

**LEGO MINDSTORMS Education jako nástroj
podpory základního technického vzdělávání**
LEGO MINDSTORMS Education as a Tool
to Support Basic Technical Education

Bc. Jaroslav Kotlán



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Jaroslav Kotlán**

Osobní číslo: **A13488**

Studijní program: **N3902 Inženýrská informatika**

Studijní obor: **Informační technologie**

Forma studia: **kombinovaná**

Téma práce: **LEGO MINDSTORMS Education jako nástroj podpory
základního technického vzdělávání**

Téma anglicky: **LEGO MINDSTORMS Education as a Tool for the Support of
Basic Technical Education**

Zásady pro vypracování:

1. Provedte literární rešerši tématu, zaměřte se na oblast projektové výuky.
2. Seznamte se s didaktickými možnostmi stavebnice LEGO Mindstorms a možnostmi jejího zařazení do výuky technického předmětu.
3. Navrhněte didaktický projekt zaměřený na modelování a řešení reálných situací.
4. Projekt s využitím stavebnice LEGO Mindstorms realizujte.
5. Vyhodnoťte silné a slabé stránky projektu z didaktického pohledu.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: tištěná/elektronická

Seznam odborné literatury:

1. W. H. KILPATRICK, The Project Method: The Use of the Purposeful Act in the Educative Process, University of California, Eleventh Impression March, 1929.
2. KRATOCHVÍLOVÁ, J., Teorie a Praxe projektové výuky. Brno: MU 2006, ISBN: 89-210-4142-0.
3. NOVÁK, D., Elektrotechnické stavebnice v technické výchově, Praha, 1997, ISBN: 80-86039-37-4.
4. SKALKOVÁ, J., Obecná didaktika. Praha: ISV, 1999, ISBN: 80-85866-33-1.
5. PAVELKOVÁ, I., Motivace žáků k učení: perspektivní orientace žáků a časový faktor v žákovské motivaci. Praha: Universita Karlova, 2002, ISBN: 80-729-0092-7.

Vedoucí diplomové práce:

doc. Mgr. Roman Jašek, Ph.D.

Ústav informatiky a umělé inteligence


Datum zadání diplomové práce:

6. února 2015

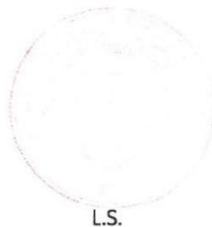
Termín odevzdání diplomové práce:

15. května 2015

Ve Zlíně dne 6. února 2015



doc. Mgr. Milan Adámek, Ph.D.
děkan



L.S.



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 29. 4. 2015

.....
podpis diplomanta

ABSTRAKT

Tato diplomová práce se zabývá vytvořením sady příkladů pro rozšířenou výuku programování na střední škole. Jako zajímavou didaktickou pomůcku jsem vybral laboratorního robota Lego Mindstorms NXT.

V teoretické části se budu zabývat možnostmi teoretické výuky a stručnému popisu základních stavebních prvků robota.

Na začátku praktické části stručně popíši základy jazyka RobotC a vývojového prostředí. Poté se zaměřím na řešení praktických úkolů. Příklady budou z prostředí Robot Virtual Worlds (RVW) a dále realizuji fotbal dvou robotů proti sobě.

Klíčová slova: Lego Mindstorms NXT, robot, fotbal robotů, projektová výuka

ABSTRACT

This Master's thesis deals with creating a set of examples for extended teaching of programming in high school. As an interesting teaching educational kit, I chose a Lego Mindstorms NXT robot.

The theoretical part will deal with the possibilities of theoretical instruction and a brief description of the basic elements of the robot.

I will briefly describe the basics of the language and development environment RobotC at the beginning of the practical part. Then, I focus on solving practical tasks. Examples are from the environment Robot Virtual Worlds (RVW) and I also realize implementation of two robots playing soccer against each other.

