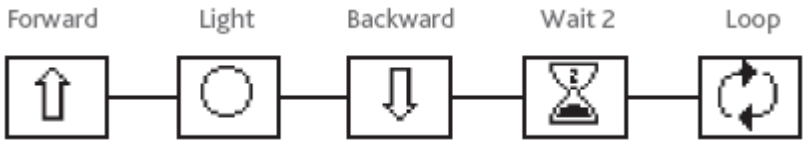


Obrázek 36: Definování programovatelných boxů

Funkce boxů jsou rozděleny na výstupní, vstupní, výstupní, vstupní a následující akci. Ve výstupní akci definujeme činnost motorů, anebo vestavěného reproduktoru. Vstupní akce pak čeká na návratovou hodnotu některého ze senzorů. Tyto dva úkony můžeme zopakovat ještě jednou a pak nastavujeme, zda se má celý proces opakovat, a nebo se má ukončit program.

Výstupy	Vstupy	Následující operace
Vpřed	Pokud je tmavý	Opakuj proces
Vpřed o 5 otáček	Pokud je světlý	Ukonči program
Doprava	Do stisku senzoru	
Doprava o 90°	Čekej 2 sekundy	
Doleva	Čekej 5 sekund	
Doleva o 90°	Čekej 10 sekund	
Vzad		
Vzad o 5 otáček		
Přehraj tón 1		

Tabulka 2: Příkazy pro programovatelné boxy

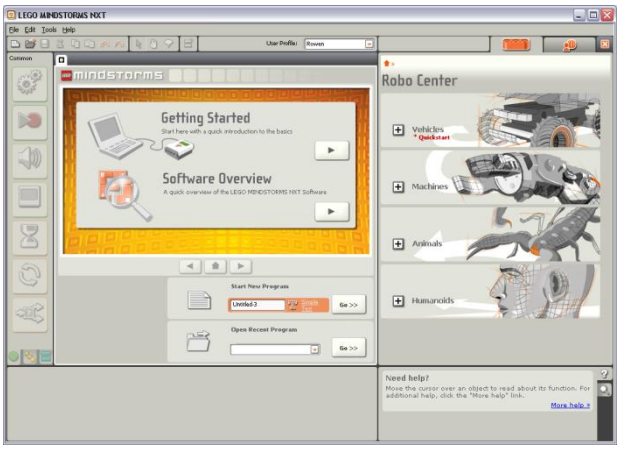


Obrázek 37: Příklad vytvořeného programu

Na Obrázek 37 můžeme vidět příklad programu, který vykonává následující funkci: Robot jede vpřed, dokud senzor zaznamenává světlou podložku. Pak se zastaví a začne couvat po dobu 2 sekund. Poslední příkaz spustí celý proces znovu.

2.1.2 Lego mindstorms NXT: NXT-G

Jedná se o programovací aplikaci dodávanou spolu se stavebnicí. Vývoj této aplikace zajišťuje LEGO Group. Někdy se taky nazývá NXT-G, podle použitého jazyka „G“.



Obrázek 38: Okno aplikace LEGO MINDSTORMS NXT

Aplikace k tvorbě programu využívá grafických ikon, bloků, které mají svůj funkční význam. Je vhodná spíše pro začátečníky, kteří se teprve seznamují s možnostmi a principy programování. Aplikace je postavena na profesionálním vývojovém nástroji LabVIEW™ (od společnosti National Instruments), který je využíván pro vývoj elektroniky, jako jsou mobilní telefony, iPody a podobné zařízení.

Systémové požadavky:

Windows - Procesor Intel Pentium nebo jiný kompatibilní, min 800 Mhz

- Windows XP, Home Edition obsahující Servis Pack 2 nebo Vista
- Minimálně 256 MB RAM
- 300 MB volného místa na HDD
- XGA displej s rozlišením 1024x768
- 1 USB port
- CD-ROM
- Kompatibilní Bluetooth adaptér (volitelný)

Macintosh - PowerPC G3, G4, G5, minimálně 600 MHz

- Apple Mac05 X v.10.3.9 nebo 10.4
- Minimálně 256MB RAM
- 300 MB volného místa na HDD
- XGA displej s rozlišením 1024x768
- 1 USB port
- CD-ROM
- Kompatibilní Bluetooth adaptér (volitelný)

2.1.2.1 Popis aplikace

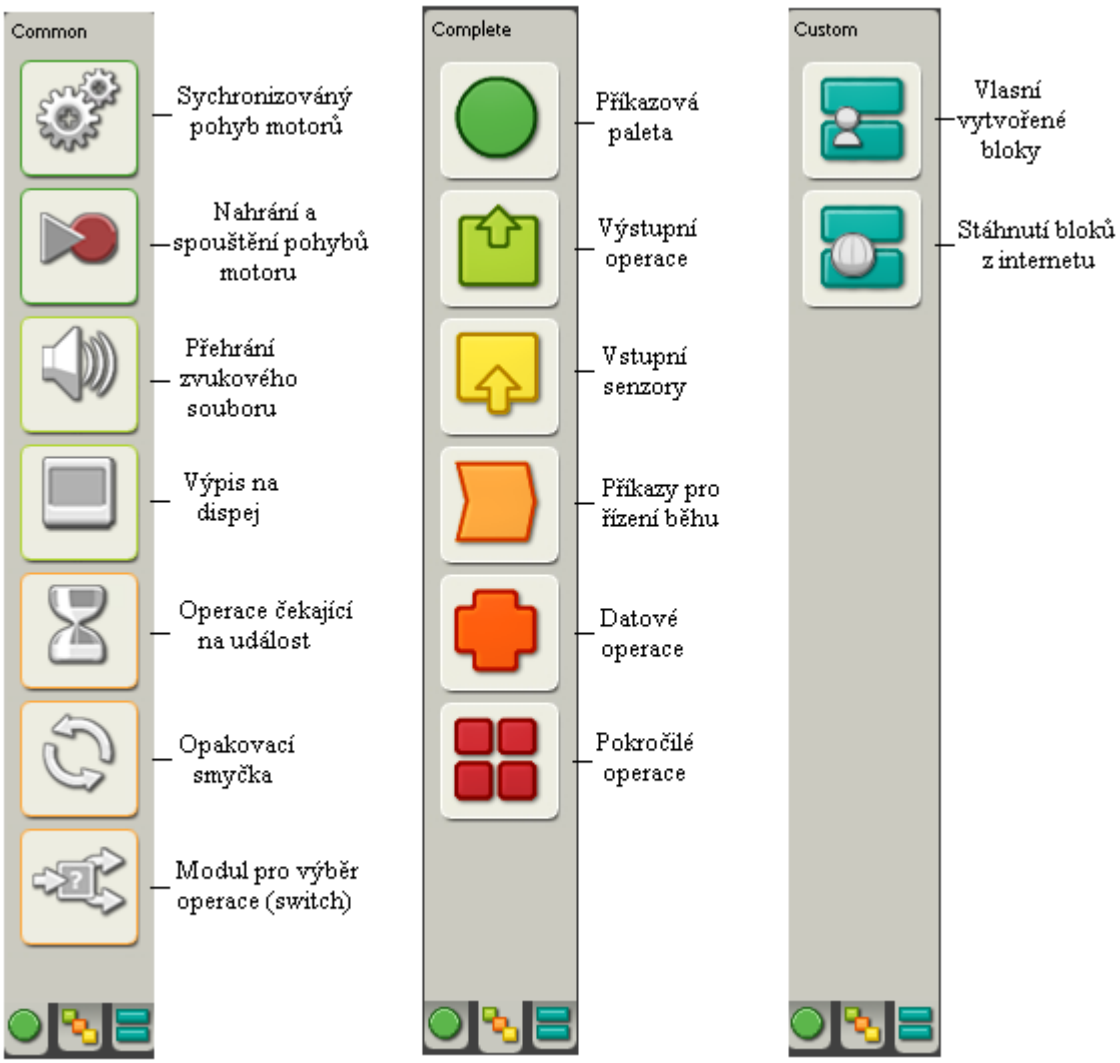
Jako téměř každá aplikace, i LEGO MINDSTORMS NXT používá k usnadnění práce panel nástrojů. V tomto případě je panel velmi jednoduchý.

Obrázek 39: Panel nástrojů LEGO Software

Nabízí základní operace pro práci s projektem, jako je jeho vytvoření, otevření již existujícího projektu a uložení. Další nástroje slouží k práci se schránkou, jako je vyjímání, kopírování a vkládání. Poslední čtyři nástroje už slouží k vlastní práci v projektu. Jedná se

o výběrový nástroj, nástroj posuvu plátna, komentář a nástroj pro vytváření vlastních funkčních bloků. Jako poslední, co můžeme na panelu vidět, je výběr uživatelského profilu, který v sobě uchovává názvy projektů, se kterými daný uživatel pracuje či pracoval.

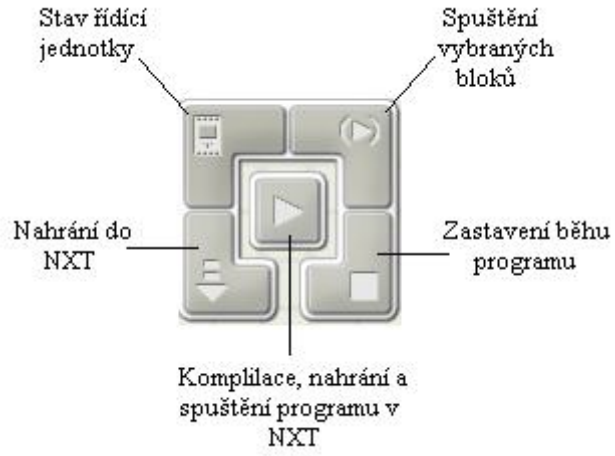
Podél levého okraje okna aplikace se nachází nástrojová paleta, která je rozdělena do tří záložek.



Obrázek 40: Palety nástrojů LEGO softwaru

První záložka je příkazová a obsahuje pouze základní příkazové operace. Druhá záložka poskytuje kompletní nabídku všech potřebných bloků k programování. Základním prvkem úspěšného programování jsou příkazy pro řízení běhu programu, které nám nahrazují cykly „for“, „while“, „until“, podmínku „if“ a výběr „switch“ známé z jiných programovacích jazyků.

Pro kompilaci a spouštění vytvořených programů slouží nenápadný ovládací prvek v pravém dolním rohu. Pomocí něj můžeme nahlédnout na stav řídící jednotky robota a nahrávat do jednotky a spouštět vytvořené programy.



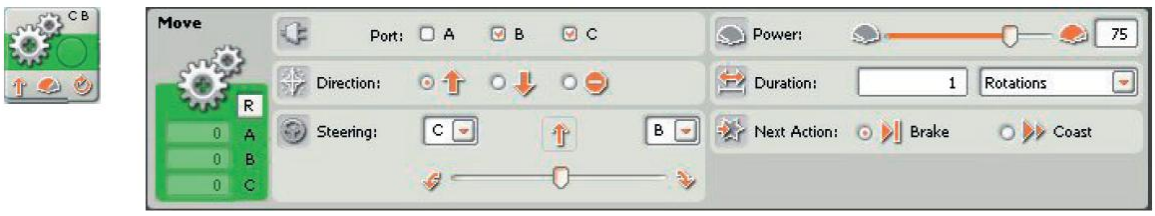
Obrázek 41: Ovládací prvek programu

2.1.2.2 Základní funkční bloky

Základem vytvoření nějakého programu je nutné se seznámit s funkčními bloky, které aplikace nabízí svému uživateli.

Pohyb motorů:

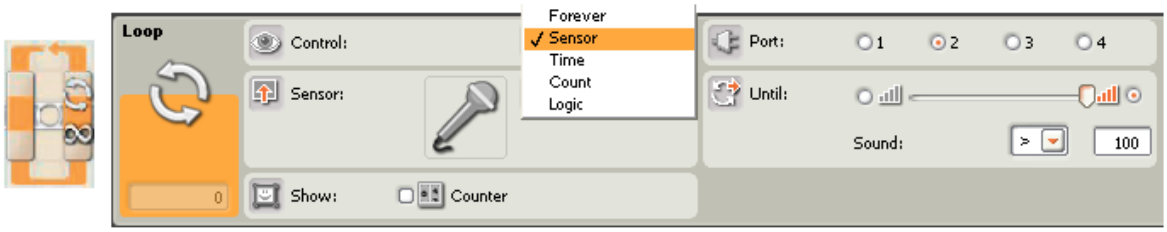
Tento druh operace ovládá blok s příznačnou ikonou převodu. Najdeme jej v příkazové paletě. Při označení bloku se nám ve spodní části okna otevře nabídka nastavení funkce bloku v programu. Umožní nám výběr patřičného portu nebo skupiny portů, které bude blok ovládat. Dále zde nastavujeme směr pohybu a pokud pracuje se dvěma porty, můžeme pomocí posuvníku ovládát zatačení robota. Mezi důležitější nastavení patří síla motoru a druh operace, kterou bude motor vykonávat. Můžeme zde nastavit otočení o počet otáček nebo stupňů, případně se může motor otáčet neomezeně nebo v závislosti na čase.
V případě omezeného otáčení lze nastavit následující akci a to buď zabrždění motoru nebo volnoběžné otáčky.



Obrázek 42: Funkční blok ovládání motoru

Opakovací smyčka:

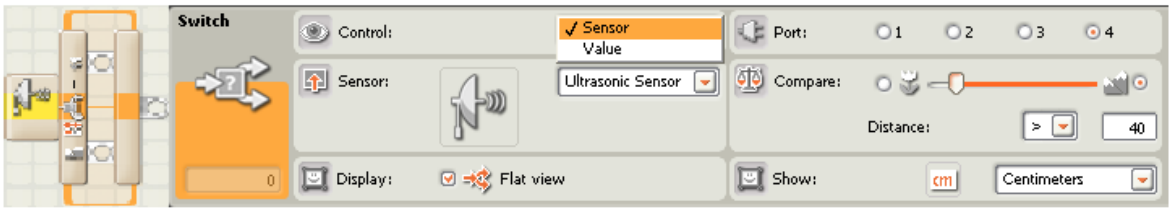
Nedílnou součástí každého programu je nějaký druh smyčky. Tato aplikace nabízí sice jen jednu smyčku, ale za to širokou škálu možností jejího využívání. Tato smyčka může pracovat do nekonečna nebo se může řídit událostí na senzoru, vykonáním určitého počtu cyklů, vypršením času nebo logickou hodnotou.



Obrázek 43: Funkční blok opakovací smyčky

Podmínka:

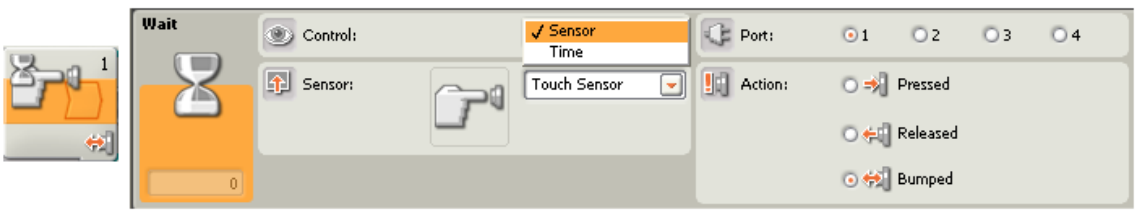
Pokud se při programování neobejdeme bez smyčky, tak bez rozhodovací podmínky už vůbec ne. Zde nám vývojáři nabídli tzv. „switch block“, který po pravdivém vyhodnocení podmínky provede nějakou část programu a při nepravdivé podmínce zase jinou část programu. Podmínka může být realizována hodnotou, a nebo senzorem.



Obrázek 44: Funkční blok podmínka

Čekání na událost:

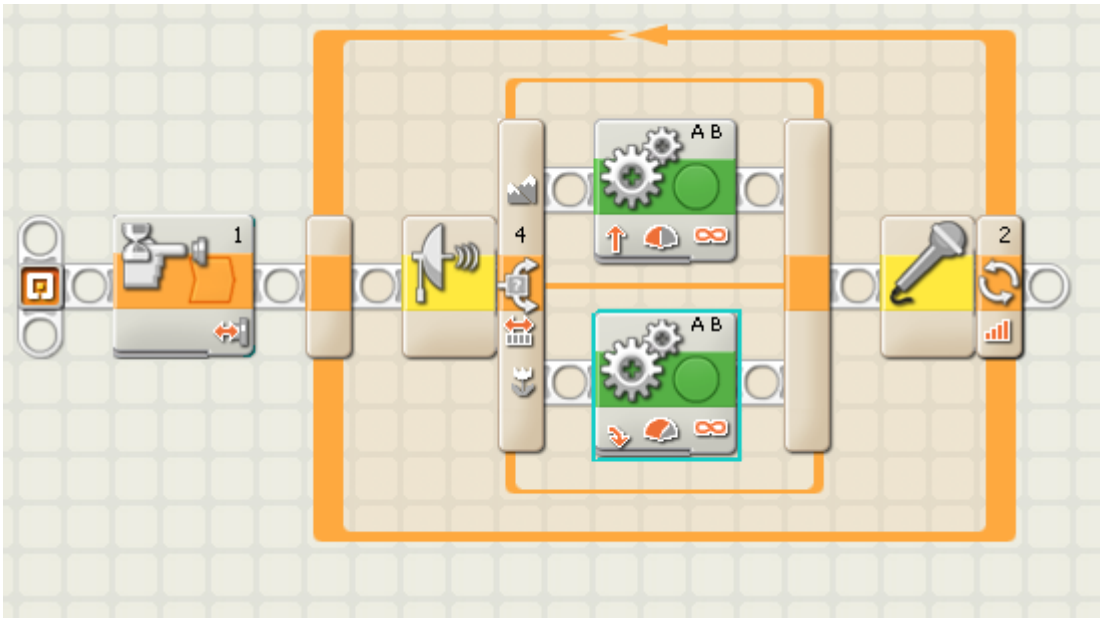
Další mocný nástroj pro programování robota NXT. Dokáže zastavit chod programu dokud nenastane nějaká událost na čidlech nebo dokud nevyprší časový limit. Opět nabízí sérii nastavení pro každý senzor.



Obrázek 45: Funkční blog čekání na událost

2.1.2.3 První program pro robota NXT

Když už jsme seznámení s uživatelským rozhraním programovací aplikace pro NXT robota od skupiny vývojářů LEGO Group, můžeme se pustit do vytváření prvního funkčního programu. K tomu nám postačí 5 bloků. Náhled programu můžeme vidět na Obrázek 46.



Obrázek 46: Jednoduchý program pro NXT

První blok způsobí čekání na událost z tlakového senzoru. Dokut nestiskneme tlakový senzor, nic se neděje a robot stojí. Po stisku senzoru se program dostane do smyčky, ve které se kontrolují údaje z ultrazvukového senzoru a pokud je vzdálenost od překážky větší, než nastavená hodnota (40 cm) uvedou se do pohybu motory připojené na portu A a B a robot směřuje vpřed. Pokud je ale překážka blíž, začne se robot otáčet takovým způsobem, že motor na portu A jede vpřed a motor na portu B couvá. Až se otočí do vzdálenosti větší 40 cm od překážky, bude opět pokračovat v jízdě vpřed, dokud zvukové čidlo nezaznamená větší hladinu než 90dB. Takže pokud zatleskáme nebo pískneme na robota, ukončí se program a robot se zastaví.

2.1.3 Bricx Command Center

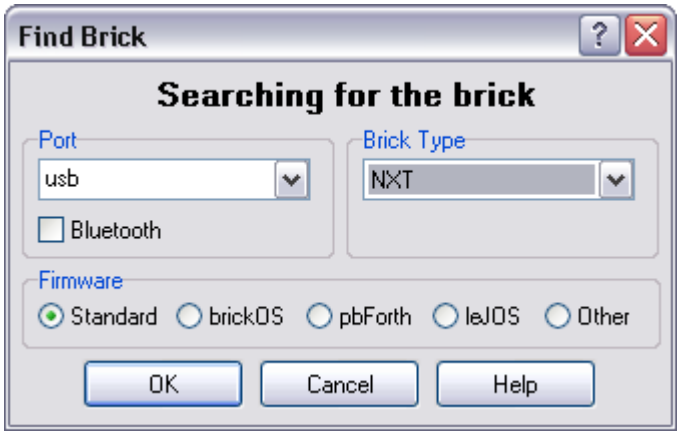
NXT robota lze krom dodávaného grafického programu LEGO Mindstorms NXT programovat i pomoci vyššího programovacího jazyka, který je podobný C.

2.1.3.1 Popis programu BricxCC

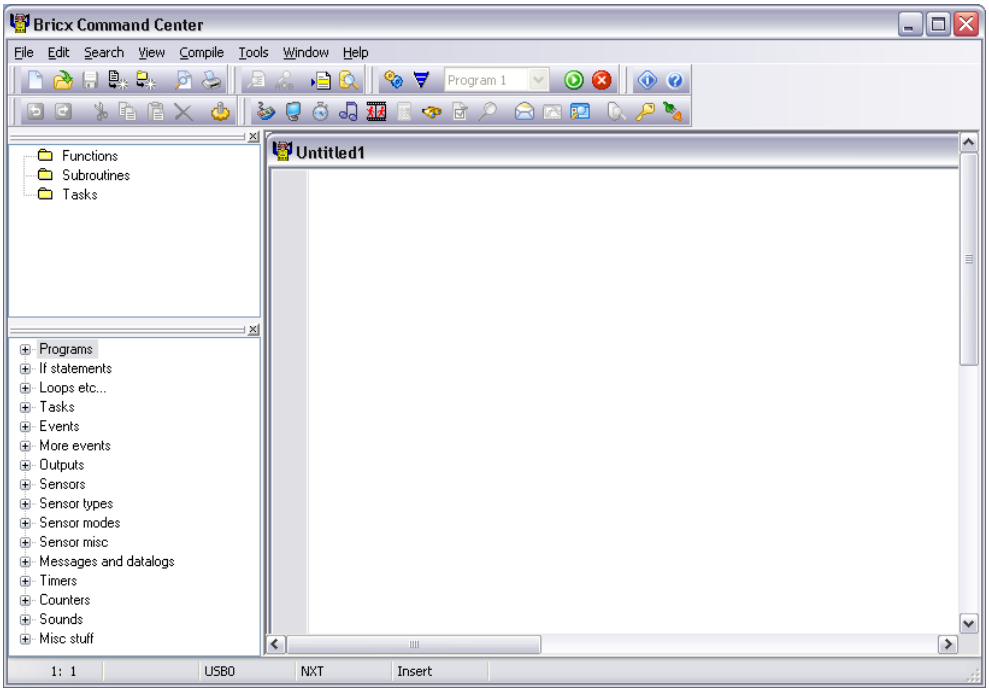
Bricx Command Center (BricxCC) je plnohodnotný program, který zvládne naprogramovat LEGO Mindstorms NXT použitím jazyka Not eXactly C (NXC), dále pak pomocí Byte Codes (NBC) – assemblerovsky orientovaný programovací nástroj a i jednoduchým

programovacím jazykem NPG.

Jako první při spuštění programu na uživatele vybafne nastavení pro připojení k řídící jednotce. Nabízí možnost připojení přes USB kabel, COM port nebo Bluetooth. Pokud se nechceme připojit, stačí formulář stornovat.

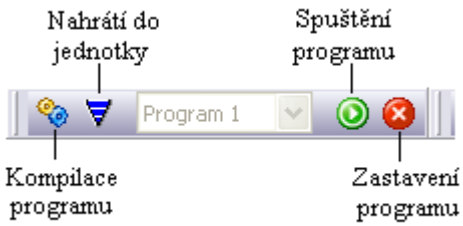


Obrázek 47: Formulář pro připojení BricxCC k jednotce NXT



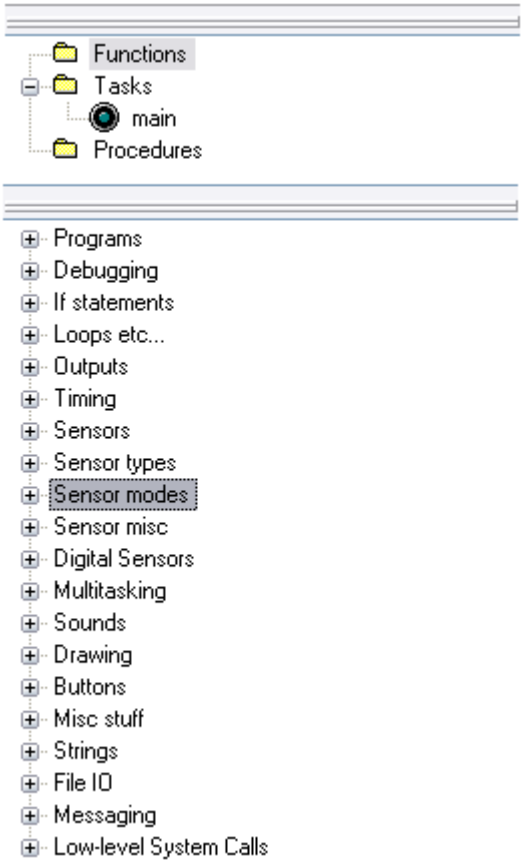
Obrázek 48: Okno aplikace BricxCC

Vizáž aplikace je podobná jiným známým vývojovým prostředí. Nabízí širou nástrojovou paletu. Nejpoužívanější jsou nástroje pro kompilaci a spuštění programu.



Obrázek 49: Nástroje pro kompilaci programu v BricxCC

V levé části okna aplikace můžeme vidět prohlížeč projektem a prohlížeč funkcí.



Obrázek 50: Prohlížeč projektu a funkcí

2.1.3.2 Syntaxe programu

Ovládání motorů:

Spuštění a zastavení motorů:

OnFwd(OUT_A, 50); // Otáčí motorem A na 50% po směru hodinových ručiček

OnFwd(OUT_B, 50); // Otáčí motorem B na 50% po směru hodinových ručiček
OnFwd(OUT_AB, 100); // Otáčí motory B a C naplno po směru hodinových ručiček
OnRev(OUT_C, 30); // Zpětně otáčí motorem C na 60 lze také použít OnFwd(OUT_C, -30)
Off(OUT_ABC); //zastaví všechny motory

Pokud chceme použít synchronizované otáčení motorů, použijeme následující příkaz:

OnFwdReg(OUT_AB, 50, 1); // Otáčí synchronizovaně motory A a B na 50% rychlosti

Pokud potřebujeme použít rotaci o nějaký stupeň otočení, pak máme následující příkaz:

RotateMotor(OUT_B, 25, 60); // Otočí motorem B 25% rychlostí o 60°)

Tlakový senzor:

SetSensorTouch(IN_1); // nastaví port 1 jako tlakové čidlo
tast = Sensor(IN_1); // čte hodnotu senzoru - 0 rozepnuto, 1 sepnuto

Zvukový senzor:

SetSensorSound(IN_2); //nastaví na port 2 zvukový senzor
sound = Sensor(IN_2); // čte hodnotu hlasitosti na portu 2, vrací 0 - 100

Světelný senzor:

SetSensorLight(IN_3); // nastaví na port 3 světelný senzor
light = Sensor(IN_3); // čte intenzitu světla na portu 3, vrací 0 - 100

Ultrazvukový snímač:

SetSensorLowspeed(IN_4); // nastaví na portu 4 ultrazvukový senzor
ultra = SensorUS(IN_4); // čte hodnotu sonaru na portu 4, vrací vzdálenost v palcích

Displej:

Rozlišení displeje je 100*64. Umístnění textu je v souřadnicích x (zleva) a y (zespodu). Pro vertikální souřadnici lze rovněž LCD_LINE1, LCD_LINE2, ... Kromě toho existuje příkaz "true" a nebo "false". True znamená, že vždy je celý výstup smazán a nově zobrazen. False znamená, že budou změněna pouze aktualizovaná místa.

TextOut(0,50,"Ahoj"); // Zobrazí na daných souřadnicích text Ahoj

TextOut(30,LCD_LINE4,"Ahoj"); // Zobrazí na souř. x=30 na 4.řádku text Ahoj

Řízení běhu programu:

while(„podminka“) {} //Dělej dokud je splněna podmínka
do {} while(“podminka”);
repeat(“pocet opakovani”) {} //Opakuje akci daný počet krát

Wait("hodnota") ; //Uspí program na danou dobu – 1000 = 1 sekunda

until("podminka"); // Čeká, dokud není splněná podmínka

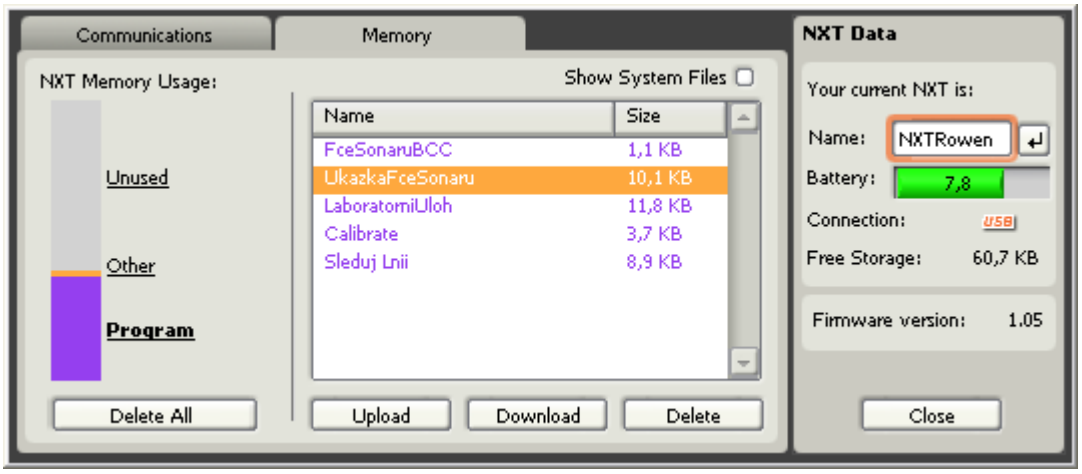
if ("podminka") {"neco"} else {"jineho"} //Pokud je splněná podmínka, udělej „neco", jinak udělej „jineho"

2.1.3.3 Jednoduchý program pomocí BricxCC

```
#define hlasitost 100
#define tlacitko SENSOR_1
#define zvuk SENSOR_2
#define sonar SENSOR_4

task main()
{
SetSensorTouch(IN_1);
until(tlacitko == 1);

do
{
 SetSensorLowspeed(IN_4);

 if ((SensorUS(IN_4))>40)
 {
  OnFwd(OUT_AB, 50);
 }
 else
 {
   OnFwd(OUT_A, 50);
   OnRev(OUT_B, 50);
  }

  SetSensorSound(IN_2);
}
while (zvuk < hlasitost);
}
```

Jedná se o funkčně i principielně stejný program jako v kapitole 2.1.2.3. Program inicializuje tlakový senzor na portu 1 a pak čeká na stisknutí. Po stisku se spustí cyklus „do,while", ve kterém se pomocí podmínky „if" ptáme na vzdálenost od překážky. Pokud je překážka vzdálenější než 40cm, tak spustí motory na portu A a B. Pokud podmínka není splněna, začne se robot otáčet. Ukončení programu je stejně jako v předchozím případě provedeno naslouchání na zvukovém senzoru. Pokud je hlasitost větší nastavená mez, podmínka ve funkci „while" není splněna a program se ukončí.

Z Obrázek 51 je patrná výhoda programování v tomto vývojovém prosředí. Program vytvořený pomocí LEGO softwaru má velikost 10,1 KB a program vytvořený pomocí BricxCC má velikost pouhých 1,1 KB, přičemž funkční hodnota programu nebyla nikterak ovlivněna.



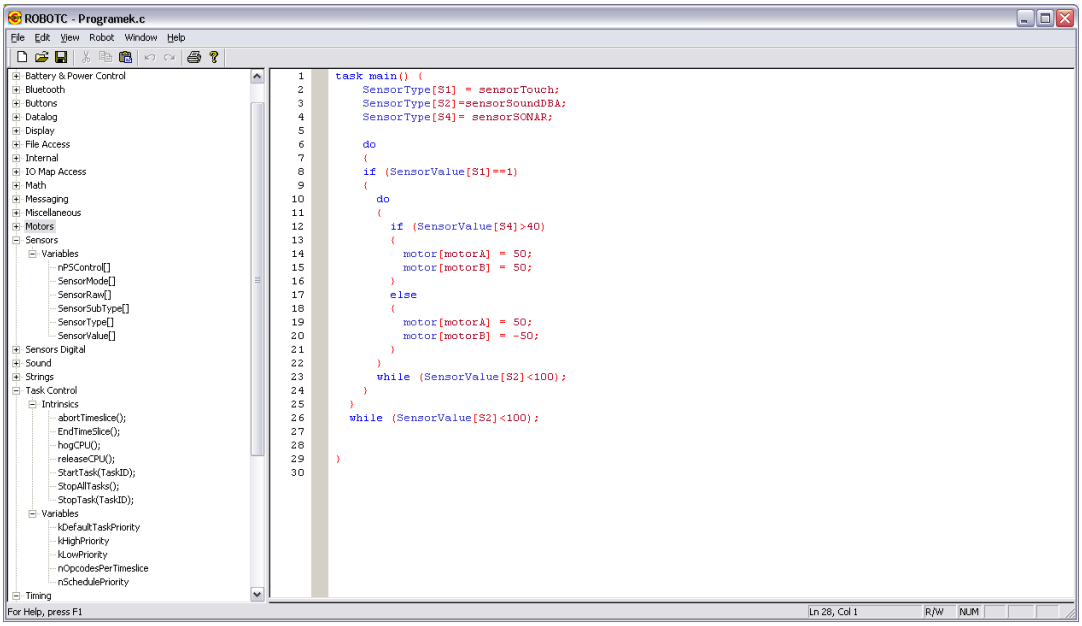
Obrázek 51: Srovnání velikosti téhož programu vytvořeného v NXT-G

2.1.4 RobotC

2.1.4.1 Popis aplikace

Jedná se o placené vývojové prostředí, které ke spouštění programu potřebuje i speciální firmware dodávaný spolu s aplikací. RobotC je velmi jednoduchá aplikace, která spojuje typické znaky programování v jazyce C s funkcemi pro činnost robota. K dispozici jsem měl jen 30 denní trial verzi s určitým omezením některých funkcí.

Obrázek 52: Dialogové okno About RobotC



Obrázek 53: Okno aplikace RobotC

2.1.4.2 Syntaxe příkazů

Programování v této aplikaci je téměř shodné s programováním v BricxCC. Rozdíl je jen v názvech a způsobu práce s příkazy pro robota.

Senzory:

SensorType[S1] = sensorTouch; // Nastavení portu pro daný senzor

SensorValue[S1] //Vrátí hodnotu senzoru na daném portu

Motory:

motor[motorA] = 50; // Motor na portu A pojede vpřed na 50%

motor[motorB] = -50; // Motor na portu B pojede vzad na 50%

2.1.4.3 Jednoduchý program v RobotC

```
task main() {
    SensorType[S1] = sensorTouch;
    SensorType[S2]=sensorSoundDBA;
    SensorType[S4]= sensorSONAR;

    do
    {
    if (SensorValue[S1]==1)
    {
      do
      {
        if (SensorValue[S4]>40)
        {
          motor[motorA] = 50;
```

```
            motor[motorB] = 50;
          }
          else
          {
            motor[motorA] = 50;
            motor[motorB] = -50;
          }
        }
        while (SensorValue[S2]<100);
      }
    }
    while (SensorValue[S2]<100);
  }
```

Program má stejnou funkční hodnotu jako v předchozích dvou kapitolách věnovaným jednoduchému programu (2.1.2.3, 2.1.3.3). Robot po stisknutí tlakového čidla pojede vpřed a po zjištění překážky ve vzdálenosti 40 cm od robota provede otočení. Na hlasitý povel se program zastaví.

Program má v paměti řídící jednotky velikost 0,3 KB, tedy ještě menší než v předchozím případě při použití BricxCC.

2.1.5 LeJOS

Jedná se o multiplatformní projekt (Windows, Mac, Linux) složený z knihovny a firmwaru pro řídící jednotku. Z originálním FW je knihovna nefunkční. Cílem projektu bylo vytvořit vývojové prostředí s využitím jazyka JAVA. LeJOS sám o sobě není žádné vývojové prostředí a tudíž je nutné využít některé ze standardních aplikací pro vývoj JAVA aplikací jako je Eclipse (pro něj existují přednastavené paginy pro jazyk JAVA) nebo i již zmíněný BricxCC. Pro tyto zmíněné existují přednastavené pluginy, pro snadnou tvorbu projektů.

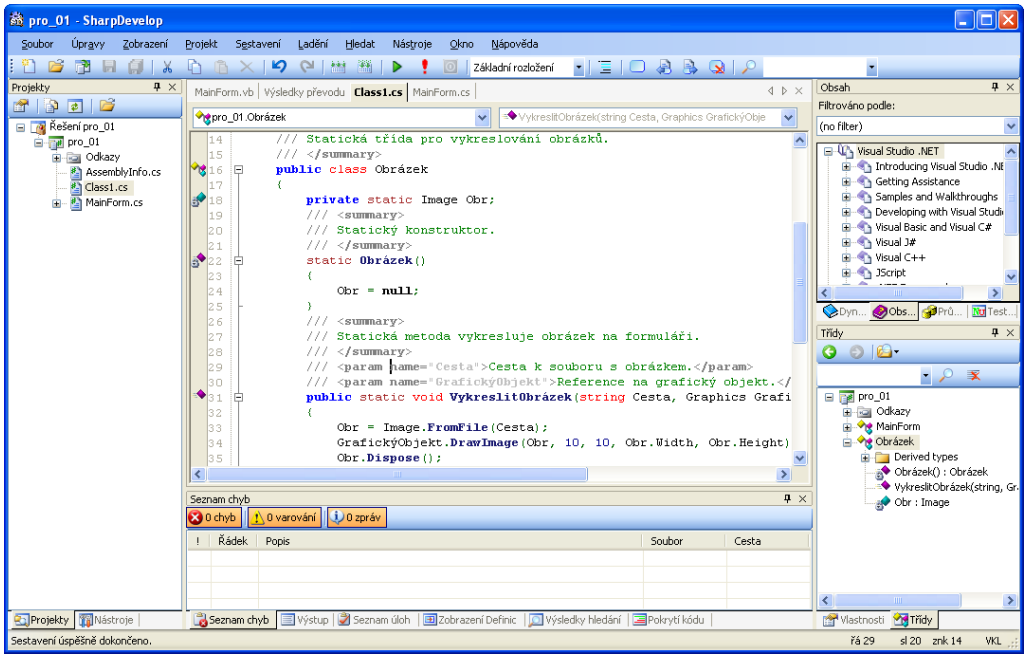
2.1.6 NXT#

Samo o sobě to není vývojové prostředí jako takové. Jedná se o knihovny pro vývojové prostředí určené pro jazyk C#. Výsledkem je Windows aplikace, která pomocí bluetooth ovládá robota.

2.1.6.1 SharpDevelop 2.2

Jedná se o otevřený projekt zkušených programátorů, jehož vývoj započal v roce 2000. Záměrem autorů bylo vytvořit otevřené a nezávislé integrované vývojové prostředí pro jazyk C#. SharpDevelop 2.0 pracuje s jazykovou specifikací C# 2.0, takže z tohoto hlediska vývojáři určitě strádat nebudou. Jako jediný produkt k nám SharpDevelop dovede

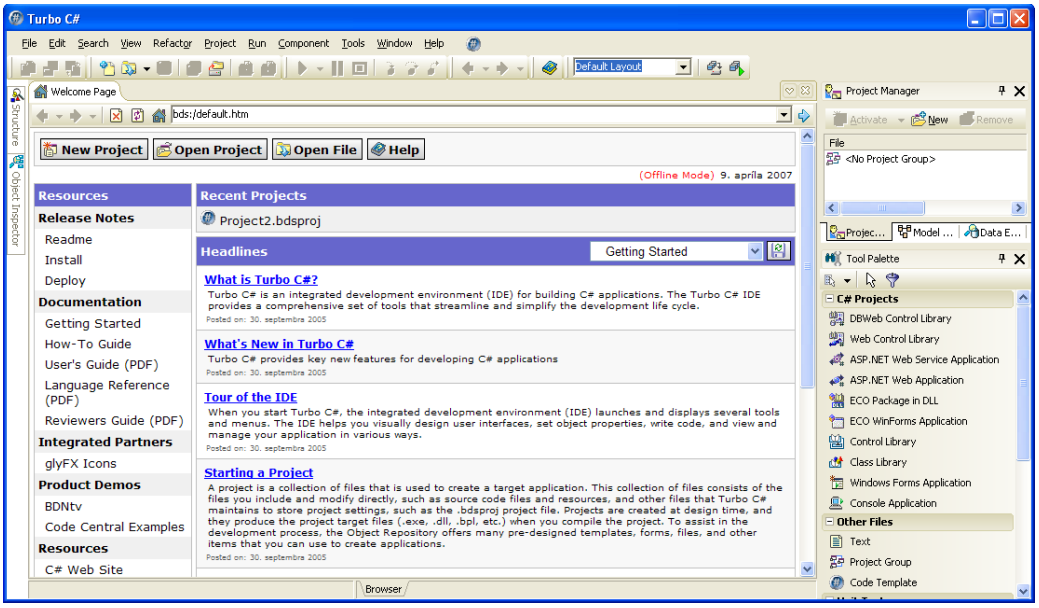
promlouvat v mateřské řeči – čeština je jedním z dvaceti jazykových balíčků, s nimiž program kooperuje. V praxi to znamená, že nabídky, dialogová okna a zprávy vývojového prostředí jsou před vás předkládány v přirozeně srozumitelné podobě.



Obrázek 54: Okno aplikace SharpDevelop 2.2

Jádrem vývojového prostředí je vizuální návrhář s kolekcí ovládacích prvků, správce projektů, okno s vlastnostmi a samozřejmě editor zdrojového kódu. [16]

2.1.6.2 Borland Turbo C# 2006 Explorer

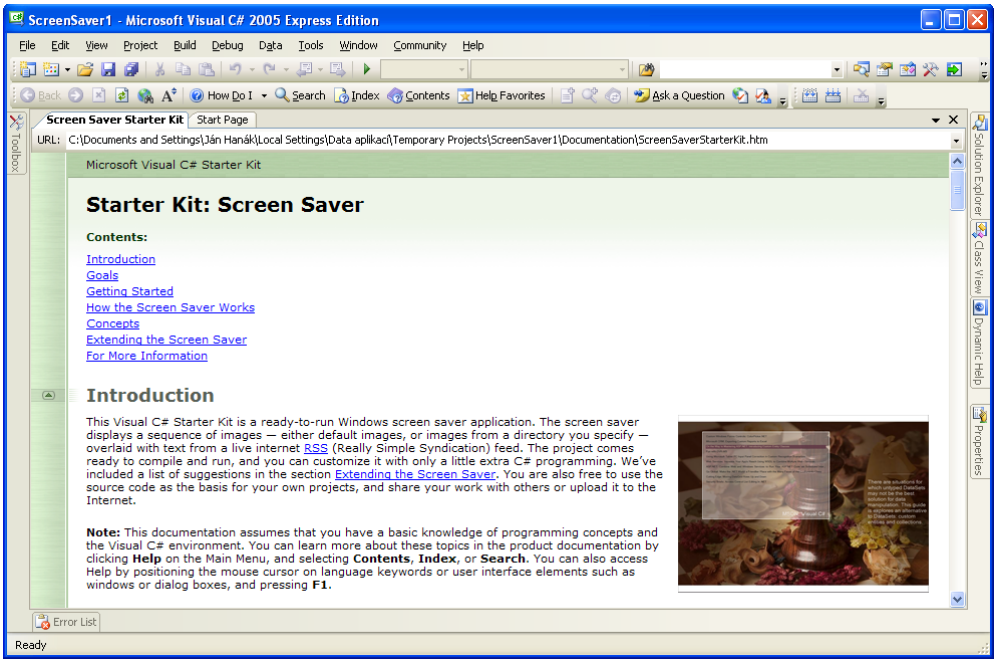


Obrázek 55: Okno aplikace Borland Turbo C# 2006

Je přímým konkurentem expresního Visual C# 2005 od Microsoftu. Vedle „povinných“ aplikací (formulářová aplikace pro Windows, konzolová aplikace a knihovna tříd) jsou zastoupeny rovněž webové aplikace a služby (ASP.NET 1.1) či knihovny standardních a webových ovládacích prvků. Novinkou je dvojice šablon využívajících služeb ECO (Enterprise Core Objects) prostředí, které zavádí účinné mechanizmy pro psaní objektově orientovaných aplikací pracujících s relačními databázovými zdroji. [16]

2.1.6.3 Microsoft Visual C# 2005 Express

Jedná se o odlehčenou verzi „velkého“ Visual C# 2005. Přichází s nejnovější implementací jazyka C#. Etapa vizuálního programování je svižná a intuitivní, na čemž má největší zásluhu skvěle vyvedený návrhář s bohatou porcí ovládacích prvků a komponent.



Obrázek 56: Okno aplikace Microsoft Visual C# Express

Editoru zdrojového kódu vévodí proslulá technologie IntelliSense, jejíž reinkarnace ve Visual C# 2005 Express je citlivější a přesvědčivější než kdykoliv předtím. Vývojář se tudíž nemusí nijak zvlášť namáhat, neboť jeho kroky jsou pod bedlivým dohledem IntelliSense. V praxi to znamená, že se před vámi na každém kroku rozprostírají nabídky s datovými členy, vlastnostmi a metodami, které je možné v daném okamžiku použít. Instrukce a příkazy zdrojového kódu jsou poznačeny barevnými filtry, které zvyšují jejich čitelnost. [16]

2.1.6.4 Syntaxe příkazů

Pomocí funkce „Add Reference“ z hlavního menu vložíme do aplikace knihovny, které obsahují moduly pro práci s řídící jednotkou NXT. Jde o soubory Bram.Lego.dll, Bram.Utilities.dll, Bram.NxtSharp.dll. Pomocí nabídky pro nástrojovou lištu si vytvoříme nové složky pro importování modulů knihoven. Následně už jen pracujeme s moduly, vytváříme události a obslužné rutiny pro dané události. [16]

2.2 Možnosti komunikace

2.2.1 Komunikace pomocí kabelu

První způsob komunikace je pomocí klasického USB kabelu typu A-B. Tento způsob umožňuje komunikaci rychlostí až 12Mbit/s. Využití kabelu je vhodné především pro statické modely robotů nebo pro nahrávání programů či přehrávání firmwaru.

Obrázek 57: USB kabel pro připojení NXT

2.2.2 Komunikace pomoci bluetooth

Druhým způsobem komunikace s jednotkou je pomocí vestavěné technologie bluetooth. Tahle možnost se dá dobře uplatnit při sestavování programu pro vzdálené ovládání robota nebo při řízení pohybujícího se robota programem spuštěným v PC. V těchto případech je nutné poznamenat, že při komunikaci pomocí bluetooth dochází k lagování. Doba reakce robota na příkaz je závislá na vzdálenosti jednotky od vysílacího zařízení, minimálně však cca 30 ms.

Aby bylo možné navázat s řídící jednotkou kontakt, je nutné použití bluetooth ovladačů třetích stran (např.: IVT Corporation BlueSoleil). Ovladače, které poskytuje firma Microsoft v operačním systému Windows XP nejsou schopny navázat kontakt s jednotkou. Dalším rozhodujícím faktorem pro úspěšnou komunikaci je výběr vhodného BT zařízení.

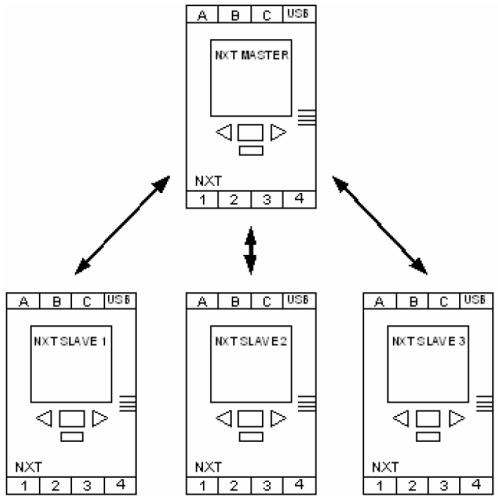
Obrázek 58: Vhodný USB dongle

Druh zařízení	**Kompatibilita**
Abe UB22S	
Belkin F8T003 ver. 2 (short range)	
BlueFRITZ! AVM BT adapter, BlueFRITZ! USB v2.0	
Cables Unlimited USB-1520	
Dell TrueMobile Bluetooth Module	
Dell Wireless 350 Bluetooth Internal Card	
Dlink DBT-120	
MSI Btoes	
MSI StartKey 3X-faster	
TDK GoBlue	
Qtrek, Bluetooth USB Adapter v2.0	

- kompatibilní zařízení - nekompatibilní zařízení

Tabulka 3: Seznam kompatibibilních USB zařízení

Jednotky je možné párovat nejen s počítačem, ale i mezi sebou. Naskytuje se tak možnost komunikovat až se 3 jednotkami současně. Samozřejmě, že absolutně současná komunikace není možná, takže se při komunikaci využívá časového multiplexu.

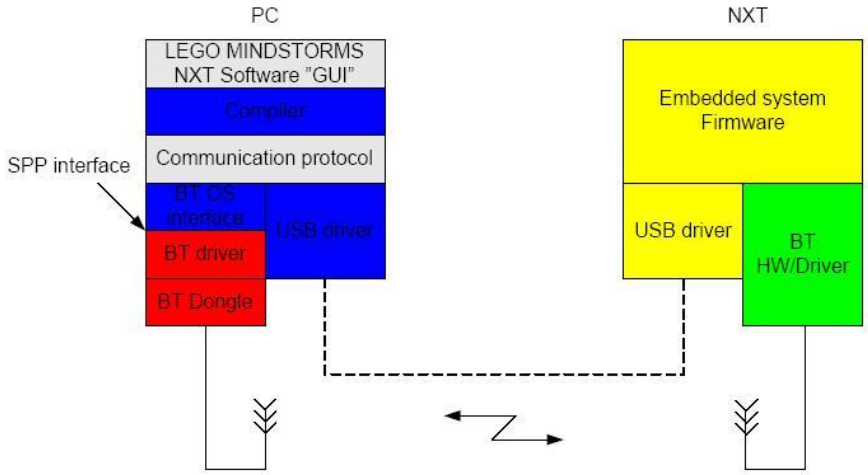


Obrázek 59: Komunikace mezi jednotkami

LEGO MINDSTORM využívá bezdrátové komunikace formou zasílání a přijímání Bluetooth paketů striktně dané struktury, použitím profilu sériového portu. Komunikace s NXT jednotkou je možná buďto pomocí LEGO MINDSTORM komunikačního protokolu (zasíláním a čtením zpráv v mailboxech), nebo pomocí syrových dat ve formě Bluetooth paketů.

2.2.2.1 *Komunikační vrstvy*

Na obrázku vidíme základní komunikační vrstvy softwaru v PC a Bluetooth zařízení v jednotce.



Obrázek 60: Komunikační vrstvy PC a NXT

2.2.2.2 *Struktura paketů*

Length, LSB	Length, MSB	Command Type	Command	Byte 5	Byte 6	Etc.

Tabulka 4: Struktura Bluetooth paketu

První dva bajty (Byte 0, Byte 1) struktury jsou informativní, určují délku posílané zprávy a pro pochopení obecného principu je není třeba dále rozebírat.

Byte 3 – Typ příkazu (Command type)

Určuje typ posílaného příkazu a vyskytuje se v těchto podobách hexadecimálního formátu:

- 0x00 – Přímý příkaz, vyžaduje odpověď
- 0x01 – Systémový příkaz, vyžaduje odpověď
- 0x02 – Odpověď
- 0x80 – Přímý příkaz, nevyžaduje odpověď
- 0x81 – Systémový příkaz, nevyžaduje odpověď

Byte 4 – Příkazový bajt (Command byte)

Určuje co by se mělo vykonat s následujícími daty (otevřít, číst, zapsat, smazat …).

Byte 5-N – Dodatečné informace

Například jméno souboru, určení portu, určení výkonu motoru apod.

Příklad:

Pokud bychom chtěli nastavit jméno řídící jednotky, poslali by jsme po otevřeném komunikačním kanále příkaz složený z následujících bytů:

- Byte 0: 0x01
- Byte: 0x98
- Byte 2-17: Nové jméno, maximálně 15 znaků

Následně obdržíme odpověď:

- Byte 0: 0x02
- Byte 1: 0x98
- Byte Status: 0 znamená úspěch, větší jak 0 neúspěch

II. PRAKTICKÁ ČÁST

3 OVLÁDÁNÍ ROBOTA NXT POMOCÍ PC

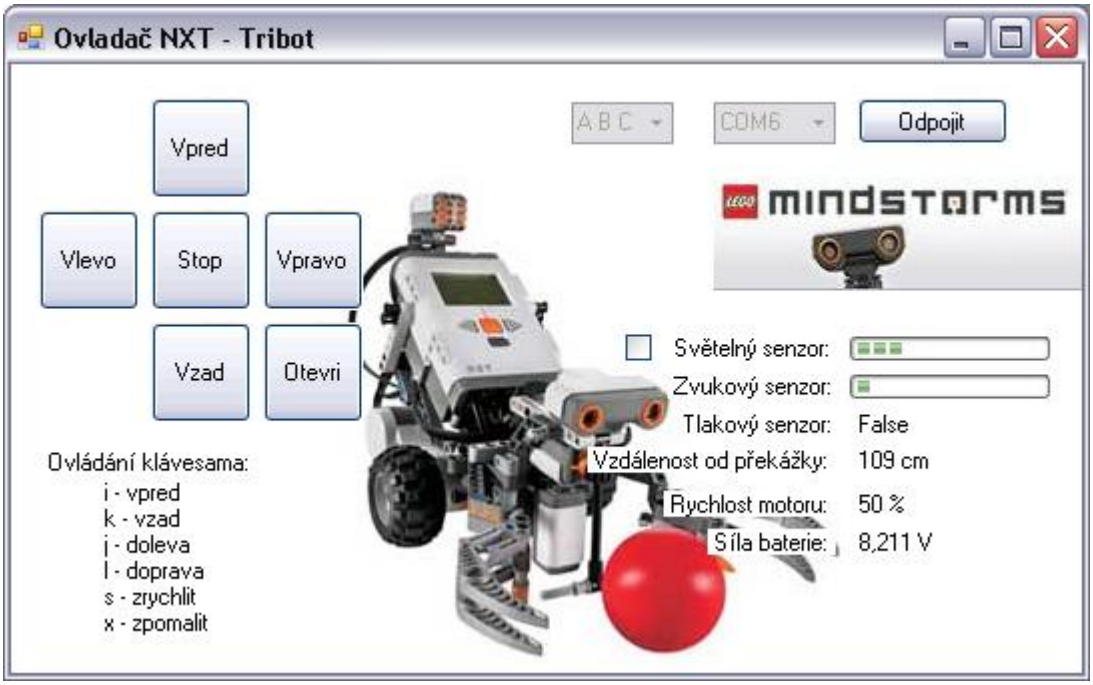
K vytvoření ovládání robota byl použit vyšší programovací jazyk C#. Komunikace a robotem je realizována pomocí technologie bluetooth.

3.1 Popis programu

Po spuštění aplikace uvidíme před sebou jednoduchý ovládací prográmek, kterým můžeme ovládat všechny funkce robota pomocí bezdrátové komunikace mezi PC a robotem.

Abychom mohli ovládat robota, je nutné nejdříve navázat spojení mezi PC a řídící jednotkou pomocí dodávaných ovladačů a softwaru k Bluetooth dongle. Komunikace se vytvoří na nějakém sériovém portu. Číslo portu zjistíme rovněž v softwaru pro BT.

Následně tento port vybere v nabídce programu, vybereme taky pořadí motorů připojených na dané porty řídící jednotky v pořadí levý, pravý, funkční a stiskneme tlačítko připojit. Pokud připojení proběhlo v pořádku, aktivují se tlačítka a senzory budou ukazovat aktuální hodnoty. Pokud se připojení nezdaří, vyskočí dialog s chybou.



Obrázek 61: Okno programu Ovladač NXT - Tribot

Vlastní ovládaní je možné dvěma způsoby. První způsob je pomocí myši najetím na příslušné tlačítko směru a stisknutím. Tím se robot rozjede a zastaví jej až stisk tlačítka stop. Opětovným stiskem tlačítka směru dojde ke zvětšení rychlosti motorů. Druhý způsob je pomocí kláves, kde klávesa „i“ přestavuje jízdu vpřed, „k“ jízdu vzad, „j“ jízdu doleva a

„l“ jízdu doprava. Puštěním klávesy jízdy dopředu nebo dozadu dojde k zastavení pohybu. Je to tedy stejné ovládání jako známe z počítačových her. Rychlost motorů můžeme ovlivnit klávesami „s“ a „x“.

3.2 Použité vývojové prostředí a syntaxe

K vývoji programu bylo použito Visual Sudio 2008 a v něm obsažený Visual C# Express 2005. Díky IntelliSense je práce s programem velmi přívětivá a programátor se nemusí bát syntaktických chyb v podobě překlepů.

Základem práce s programem pro NXT je nutnost importování knihoven Bram.NxtSharp.dll a Bram.Utilities.dll pomocí menu *Project -> Add Reference… -> browse*. Poté si v nástrojové liště vytvoříme novou záložku, například NXT#, do které pomocí výběrového menu po stisku pravého tlačítka myši *Choose Items…* naimportujeme jednotlivé objekty z knihovny Bram.NxtSharp.dll.

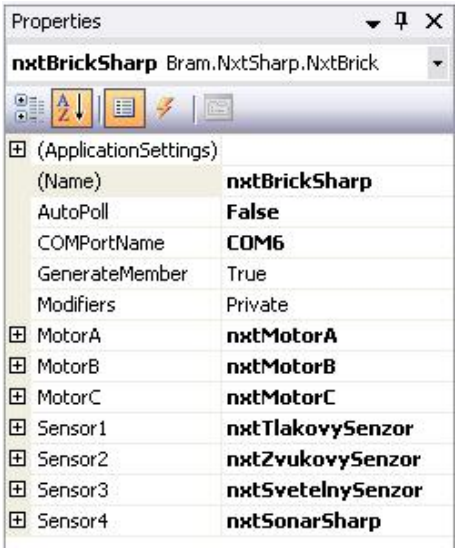
Obrázek 62: Objekty knihovny Bram.NxtSharp.dll

Pomocí těchto objektů provádíme veškeré programování přetažením objektu na plochu návrhu. Následně si objekty patřičně pojmenujeme.

Obrázek 63: Objekty použité z knihovny

Nyní je nutné nastavit objekty tak, aby byly ve vzájemné vazbě. Tedy jednotlivé objekty představující vstupní a výstupní zařízení zapouzdřit k náležité řídící jednotce, v tomto

případě nxtBrickSharp. To provedeme výběrem objektu nxtBrickSharp a v pravém okně vlastností přiřadíme periferie k jednotce.



Obrázek 64: Sloučení periferií s jednotkou

V této fázi zbývá nastavit události na objektech, které nám budou ovlivňovat funkce programu. Hlavní okno nám zachytává události stavu kláves s to bud stisknutí klávesy (událost KlavesoveOvladani) nebo puštění klávesy (událost StopPohybu).

```
private void KlavesoveOvladani(object sender, KeyPressEventArgs e){
            label5.Text =  e.KeyChar.ToString();
            klavesa = e.KeyChar.ToString();
            if (nxtBrickSharp.IsConnected){
                switch (klavesa.ToString()){
                    case "s":
                        if (rychlostA < 100){
                            rychlostA = rychlostA + 25;
                            rychlostB = rychlostA;
                        }
                        if (jizda){
                            if (smerjizdy){
                                nxtMotorA.Turn(rychlostA, 0);
                                nxtMotorB.Turn(rychlostB, 0);
                            }
                            else {
                                nxtMotorA.Turn(-rychlostA, 0);
                                nxtMotorB.Turn(-rychlostB, 0);
                            }
                        }
                        SilaMotoru.Text = rychlostA.ToString() + " %";
                        break;
                    case "x":
                        if (rychlostA > 0){
                            rychlostA = rychlostA - 25;
                            rychlostB = rychlostA;
                        }
                        if (jizda){
                            if (smerjizdy){
```

```
                                nxtMotorA.Turn(rychlostA, 0);
                                nxtMotorB.Turn(rychlostB, 0);
                            }
                            else{
                                nxtMotorA.Turn(-rychlostA, 0);
                                nxtMotorB.Turn(-rychlostB, 0);
                            }
                        }
                        SilaMotoru.Text = rychlostA.ToString() + " %";
                        break;
                    case "i":
                        if (rychlostA == 0 || rychlostB == 0){
                            rychlostA = 50;
                            rychlostB = 50;
                            SilaMotoru.Text = rychlostA.ToString()+ " %";
                        }
                        nxtMotorA.Turn(rychlostA, 0);
                        nxtMotorB.Turn(rychlostB, 0);
                        smerjizdy = true;
                        jizda = true;
                        break;
                    case "k":
                        if (rychlostA == 0 || rychlostB == 0){
                            rychlostA = 50;
                            rychlostB = 50;
                            SilaMotoru.Text = rychlostA.ToString()+ " %";
                        }
                        nxtMotorA.Turn(-rychlostA, 0);
                        nxtMotorB.Turn(-rychlostB, 0);
                        smerjizdy = false;
                        jizda = true;
                        break;
                    case "j":
                        nxtMotorA.Coast();
                        break;
                    case "l":
                        nxtMotorB.Coast();
                        break;
    }}}
private void StopPohybu(object sender, KeyEventArgs e){
        klavesa = e.KeyValue.ToString();
        label5.Text = klavesa.ToString();
        if (nxtBrickSharp.IsConnected)
        {
        if (klavesa.ToString()=="73" || klavesa.ToString()=="75"){
                        nxtMotorA.Brake();
                        nxtMotorB.Brake();
                        jizda = false; }
          if (klavesa.ToString()=="74" || klavesa.ToString()=="76"){
                    if (smerjizdy)
                    {
                        if (jizda){
                            nxtMotorA.Turn(rychlostA, 0);
                            nxtMotorB.Turn(rychlostB, 0);
                        }
                    }
                    else
                    {
                        if (jizda) {
                            nxtMotorA.Turn(-rychlostA, 0);
                            nxtMotorB.Turn(-rychlostB, 0);
```

```
                    }}}}}
```

Ovládání myší je umožněno pomocí události vzniklé na tlačítku. Tak je tomu u každého tlačítka pohybu VpredClick,VzadClick, VlevoClick,

VpravoClick, StopClick a OtevriClick.

```
private void VpredClick(object sender, EventArgs e) {
       if (rychlostA > rychlostB)
       {
           rychlostB = rychlostA;
        }
        else
        if (rychlostB > rychlostA){
            rychlostA = rychlostB;
        }
        else
        if ((rychlostA < 100) && (rychlostB < 100)){
                        rychlostA = rychlostA + 25;
                        rychlostB = rychlostB + 25;
                    }
                        nxtMotorA.Turn(rychlostA, 0);
                        nxtMotorB.Turn(rychlostB, 0);
        }
```

Události senzorů nám umožní načítání aktualizaci jejich stavu.

```
private void ZmenaSvetla(Bram.NxtSharp.NxtSensor sensor)
{
   Invoke((MethodInvoker)delegate() {
ZobrazSvetlo.Value =
(int)Bram.Utilities.Util.Clamp(nxtSvetelnySenzor.Value, 0, 100);
 }); }
```

Připojení a odpojení jednotky pomocí Bluetooth:

```
private void PripojitClick(object sender, EventArgs e){
            Pripojit.Enabled = false;

            if (prihlasen == false){
                serialPort1.PortName = JmenoCOMPortu.Text.ToString();
                nxtBrickSharp.COMPortName =
JmenoCOMPortu.Text.ToString();
                NastaveniMotoru.Enabled = false;
                JmenoCOMPortu.Enabled = false;
switch (NastaveniMotoru.Text) {
case "A B C":
                        label5.Text = nxtMotorA.Port.ToString();

nxtBrickSharp.AttachMotor(Bram.NxtSharp.NxtMotorPort.PortA, nxtMotorA);

nxtBrickSharp.AttachMotor(Bram.NxtSharp.NxtMotorPort.PortB, nxtMotorB);

nxtBrickSharp.AttachMotor(Bram.NxtSharp.NxtMotorPort.PortC, nxtMotorC);

break;
case "A C B":
```

```
nxtBrickSharp.AttachMotor(Bram.NxtSharp.NxtMotorPort.PortA, nxtMotorA);

nxtBrickSharp.AttachMotor(Bram.NxtSharp.NxtMotorPort.PortC, nxtMotorB);

nxtBrickSharp.AttachMotor(Bram.NxtSharp.NxtMotorPort.PortB, nxtMotorC);
break;
}
try {
     serialPort1.Open();
    }
    catch (Exception) {
MessageBox.Show("Port neexistuje nebo není dostupný!!", "Chyba spojeni!",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk,
MessageBoxDefaultButton.Button1);
    return;
    }
if (!serialPort1.CtsHolding){
MessageBox.Show("Port neexistuje nebo není dostupný!!", "Chyba spojeni!",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk,
MessageBoxDefaultButton.Button1);
      serialPort1.Close();Pripojit.Enabled = true;
      NastaveniMotoru.Enabled = true;JmenoCOMPortu.Enabled = true;
      }
      else {
      serialPort1.Close();
      serialPort1.Dispose();
      nxtBrickSharp.Connect();
      Pripojit.Enabled = true;
      }
      if ( nxtBrickSharp.IsConnected){
      nxtBrickSharp.StartPolling();
      Vzad.Enabled = true;Vlevo.Enabled = true;
      Vpravo.Enabled = true;Vpred.Enabled = true;
      Stop.Enabled = true;OtevriKlepeta.Enabled = true;
      Pripojit.Text = "Odpojit";
      prihlasen = true;
      SilaBaterie =
(double)nxtBrickSharp.Comm.GetBatteryLevel()/1000;
      Baterie.Text = SilaBaterie.ToString() + " V";
      SilaMotoru.Text = rychlostA.ToString()+" %";
      }}
      else
      if (prihlasen == true) {
      if (nxtBrickSharp.IsConnected) {
     nxtBrickSharp.Disconnect();
     nxtBrickSharp.StopPolling();
     Pripojit.Text = "Připojit";
     prihlasen = false;Vzad.Enabled = false;
     Vlevo.Enabled = false;Vpravo.Enabled = false;
     Vpred.Enabled = false;Stop.Enabled = false;
     OtevriKlepeta.Enabled = false;NastaveniMotoru.Enabled = true;
     JmenoCOMPortu.Enabled = true;
     ZobrazSvetlo.Value=0;ZobrazZvuk.Value = 0;
     Tlak.Text = "-";Sonar.Text = "-";
                       Pripojit.Enabled = true;
                       SilaMotoru.Text = "-";
                       Baterie.Text = "-";
                  } } }
```

4 NÁVRH LABORATORNÍCH ÚLOH

text

4.1 Inteligentní vyhledávač stopy

4.1.1 Zadání

Navrhněte a vytvořte ve vývojovém prostředí LEGO MINDSTORMS NXT (NXT-G) takový ovládací program pro robota NXT Tribot, aby byl schopen automatické detekce podkladu a barevné stopy, po které se bude pohybovat. Program odlaďte na testovací podložce (ovál) z příslušenství stavebnice MINDSTORMS. Robot musí být schopen jízdy po oválu ve směru i protisměru hodinových ručiček.

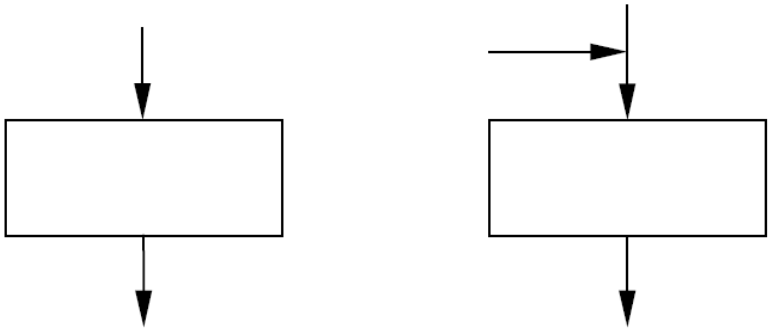
4.1.2 Návrh vývojového diagramu

Vývojový diagram byl vytvořen pomocí programu Diagram Designer 1.2. Jedná se o hrubý náčrt funkce programu.

4.1.2.1 Význam symbolů

4.1.2.1.1 Zpracování

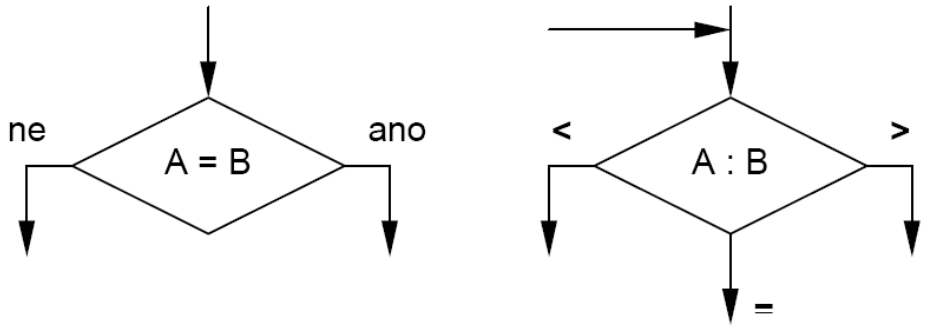
Symbol představující jakýkoliv druh zpracování nebo provedení definované operace nebo skupiny operací, jejichž výsledkem je transformace informace, např. změna hodnoty, umístění apod. Možnost vstupu do tohoto symbolu je z libovolné strany a těchto vstupů může být i několik. Výstup je však v zásadě vždy jenom jeden.



Obrázek 65: Symbol vývojového diagramu – zpracování

4.1.2.1.2 Rozhodování

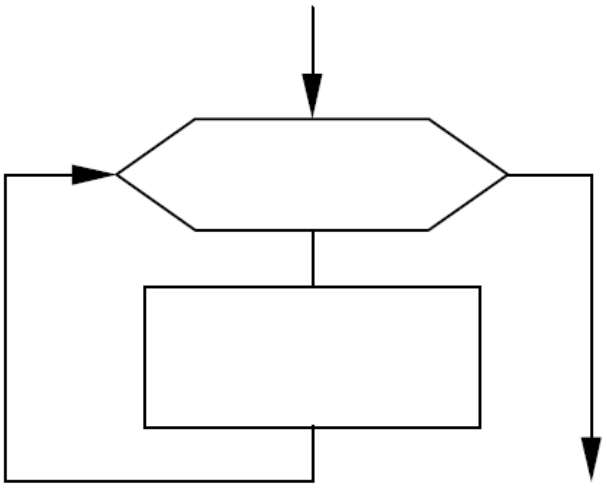
Symbol představuje rozhodovací nebo přepínací funkci. Symbol má jeden vstup a alternativní výstupy. Daný výstup je aktivován po vyhodnocení podmínek uvnitř symbolu.



Obrázek 66: Symbol vývojového diagramu – rozhodování

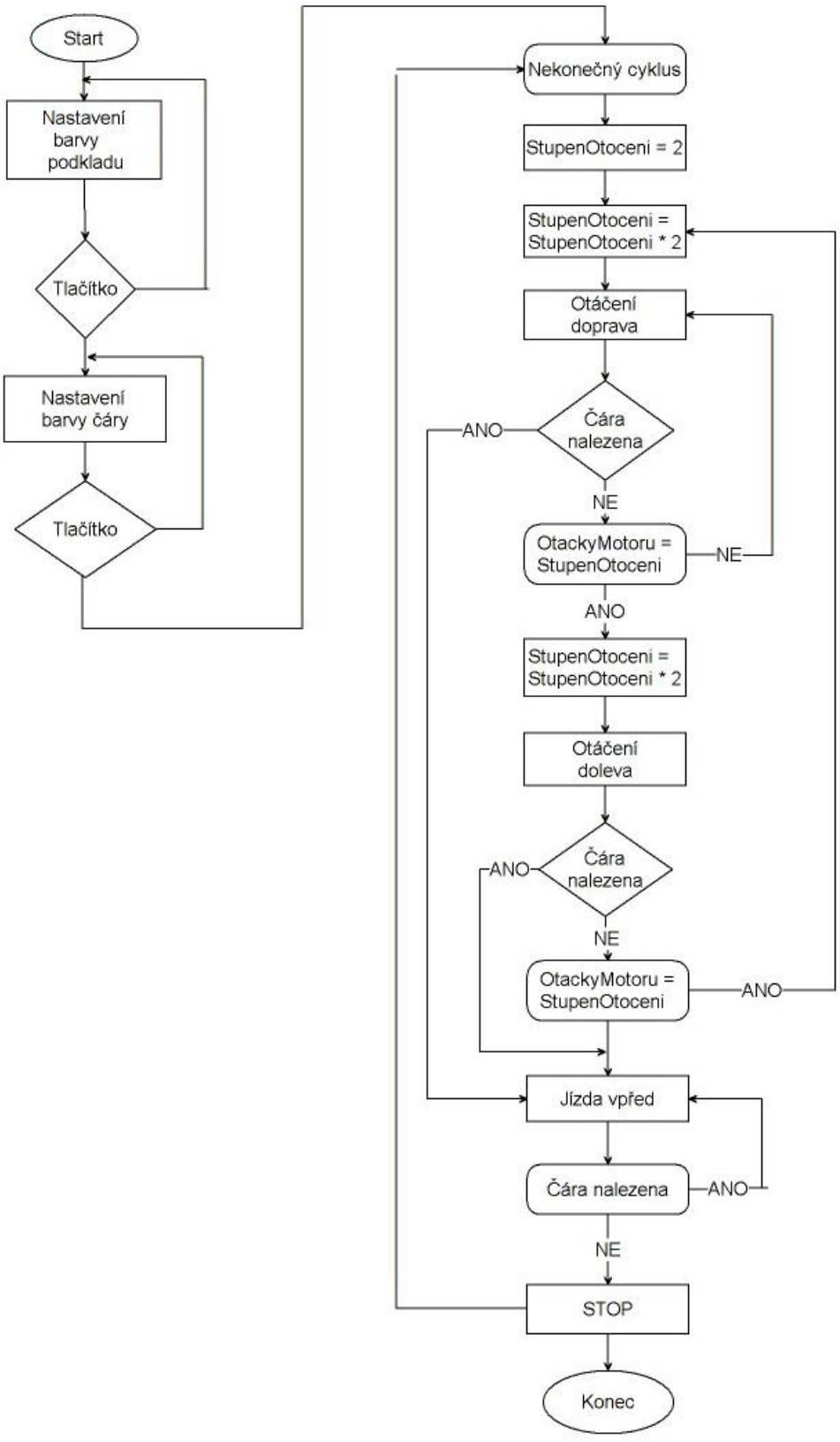
4.1.2.1.3 Cyklus

Tento symbol představuje úpravu nebo modifikaci činnosti, která mění vlastní postup následné činnosti, např. nastavení přepínače, vyjmenování hodnot, kterých nabývá proměnná cyklu, úprava indexového registru a jiné. Symbol má dva vstupy, jeden sekvenční, druhý pro návrat po provedení příslušného bloku operací a dva výstupy, jeden vstupující do daného bloku operací, druhý sekvenční, který pokračuje do další části programu.



Obrázek 67: Symbol vývojového diagramu – cyklus

4.1.2.2 Vývojový diagram

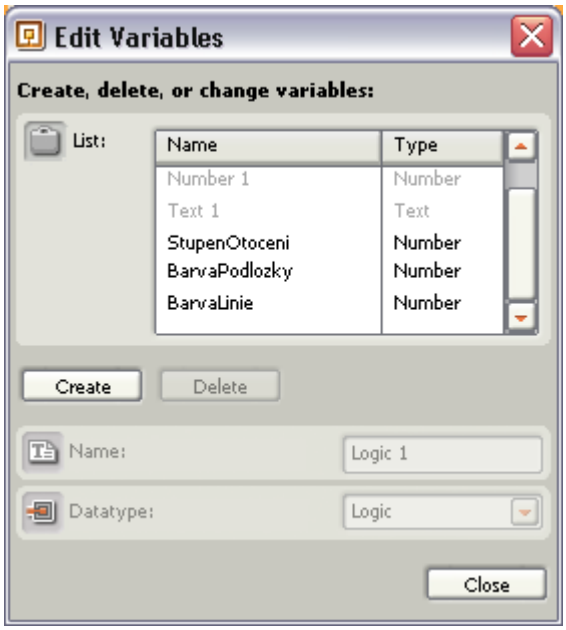


Obrázek 68: Vývojový diagram programu

Program je založen na jednoduchém principu vyhledávání a to tak, že se robot postupně otáčí doprava a doleva o určitý stupeň otočení. Pokud nenalezne linii, zvětší stupeň otočení. Pokud nalezne barevnou linii, spustí jízdu vpřed. Dopředu jede tak dlouho, dokud senzor zaznamenává barevnou linii. Pokud nezaznamená barevnou čáru, zastaví se a spustí se znovu vyhledávání linie. Nastavení barvy podkladu a linie je realizováno pomocí tlakového senzoru. Obsluha robota položí na podklad, stiskne tlakový senzor, následně jej přemístí na barevnou linii a opět stiskne tlakový senzor. Následně se spustí program.

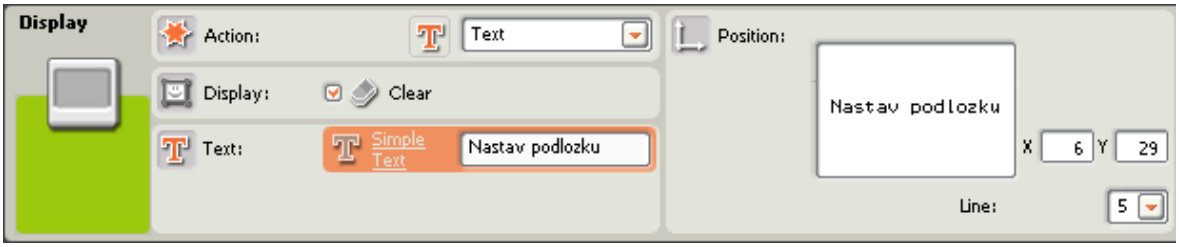
4.1.3 Vytvoření programu

V aplikaci LEGO MINDSTORMS NXT (NXT-G) založíme nový projekt. V položce menu Edit -> Define Variables vytvoříme tři proměnné typu Number, které pojmenujeme: StupenOtoceni, BarvaPodlozky a BarvaLinie.



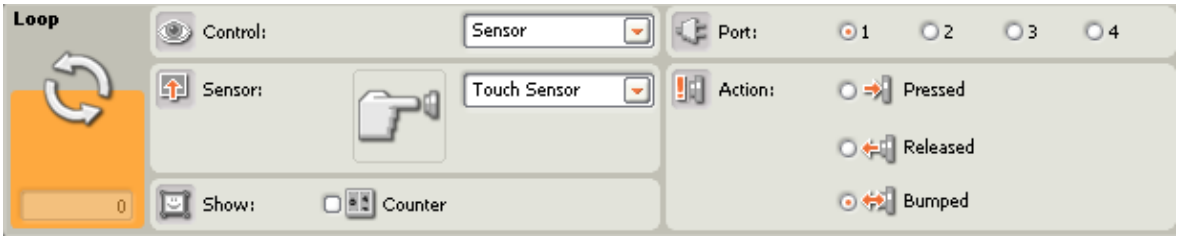
Obrázek 69: Definice proměnných NXT-G

Přemneme se do kompletní palety nástrojů a začneme vybírat potřebné funkční bloky. Jako první začneme výběrem z výstupních operací (Action) bloku Display a stisknutím ltm (levé tlačítko myši) a tažením přesuneme blok na začátek programové linie, která je představována jako bíle kostičky lega. V dialogu pro nastavení bloku vybere akci Text a do příslušného políčka zadáme hodnotu, kterou chceme zobrazit („Nastav podlozku“) a nastavíme souřadnice a řádek umístění.



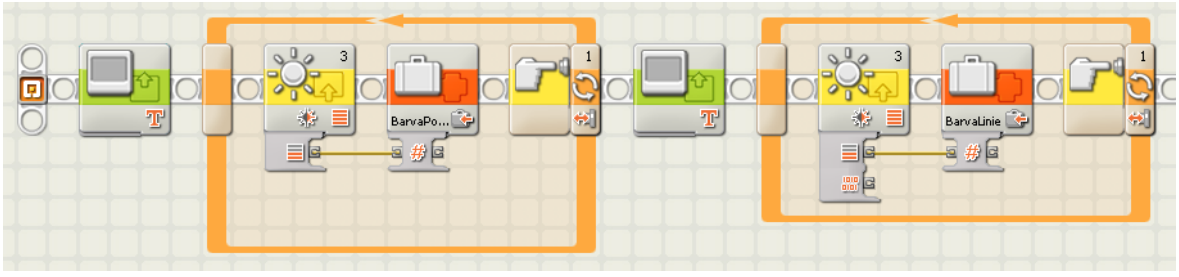
Obrázek 70:Dialog nastavení bloku Display

Jako další prvek vložíme smyčku Loop, u které nastavíme řízení ukončení v závislosti na stisku a puštění tlačítka (Bumped).



Obrázek 71: Dialog nastavení smyčky

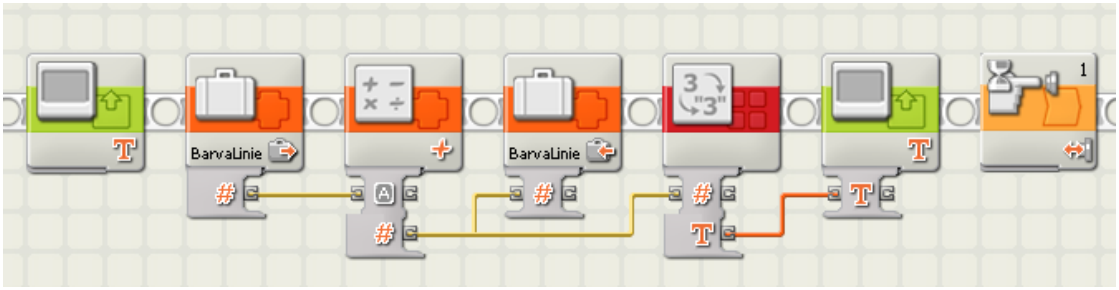
Do smyčky vložíme z palety Sensor blok světelného senzoru. Za něj umístíme z palety Data blok Variable, kterému v dialogovém okně přiřadíme jméno BarvaPodlozky a nastavíme jej do režimu Write. Posunutím kurzoru na spodní okraj bloku světelného senzoru, tak, aby se kurzor změnil na dvojici od sebe směřujících vertikálních šipek a následným kliknutím myši docílíme rozbalení všech pinů. Vybereme pin s názvem Intensity (místo kurzoru se objeví ikona špulky s drátem) a spojíme jej s pinem bloku proměnné. Tento pin vrací procentuální hodnotu intenzity světla. Spojením docílíme, že se načtená hodnota ze světelného čidla uloží do příslušné proměnné. Stejné seskupení bloků vytvoříme pro načtení barvy linie. Při spojování pinů je nutné brát zřetel na jejich druh. Pokud bychom spojili nevhodné piny, čára bude přerušovaná. V případě, že spoj přenáší booleový hodnotu, je zelený. Naopak pokud přenáší číselnou hodnotu, je nažloutlý a v případě, že se jedná o text, je červený.



Obrázek 72: Načtení hodnot barev

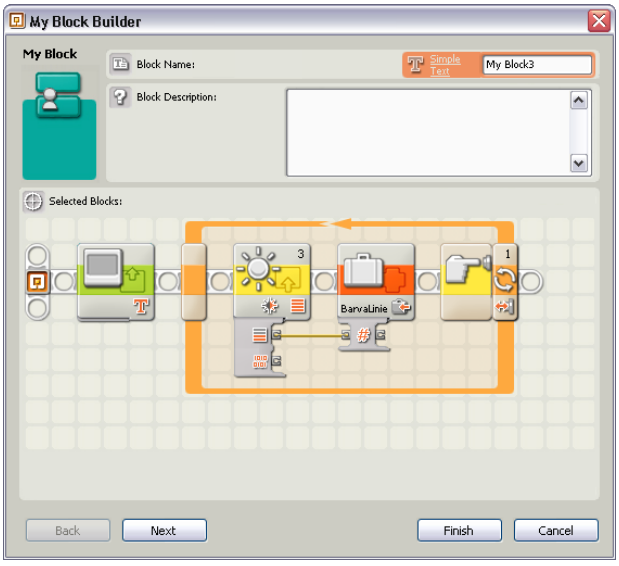
Protože hodnota snímané intenzity světla kolísá v závislosti na změně okolního světla, je nutné hodnoty proměnných upravit tak, abychom docílili požadované citlivosti. V tomto případě se proměnná BarvaPodložky snížila o hodnotu 20 (čím větší číslo odečteme, tím robot dříve zaznamená, že opouští linii) a proměnná BarvaLinie se zvýšila o hodnotu 10.

Výsledné hodnoty následně můžeme zobrazit na displeji jednotky, v prvním bloku zobrazíme popisek proměnné a v druhém bloku Display zobrazíme konkrétní hodnotu, kterou ale musíme přetypovat na text. Za zobrazení hodnot proměnných umístíme čekací blok (paleta Flow -> Wait), který nastavíme opět na událost stisku tlačítka.



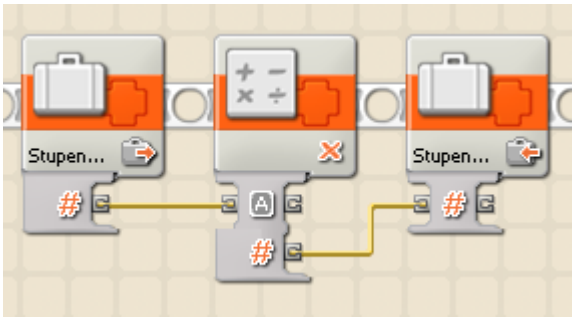
Obrázek 73: Změna a zobrazení proměnné

Pro zpřehlednění programu můžeme z této posloupnosti bloků udělat jeden vlastní. Tedy vytvoříme nějako jako funkci programu. To provedeme tak, že tažením myši označíme patřičné bloky a vybereme z hlavní nástrojové lišty tlačítko Create Block. Zde vyplníme jméno a popis bloku a pokračujeme tlačítkem Next, kde nastavíme ikonu bloku. Místo skupiny bloků tak vidíme pouze jeden s příslušným jménem.



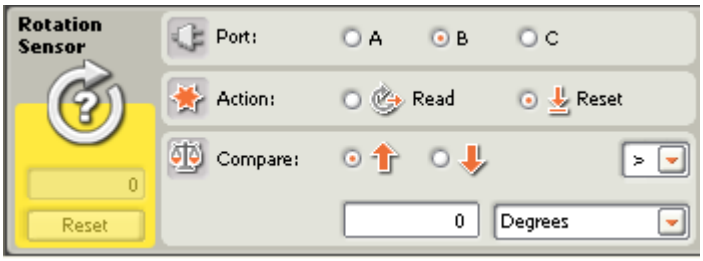
Obrázek 74: Vytvoření vlastního bloku

Nyní už se dostáváme k vytvoření vlastního algoritmu vyhledávání. Základem bude nekonečná smyčka, ve které bude jako první blok proměnná StupenOtoceni v režimu Write a nastavíme mu hodnotu 2. Následující blok bude opět smyčka, kterou bude ukončovat booleova hodnota. V této smyčce začínáme matematickou operací, která dvakrát zvětší hodnotu proměnné StupenOtoceni.



Obrázek 75: Zvětšení stupně otočení

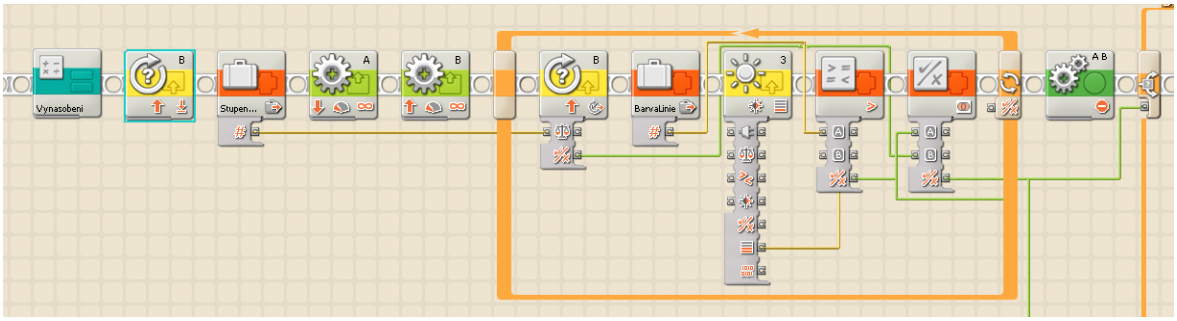
Tuto operaci můžeme opět sjednotit do jednoho bloku Vynasobeni. Dále vložíme blok z palety Sensor s názvem Rotation Sensor ve kterém si vynulujeme senzor otáček pravého kola (port B), protože využijeme jeho funkce k získání hodnoty otáčení doleva.



Obrázek 76: Dialog nastavení rotačního senzoru

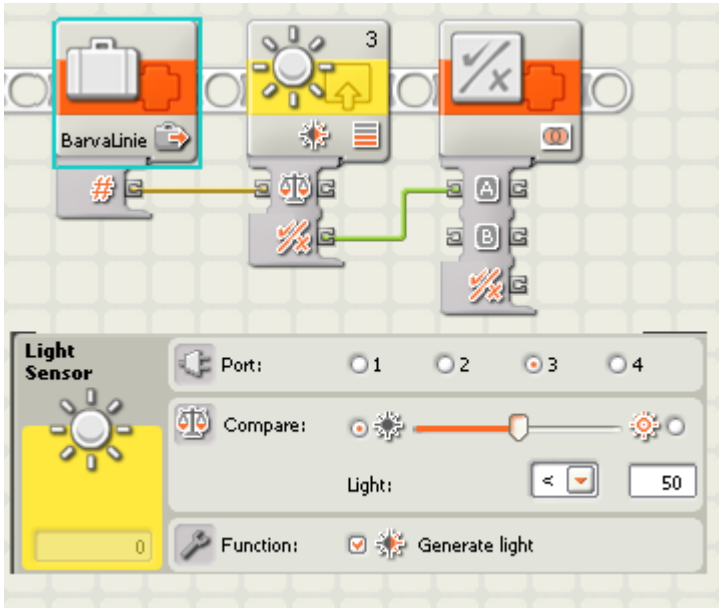
Dále následuje blok proměnné StupenOtoceni v režimu Read a za ním jsou dva bloky nastavující pohyb motorů. Levý motor směřuje vzad a pravý vpřed. Oba jsou nastavené na 30% výkonu. Jako další blok je smyčka ukončená pravdivostní hodnotou, ve které se jako první nachází senzor pravého kola, do kterého je na aktivační pin (Trigger Point) přivedena hodnota proměnné StupenOtoceni. Další tři bloky jsou složeny z proměnné BarvaLinie, světelného senzoru a porovnávací funkce. Jejich funkce je následující: pokud je hodnota na senzoru menší než hodnota v proměnné BarvaLinie vrátí porovnávací funkce pravdivou hodnotu. Tento výsledek, spolu s výsledkem senzoru pravého kola (vrací pravda, pokud otáčky motoru dosáhly hodnoty v proměnné StupenOtoceni) jsou přivedeny do logické funkce OR, jejíž výsledek ovlivňuje ukončení smyčky. V praxi to znamená, že pokud

světelný senzor zaznamenal čáru a nebo otáčky motoru dosáhly požadované hodnoty dojde k ukončení smyčky a motory se zastaví.



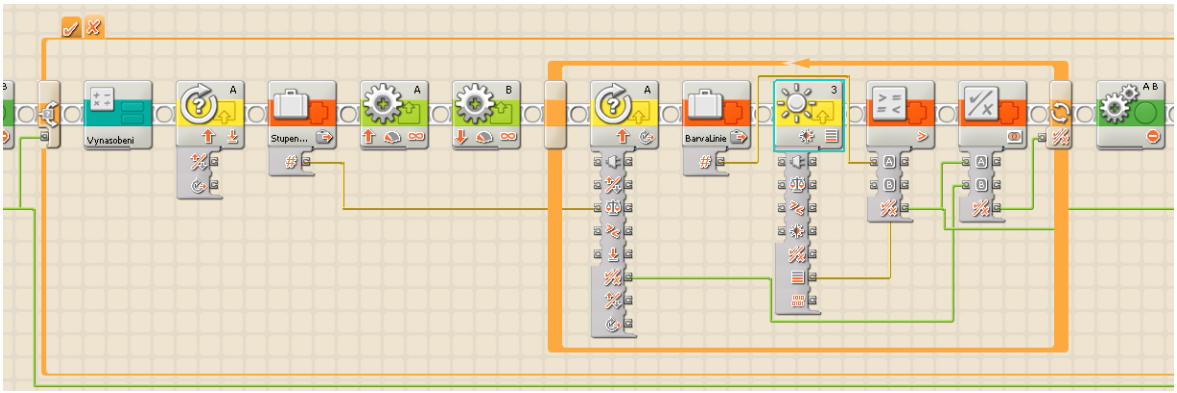
Obrázek 77: Otáčení doleva

Porovnávací funkci světelného senzoru a proměnné lze realizovat i způsobem uvedeném na Obrázek 78. V tomto případě musíme dát pozor na nastavení položky Compare (vrací pravdu, jeli hodnota senzoru menší než hodnota přivedená na pinu).



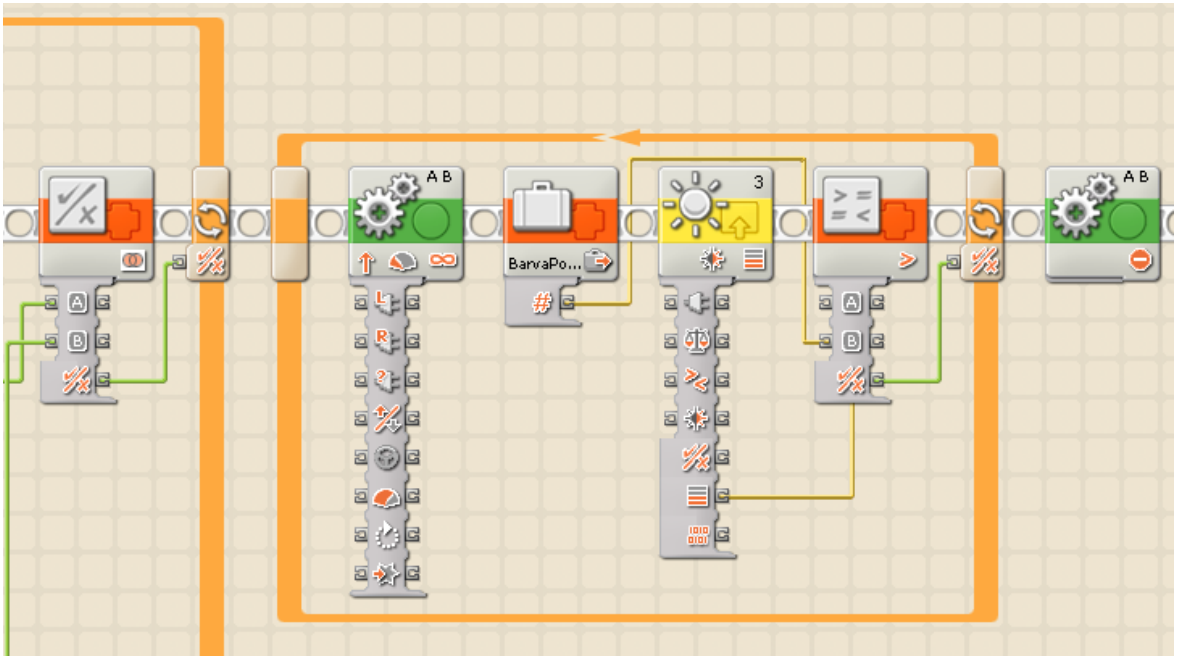
Obrázek 78: Porovnání hodnoty senzoru s proměnnou

Nalezení čáry ovlivní následující blok Switch, ve kterém je obsaženo otáčení doprava stejným způsobem, jako tomu bylo při otáčení doleva. Takže pokud byla naleznuta čára a tedy porovnávací funkce vrací pravdu, tak se blok přeskočí. Pokud nebyla nalezena, provádí se otáčení doprava.



Obrázek 79: Otáčení doprava

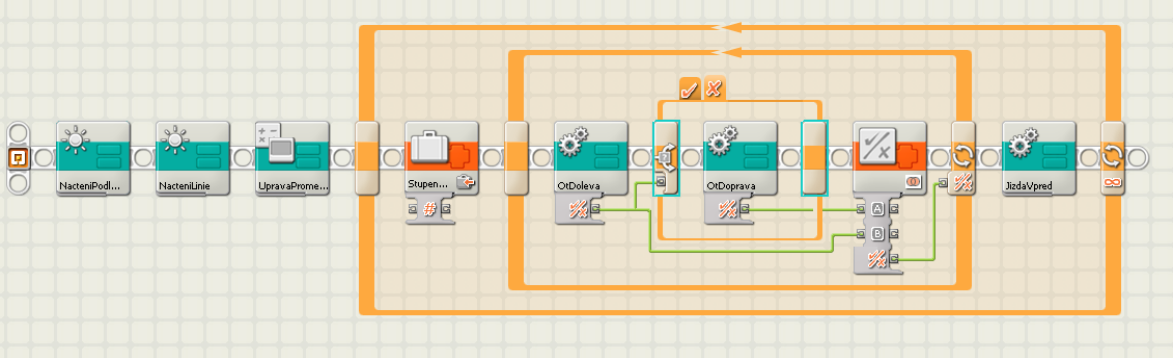
Při procesu otáčení doprava se opět vynásobí proměnná StupenOtoceni hodnotou 2, čímž se dosáhne návratu robota zpět do původní pozice a otočení o stejný stupeň doprava, jako tomu bylo u otáčení doleva. Výsledky logických funkcí nalezení čáry z obou směrů se přivedou do následující logické funkce, která ukončí celý proces otáčení a spustí jízdu vpřed.



Obrázek 80: Smyčka jízdy vpřed po linii

Jízda vpřed je realizována pomocí smyčky, kterou ukončuje pravdivostní hodnota porovnávací funkce, která porovnává hodnotu světelného senzoru a hodnotu BarvaPodlozky. Pokud je hodnota senzoru větší než uložená hodnota v proměnné, ukončí se smyčka jízdy, zastaví se motory a program se vrací na začátek nekonečné smyčky, kde se znovu nastaví stupeň otáčení na hodnotu 2 a spustí se vyhledávání čáry formou otáčení.

Výsledný program po zformování do bloků vypadá následovně.



Obrázek 81: Výsledný program sledování linie

Nyní zbývá jen stáhnout program do jednotky NXT pomocí USB kabelu a provést vyzkoušení a případně doladění citlivosti senzorů.

4.1.4 Závěr

Pomocí vývojového prostředí NXT-G byl vytvořen program, který řídí robota NXT Tribot po černé linii zobrazené na testovací podložce, která je příslušenstvím stavebnice MINDSTORMS NXT (test pad). Předvedení v programu v praxi ukázalo, že světelný senzor hodně ovlivňuje okolní osvětlení. Snížení intenzity okolního světla zapříčinilo změnu dekované hodnoty bílé plochy podložky.

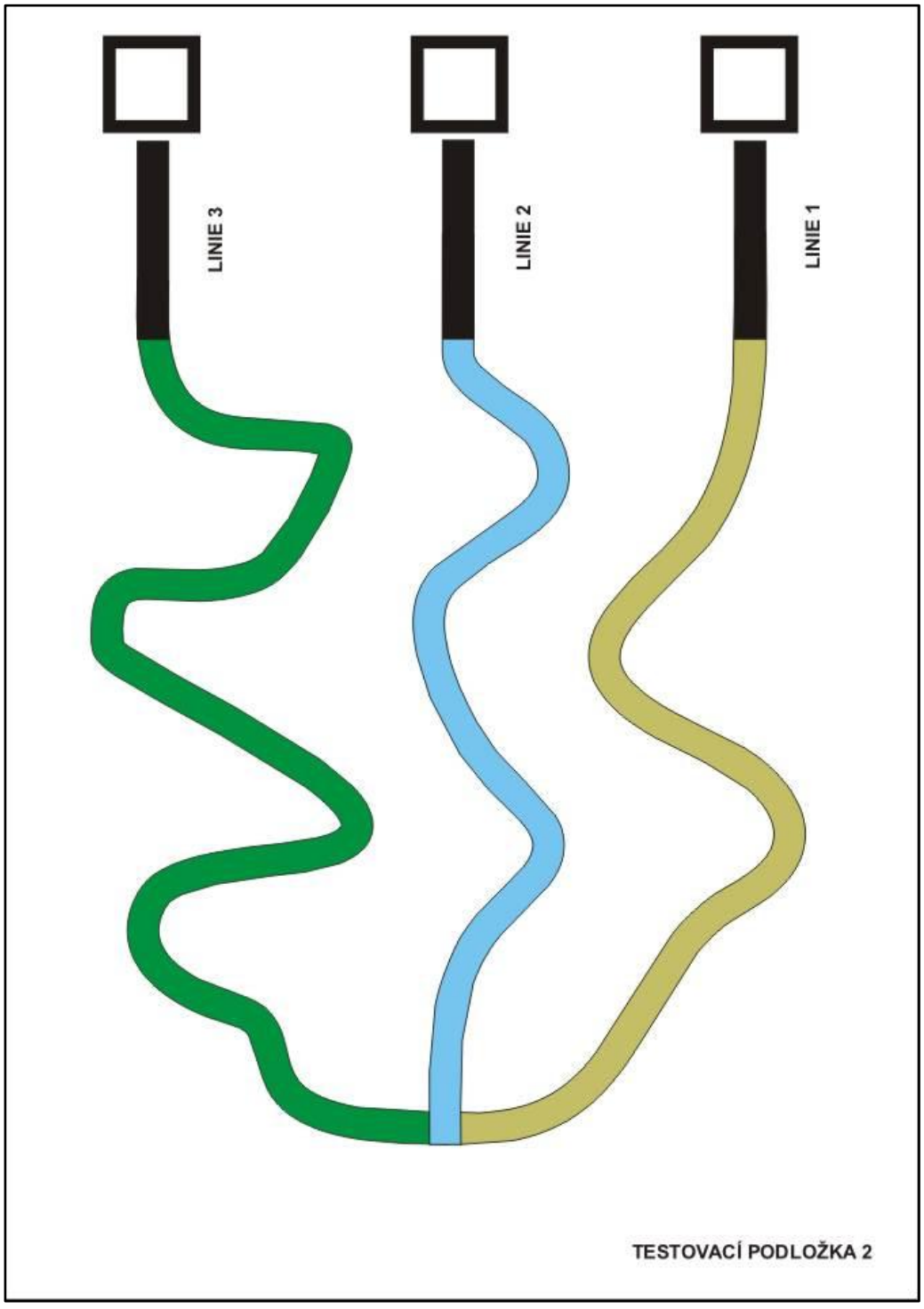
Dalším faktorem ovlivňující správnou funkčnost programu je stav napájecích baterii řídící jednotky. Při nedostatečném napájení dochází k nesynchronizovanému otáčení motorů, což má nežádoucí vliv především na jízdu vpřed.

4.2 Automatický převozník nákladu

4.2.1 Zadání

Pomocí vývojového prostředí BricxCC vytvořte program pro robota NXT Tribot, který bude mít za úkol dovézt kuličku na úložiště na konci linie 1 testovací podložky číslo 2, vyložit ji a přepravit se k úložišti na linii 2, pak naložit kuličku a převést na zbylé úložiště na linii 3.

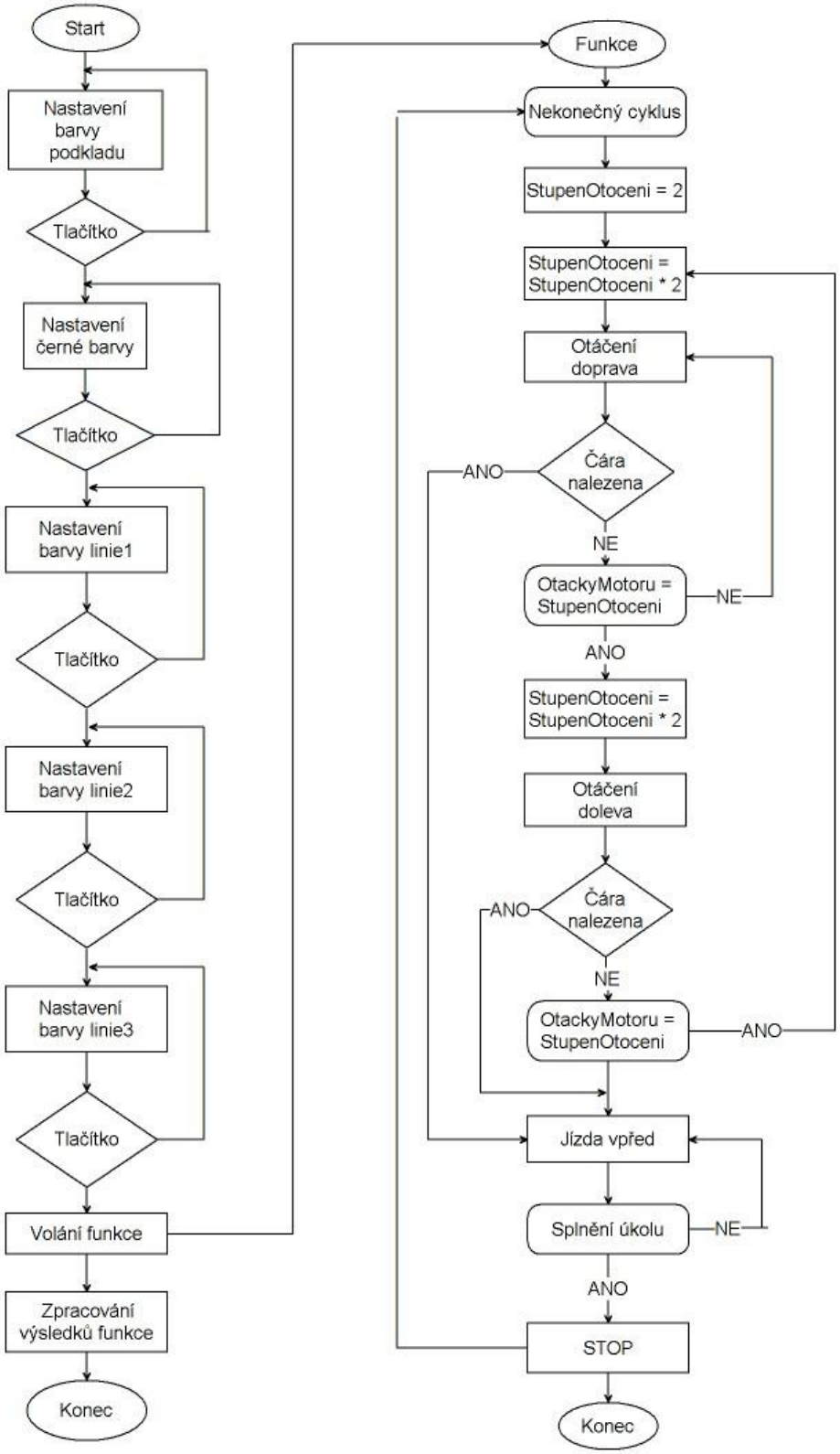
4.2.2 Testovací podložka



Obrázek 82: Testovací podložka

Testovací podložka byla vytvořena na papíru formátu A1.

4.2.3 Návrh vývojového diagramu

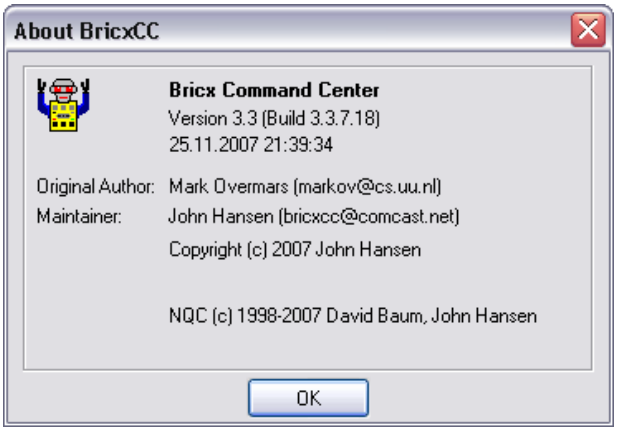


Obrázek 83: Vývojový diagram

4.2.4 Vytvoření programu

4.2.4.1 Popis prostředí

Program byl vytvořen pomocí vývojového prostředí BricxCC verze 3.3 test release a programovacího jazyka NXC.



Obrázek 84: Verze vývojového prostředí BricxCC

V řídící jednotce byl použit originální firmware od společnosti LEGO Group verze 1.05.

4.2.4.2 Zdrojový kód

Základem programu je načtení barev podkladu a jednotlivých linií. Tuto operaci musí zajistit obsluha robota. K nastavení jednotlivé barvy je vyzvána na displeji jednotky. Obsluha položí robota tak, aby světelný senzor byl na potřebné barvě a potvrdí nastavení stisknutím tlakového senzoru. Program využívá jednu funkci, která se v těle volá s 5 parametry, kterými jsou barva čáry, po které robot jede, barva podložky, koncová barva, na kterou má robot dojet, rychlost pohybu a úkol, který funkci ukončí. Po splnění úkolu vrací funkce pravda.

4.2.4.2.1 Definice globálních proměnných a fuknce:

```
#define tlacitko SENSOR_1     // pojmenovanisenzoru
#define svetlo SENSOR_3
int BarvaLiniel, BarvaLinie2, BarvaLinie3,
    BarvaPodkladu, BarvaCerna; //globalni promenne
// definice funkce
inline bool SledujCaru (int BarvaCary, int BarvaPodlozky,
                        int KoncovaBarva, int rychlost, int ukol)
{
int StupenOtoceni;
bool Nalezeno=false;
bool OtaceniDoPrava=true;
bool OtaceniDoLeva=false;
```

4.2.4.2.2 Tělo funkce:

```
//opakuj az do koncoveho ukolu
do
{
          StupenOtoceni = 2;
   do //Hledani linie doprava
   {
          StupenOtoceni=StupenOtoceni*2;
          ResetTachoCount(OUT_A);
          OnFwd(OUT_A,rychlost);
          OnRev(OUT_B,rychlost);
          do
          {
          SetSensorLight(IN_3);
          if (svetlo <= BarvaCary)
          {
          Nalezeno = true;
          OtaceniDoPrava = false;
          }
          else
          if(MotorTachoCount(OUT_A)>StupenOtoceni)
          {
          OtaceniDoPrava=false;
          OtaceniDoLeva=true;
          }
          }
          while(OtaceniDoPrava);     //Konec hledani doprava

          if (Nalezeno==false)       //Spusteni hledani doleva
          {
          StupenOtoceni=StupenOtoceni*2;
          ResetTachoCount(OUT_B);
          OnFwd(OUT_B,rychlost);
          OnRev(OUT_A,rychlost);

          do
          {
           SetSensorLight(IN_3);
            if (svetlo <= BarvaCary)
              {
              Nalezeno = true;
              OtaceniDoLeva = false;
              }
              else
              if(MotorTachoCount(OUT_B)>StupenOtoceni)
              {
              OtaceniDoLeva=false;
              OtaceniDoPrava=true;
              }
           }
           while(OtaceniDoLeva); // konec hledani doleva

           } //konec if nalezeno false
  }
while(Nalezeno==false); //Konec otaceni
```

```
while(Nalezeno==false); //Konec otaceni

Nalezeno=false;
OtaceniDoPrava=true;
OnFwd(OUT_AB,rychlost);     //jizda po care

// plneni ukolu
do
{
switch (ukol)
{
       case 1:
            SetSensorLight(IN_3);
            if (svetlo <= KoncovaBarva)
            {
             Off(OUT_AB);
             Wait(500);
             return true;
             }
        break;
        case 2:
             SetSensorTouch(IN_1);
             if (tlacitko==1)
             {
             Off(OUT_AB);
             Wait(500);
             return true;
             }

        break;
        case 3:
           SetSensorLight(IN_3);
            if (svetlo > BarvaCary && svetlo <= KoncovaBarva)
            {
             Off(OUT_AB);
             Wait(500);
             return true;
             }
        break;

}
}
while((svetlo < BarvaPodkladu) ); //Konec jizdy po care
Off(OUT_AB);

SetSensorLight(IN_3);
}
while (true);
Off(OUT_AB);
return true;
}
```

Funkce zajišťuje jízdu po dané linii. Vyhledávání probíhá postupným otáčením se doprava a doleva o postupně zvyšující se úhel otočení. Vyhledávání probíhá tak dlouho, dokud světelný senzor nenarazí na požadovanou barvu. Když detekuje barvu linie, spustí jízdu

vpřed, dokud senzor nenarazí na barvu podkladu nebo nedojte ke splnění úkolu. Úkol spočívá buď v nalezení cílové barvy a nebo dotyku čidla s úložištěm. Pokud splní funkce svůj úkol, vrací hodnotu pravda.

4.2.4.2.3 Tělo programu, načtení barev:

```
task main()
{
 //nacteni barvy podkladu
TextOut(5,LCD_LINE3,"Set Podklad",true);
do{
SetSensorLight(IN_3);
BarvaPodkladu = Sensor(IN_3);
SetSensorTouch(IN_1);
} while (tlacitko == 0);
NumOut(12,LCD_LINE4,BarvaPodkladu);
Wait(1000);
//nacteni cerne barvy
TextOut(5,LCD_LINE3,"Set Cerna",true);
do {
SetSensorLight(IN_3);
BarvaCerna = svetlo;
SetSensorTouch(IN_1);
} while (tlacitko == 0);
NumOut(12,LCD_LINE4,BarvaCerna);
Wait(1000);
//nacteni barvy linie1
TextOut(5,LCD_LINE3,"Set Linie 1",true);
do {
SetSensorLight(IN_3);
BarvaLiniel = svetlo;
SetSensorTouch(IN_1);
} while (tlacitko == 0);
NumOut(12,LCD_LINE4,BarvaLiniel);
Wait(1000);
//nacteni barvy linie2
TextOut(5,LCD_LINE3,"Set Linie 2",true);
do {
SetSensorLight(IN_3);
BarvaLinie2 = svetlo;
SetSensorTouch(IN_1);
} while (tlacitko == 0);
NumOut(12,LCD_LINE4,BarvaLinie2);
Wait(1000);
//nacteni barvy linie3
TextOut(5,LCD_LINE3,"Set Linie 3",true);
do {
SetSensorLight(IN_3);
BarvaLinie3 = svetlo;
SetSensorTouch(IN_1);
} while (tlacitko == 0);
NumOut(12,LCD_LINE4,BarvaLiniel);
Wait(1000);
```

Po načtení jednotlivých barev je nutné je upravit tak, aby s nimi mohl světelný senzor bez problému pracovat. Jde vlasně o jakési nastavení citlivosti, protože světelný senzor není přesný a během jízdy intenzita odraženého světla kolísá.

```
//Nastveni citlivosti senzoru

BarvaCerna=BarvaCerna+2;
BarvaPodkladu=BarvaPodkladu-3;
BarvaLiniel=BarvaLiniel+3;
BarvaLinie2=BarvaLinie2+3;
BarvaLinie3=BarvaLinie3+3;
ClearScreen ();
TextOut(5,LCD_LINE2,"Upravene barvy",true);
NumOut(12,LCD_LINE3,BarvaPodkladu);
NumOut(12,LCD_LINE4,BarvaCerna);
NumOut(12,LCD_LINE5,BarvaLiniel);
NumOut(12,LCD_LINE6,BarvaLinie2);
NumOut(12,LCD_LINE7,BarvaLinie3);
 Off(OUT_C);
//cekani na stisk tlacitka a zahajeni programu
SetSensorTouch(IN_1);
until(tlacitko == 1);
Off(OUT_C);
```

Pak se na displeji zobrazí upravené hodnoty barev a program čeká na postavení robota na linii 1 s potvrzení stiskem tlakového senzoru.

4.2.4.2.4 Vlastní funkce programu

Jádrem programu je volání funkce, která plní požadované úkoly. Tedy robot dojede na požadované místo nebo provede požadovanou operaci a vrátí úspěch vykonání. Tudíž z těla programu jen zavoláme funkci s příslušným úkolem. Úkol 1 doveze robota po požadované linii k černému zakončení. Pak následuje opět volání stejné funkce, ale barva po které se má robot pohybovat je černá a plní úkol číslo 2. V tomto úkolu robot pomaloučku přijíždí k úložišti. Až je stlačen tlakový senzor, ukončí se funkce a vrátí úspěch. Následně je potřeba couvnout od úložiště a otočit robota. Otáčení je provedeno pomocí čekací operace Until, která čeká, doku senzor při otáčení nenarazí na barvu linie, po které se bude vracet. Po otočení se opět volá funkce, tentokrát s úkolem číslo 3, který má najít následující linii. Program končí po vyložení kuličky na barevné linii 3.

V paměti řídící jednotky tento program zabírá méně, než program pro sledování černé linie vytvořený v originálním prostředí, přičemž je mnohem složitější.

```
if (SledujCaru(BarvaLiniel,BarvaPodkladu-5,BarvaCerna,35,1)){
 if (SledujCaru(BarvaCerna,BarvaCerna+10,0,25,2)) {
 RotateMotor(OUT_C,20,60);  Wait(500);
 RotateMotor(OUT_AB,30,-180);
 RotateMotor(OUT_C,20,-61);  Off(OUT_C);
 ResetTachoCount(OUT_B); OnRev(OUT_B,40); OnFwd(OUT_A,40); Wait(500);
 SetSensorLight(IN_3);
 until (svetlo>BarvaCerna && svetlo<BarvaLiniel);
 Off(OUT_AB); Wait(500);
 if (SledujCaru(BarvaLiniel,BarvaPodkladu,BarvaLinie2-3,30,1)) {
  OnFwd(OUT_A,40); Wait(500);
 SetSensorLight(IN_3);
 until (svetlo>BarvaCerna && svetlo<=BarvaLinie2);
 Off(OUT_AB);  Wait(500);
  if (SledujCaru(BarvaLinie2,BarvaPodkladu-10,BarvaCerna,30,1))  {
    ResetTachoCount(OUT_C); RotateMotor(OUT_C,20,60); Wait(500);
   if (SledujCaru(BarvaCerna,BarvaCerna+10,0,20,2)) {
       ResetTachoCount(OUT_C); RotateMotor(OUT_C,30,-61);Wait(500);Off(OUT_C);
       RotateMotor(OUT_AB,30,-180); Off(OUT_C);
       OnRev(OUT_B,40);OnFwd(OUT_A,40); Wait(500);
       SetSensorLight(IN_3);
       until (svetlo>BarvaCerna && svetlo<BarvaLiniel);
       Off(OUT_AB); Wait(500);
       if (SledujCaru(BarvaLinie2,BarvaPodkladu-12,BarvaLinie3,30,1)) {
          Wait(400);
          if (SledujCaru(BarvaLinie3,BarvaPodkladu-15,BarvaCerna,30,1)){
            if (SledujCaru(BarvaCerna,BarvaCerna+10,0,20,2)) {
             RotateMotor(OUT_C,20,60); Wait(500);
             RotateMotor(OUT_AB,30,-180);
             RotateMotor(OUT_C,30,-61); Off(OUT_C);
             OnRev(OUT_B,40);OnFwd(OUT_A,40);Wait(500);
             SetSensorLight(IN_3);
             until (svetlo>BarvaCerna && svetlo<BarvaLiniel);
             Off(OUT_AB); Wait(500);
             if (SledujCaru(BarvaLinie3,BarvaPodkladu-10,BarvaLinie2-5,30,3))
}}}}}}}}}
```

4.2.5 Závěr

Program BricxCC plně nahradil originální vývojové prostředí pro robota NXT. Výsledný program zabírá v paměti NXT 16KB. Jelikož světelný senzor není přesný, je potřeba při nastavování barevných linii dodržet přesnost nastavení čidla na barvu. Ke správnému vykonávání programu je nutné mít dostatečné nabité baterie, jinak krokování motorů selže.

ZÁVĚR

Cílem teoretické části této diplomové práce bylo zpracovat informace o softwarovém a hardwarovém vybavení stavebnice LEGO MINDSTORMS a vyhledat možnosti programovacích jazyků.

V praktické části se jednalo o vytvoření ovládacího programu robota NXT, což bylo realizováno pomocí jazyka NXT# ve vývojovém prostředí MS Visual Studio 2008. Program komunikuje s robotem pomocí bezdrátové technologie Bluetooth a chová se tedy jako dálkové ovládání se zpětnou odezvou ze senzorů robota. Druhá část práce byla věnována návrhu dvou laboratorních úloh včetně dokumentace a vypracování. První laboratorní úloha využívá možností samotné stavebnice. Pro druhou úlohu byla vytvořena barevná podložka, na níž robot vykonává svou činnost.

ZÁVĚR V ANGLIČTINĚ

The main aim of the theoretical part of this master thesis was to elaborate information about software and hardware equipments of construction kit LEGO MINDSSTORMS NXT and also to found all possibilities of programming languages.

Practical part is focused on the development of control application for NXT robot. This application was created by means of NXT# language in development program MS Visual Studio 2008. The application communicates with robot via Bluetooth technology and behaves like remote control with backward response from robots sensors. The second part deals with the proposal of laboratory tasks including all documentation and exemplary elaborations. The first laboratory task takes advantages of construction kit. For the purpose of second laboratory task a color pad was made, which is used by robot to perform its own activity.

SEZNAM POUŽITÉ LITERATURY

[1] BAGNALL, Brian. *Maximum LEGO NXT : Building Robots with Java Brains*. Editied by Sylvia Philipps. 1st edition. Canada : Variant Press, c2007. 505 s. ISBN 978-0-9738649-1-5.

[2] PERDUE, David J. *THE UNOFFICIAL LEGO MINDSTORMS NXT INVENTOR\'S GUIDE*. Megan Dunchak; Christina Samuell. 1st edition. San Francisco : No Starch Press, Inc, c2008. 296 s. ISBN 978-1-59327-154-1.

[3] ASTOLFO, Dave, FERRARI, Mario, FERRARI, Giulio. *BUILDING ROBOTS WITH LEGO MINDSTORMS NXT*. Audrey Doyle. 1st edition. Burlington : Syngress Publishing, Inc, c2007. 447 s. ISBN 978-1-59749-152-5.

[4] SCHOLZ, Matthias Paul. *Advanced NXT: The Da Vinci Inventions Book*. Jennifer Whipple. Apress, c2007. 369 s. ISBN 978-1-59059-843-6.

[5] GASPERI, Michael, HURBAIN, Philippe, HURBAIN, Isabelle. *Extreme NXT : Extending the LEGO MINDSTORMS NXT to the Next Level*. Susannah Davidson Pfalzer. [s.l.] : Apress, c2007. 286 s. ISBN 978-1-59059-818-4.

[6] BOOGAARTS, Martijn, et al. *THE LEGO MINDSTORMS NXT IDEA BOOK : design, invent, and built*. Nancy Sixsmith, Megan Dunchak. 1st edition. San Francisco : No Starch Press, Inc, c2007. 344 s. ISBN 978-1-59327-150-3.

[7] Hardware Development Kit – balíček dokumentů PDF dostupný na stránkách www.mindstorms.lego.com

[8] Návod k vývojovému prostředí LeJOS dostupný na stránkách http://lejos.sourceforge.net/index.php

[9] Návod k vývojovému prostředí BricxCC dostupný na stránkách http://bricxcc.sourceforge.net/nbc/

[10] Návod k vývojovému prostředí RobotC.net dostupný na stránkách http://www.robotc.net/index.html

[11] Dokumentační balíček dostupný na stránkách http://www.legoeducation.com/

[12] Originální stránky produktu BrickOS dostupné na adrese http://brickos.sourceforge.net/

[13] Informace dostupné na internetové adrese http://nxtsharp.fokke.net/

[14] Podporující balíčky k produktu NXT Education dostupné na adrese http://www.active-robots.com

[15] Popis produktů na stránkách společnosti HITECHNIC http://www.hitechnic.com/

[16] Srovnávací test vývojových prostředí dostupný na adrese http://www.zive.cz/Titulni-strana/Tri-veterani-uzivatelske-srovnani-integrovanych-prostredi-pro-jazyk-C/Microsoft-Visual-C-2005-Express-Borland-Turbo-C-2006-Explorer/sc-21-sr-1-a-135680-ch-53813/default.aspx

[17] Informace dostupné na stránkách http://www.hluchak.cz/~krhanek/?q=node/1414

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ltm	Levé tlačítko myši
ptm	Pravé tlačítko myši
BT	BlueTooth
FW	Firmware

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

1. CD s elektronickou podobou diplomové práce
2. Zadání laboratorní úlohy 1
3. Zadání laboratorní úlohy 2

PŘÍLOHA P I: ZADÁNÍ LABORATORNÍ ÚLOHY 1

Inteligentní vyhledávač stopy

1. Zadání

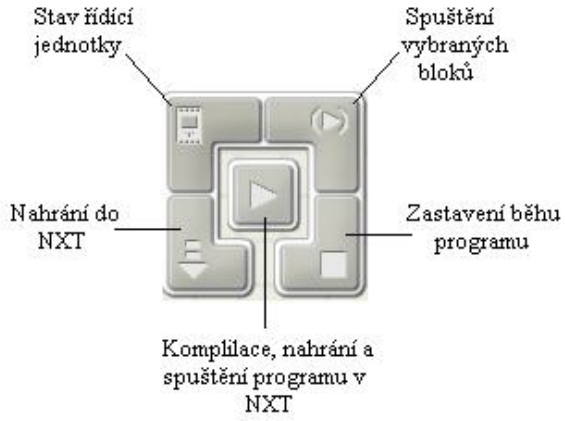
Navrhněte a vytvořte ve vývojovém prostředí NXT-G takový ovládací program pro robota NXT Tribot, aby byl schopen automatické detekce podkladu a barevné stopy, po které se bude pohybovat. Program odlaďte na testovací podložce (ovál) z příslušenství stavebnice MINDSTORMS. Robot musí být schopen jízdy po oválu ve směru i protisměru hodinových ručiček.

2. Návrh vývojového diagramu

Sestavte vývojový diagram ovládacího programu (například v programu SmartDraw, Diagram Designer 1.20 nebo MSWord) a popište funkci jednotlivých bloků.

3. Vytvoření programu

1. Spusťte vývojové prostředí NXT-G a seznamte se uživatelským rozhraním
2. Přetahováním funkčních bloků z nástrojové palety (doporučuji používat kompletní paletu) na časovou linii sestavte program.
3. Vytvořený program stáhněte přes USB kabel pomocí ovládacího prvku v pravém dolním rohu aplikace do řídící jednotky NXT.

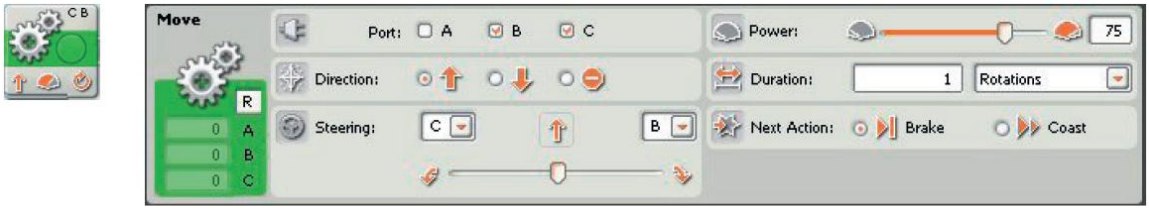


4. Spusťte a odlaďte vytvořený program s robotem NXT Tribot na testovací podložce.
5. Vypracujte protokol

Základní funkční bloky:

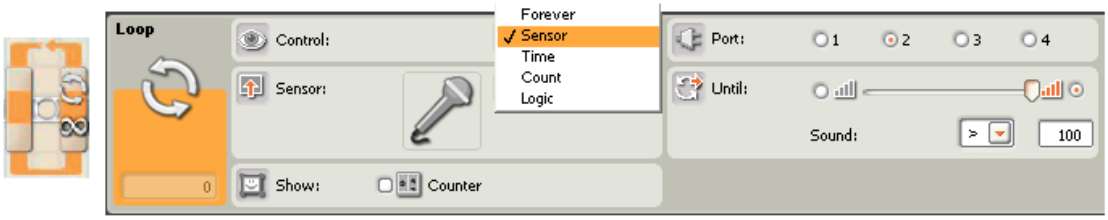
Pohyb motorů:

Umožňuje nastavování rychosti, směru, brždění, dobu chodu a výběr motoru.



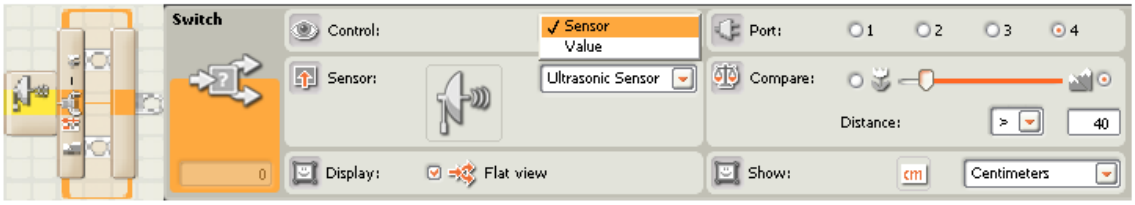
Opakovací smyčka:

Umožňuje opakování akce na základě stanovené podmínky.



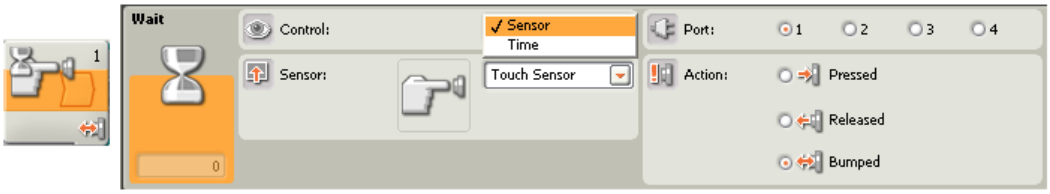
Podmínka:

Jako podmínka může sloužit údaj senzoru nebo vstupní logická hodnota. Podle vyhodnocení podmínky se provede daná operace. (Alternativa k if{} else{}).



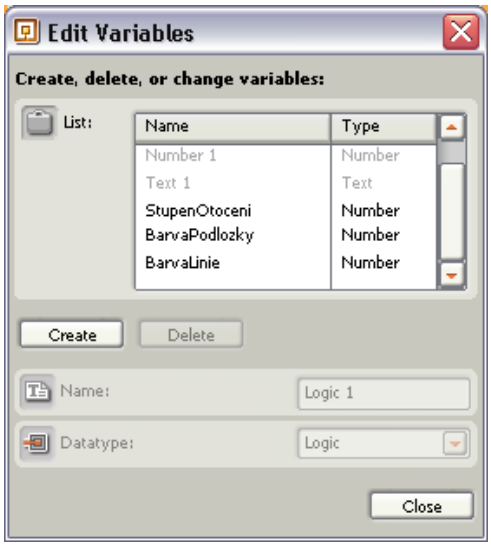
Čekání na událost:

Zastaví chod programu dokud nenastane nějaká událost na čidlech nebo dokud nevyprší časový limit.

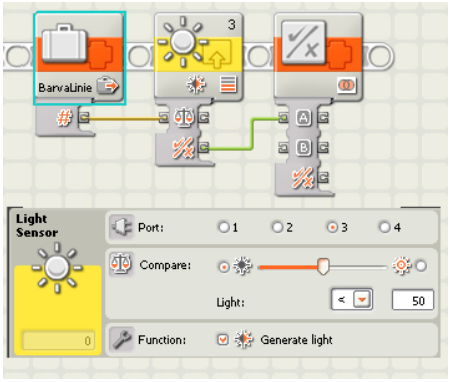


Vytvoření proměnné:

Položka menu Edit -> Define Variables



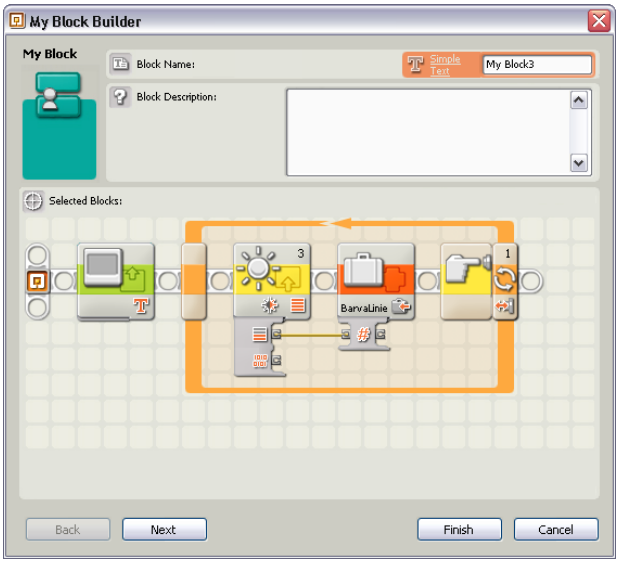
Porovnání proměnné s údajem čidla:



Pokud je údaj světelného senzoru menší než hodnota v proměnné BarvaLinie vrátí hodnotu pravda.

Vytvoření vlastního bloku:

Tažením myši označíme patřičné bloky a vybereme z hlavní nástrojové lišty tlačítko Create Block. Zde vyplníme jméno a popis bloku a pokračujeme tlačítkem Next, kde nastavíme ikonu bloku. Místo skupiny bloků pak vidíme pouze jeden s příslušným jménem.



V položce menu Edit -> MyBlock můžeme vytvořený blok dodatečně editovat.

4. Závěr

Zhodnoťte průběh vytváření programu a výsledky práce. Odůvodněte svůj algoritmus.

PŘÍLOHA P II: ZADÁNÍ LABORATORNÍ ÚLOHY 2

Automatický převozník nákladu

1. Zadání

Pomocí vývojového prostředí BricxCC vytvořte program pro robota NXT Tribot, který bude mít za úkol dojet pro kuličku na linii 1, naložit ji a převézt na úložiště na linii 2, pak naložit kuličku na linii 3 a převést na zbylé úložiště na linii 1.

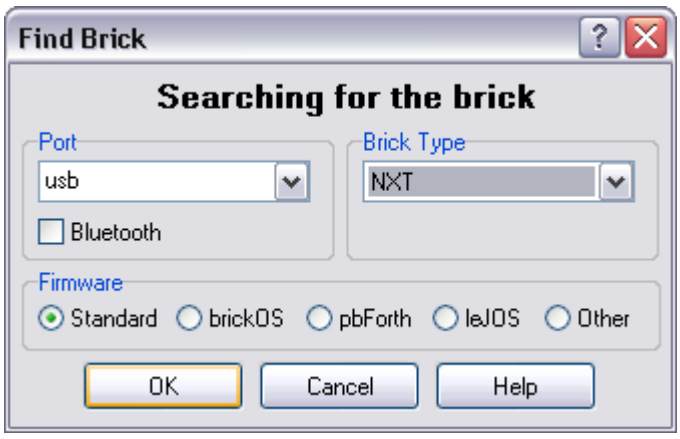
K programu vytvořte průvodní protokol.

2. Návrh vývojového diagramu

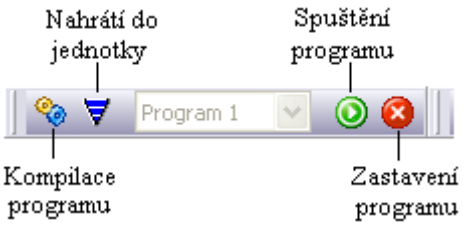
Sestavte vývojový diagram ovládacího programu (například v programu SmartDraw, Diagram Designer 1.20 nebo MSWord) a popište funkci jednotlivých bloků.

3. Vytvoření programu

1. K vytvoření programu použijte vývojové prostředí BricxCC. Programování je založeno na jazyku C.
2. Postup kompilace a stahování:
3. Pomocí USB kabelu připojte jednotku NXT k PC a zapněte ji.
4. Spusťte program BricxCC
5. Nastavte následující parametry:



6. Kompilace a stažení programu do řídící jednotky se provádí pomocí nástrojových tlačítek.



7. Pomocí USB kabelu stáhněte vytvořený program do řídící jednotky NXT a proveďte jeho odladění na testovací podložce se třemi liniemi.
8. Vypracujte protokol.

Příklady syntaxe:

#define tlacitko SENSOR_1 // definuje jméno proměnné jako senzor na portu 1

```
#define svetlo SENSOR_3 // definuje jméno proměnné jako senzor na portu 3
inline bool SledujCaru (int promena1, int promena2)
{return true;} // definice funkce se 2 vstupy, vrací pravdivostní hodnotu

task main
{
ResetTachoCount(OUT_A); // vynuluje sensor otáček
OnFwd(OUT_A,rychlost); // spustí motor na portu A vpřed definovanou rychlostí (0-100)
OnRev(OUT_B,rychlost); // spustí motor na portu B vzad definovanou rychlostí (0-100)
until (MotorTachoCount(OUT_A)>180); // čeká, až se motor otočí o vice než 180°
Off(OUT_AB); //Zabrzdí motory na portu A,B
SetSensorLight(IN_3); // Inicializuje světelný senzor na portu 3
SetSensorTouch(IN_1); // nastaví port 1 jako tlakové čidlo
tast = Sensor(IN_1); // čte hodnotu senzoru - 0 rozepnuto, 1 sepnuto
TextOut(30,LCD_LINE4,"Ahoj"); // Zobrazí na souř. x=30 na 4.řádku text Ahoj (displej má 8 řádků)
while(„podminka“) {} //Dělej dokud je splněna podmínka
do {} while(“podminka”);
repeat(“pocet opakovani”) {} //Opakuje akci daný počet krát
Wait(“hodnota”) ; //Uspí program na danou dobu – 1000 = 1 sekunda
until(“podminka”); // Čeká, dokud není splněná podmínka
if (“podminka”) {“neco”} else {“jineho”}
 } //hlavní fce program
```

4. Závěr

Zhodnocení výsledků.