Keywords: Lego Mindstorms NXT, robot , soccerrobots, project education

Tímto bych chtěl poděkovat Doc. Mgr. Romanu Jaškovi, Ph.D. za ochotu, pomoc a cenné rady při tvorbě této práce. Děkuji také své rodině – manželce a rodičům za trpělivost a podporu, kterou mne provázeli po dobu mého studia. Déle bych chtěl poděkovat vedení školy za vstřícnost, kterou projevilo při uvolňování ze zaměstnání.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 MOŽNOSTI PROJEKTOVÉ VÝUKY	11
1.1 PROJEKTOVÁ VERSUS TEMATICKÁ VÝUKA	11
1.1.1 Projektová výuka.....	11
1.1.2 Tematická výuka	13
1.1.3 Srovnání projektové a tematické výuky	13
2 NÁVRH PROJEKTU FOTBAL ROBOTŮ	14
2.1 VOLBA PROJEKTU	14
2.2 PLÁNOVÁNÍ PROJEKTU	14
2.2.1 Možnosti RVW	15
2.2.2 Části RVW	16
2.2.2.1 Robots (roboti)	16
2.2.2.2 Movement (pohyb)	17
2.2.2.3 Sensing (snímání hodnot senzorů).....	17
2.2.2.4 Variables (proměnné)	17
2.2.2.5 Remotecontrol (dálkové ovládání).....	18
2.2.2.6 Utility (užitečné – nadstavbové úkoly).....	18
2.3 REALIZACE PROJEKTU	18
2.3.1 Cvičné úkoly v RVW (1. pololetí)	18
2.4 FOTBAL ROBOTŮ (2. POLOLETÍ)	19
2.5 ZAKONČENÍ PROJEKTU	20
3 POPIS STAVEBNICE LEGO MINDSTORMS NXT	21
3.1 ZÁKLADNÍ STAVEBNÍ PRVKY A SENZORY.....	22
3.1.1 Řídicí jednotka NXT (kostka).....	22
3.1.2 Základní senzory a servomotor	23
3.1.2.1 Dotykový senzor	23
3.1.2.2 Ultrazvukový senzor	24
3.1.2.3 Zvukový senzor.....	24
3.1.2.4 Světelný senzor	25
3.1.2.5 Barevný senzor	26
3.1.2.6 Servomotor.....	26
3.1.3 Rozšiřující senzory.....	27
3.1.3.1 Gyroskop.....	27
3.1.3.2 Kompas	27
3.1.3.3 Senzor vyhledávání IR signálu	28
3.1.3.4 IR Fotbalový míč	29
II PRAKTICKÁ ČÁST	30
4 POPIS JAZYKA A VÝVOJOVÉHO PROSTŘEDÍ ROBOTC.....	31
4.1 VÝVOJOVÉ PROSTŘEDÍ	31
4.2 SYNTAXE JAZYKA.....	34
5 SADY PŘÍKLADŮ V PROSTŘEDÍ RVW.....	36

5.1	LABYRINTHCHALANGE (MOVEMENT)	36
5.1.1	Zadání.....	36
5.1.2	Řešení.....	37
5.2	ROBO 500 1 (MOVEMENT)	39
5.2.1	Zadání.....	39
5.2.2	Řešení.....	39
5.3	ROBO SLALOM (MOVEMENT)	40
5.3.1	Zadání.....	40
5.3.2	Řešení.....	40
5.4	LINE RUNNER 1 (SENSING).....	41
5.4.1	Zadání.....	41
5.4.2	Řešení.....	41
5.5	FIREFLY BOT 1 (SENSING)	42
5.5.1	Zadání.....	42
5.5.2	Řešení.....	43
5.6	OBSTACLECOURSE (SENSING).....	44
5.6.1	Zadání.....	44
5.6.2	Řešení.....	44
5.7	AUTO ATTENDANCE (VARIABLES)	45
5.7.1	Zadání.....	45
5.7.2	Řešení.....	45
5.8	PIPEBOTLEVEL 1 (VARIABLES).....	46
5.8.1	Zadání.....	46
5.8.2	Řešení.....	47
5.9	MINE REMOVALCHALLENGE, SOCCERCHALANGE (REMOTECONTROL)	47
5.9.1	Zadání.....	47
5.9.2	Řešení.....	48
6	FOTBAL ROBOTŮ	51
6.1	STAVBA ROBOTA	51
6.2	KALIBRACE ROBOTA	53
6.3	JÍZDA ROBOTA	53
6.4	VYHLEDÁNÍ MÍČE	54
6.5	VYTVOŘENÍ BRANEK	56
6.6	FUNKCIONALITA BRANEK.....	56
6.7	STŘELBA NA BRÁNU	57
6.8	HRA ROBOTŮ PROTI SOBĚ	58
	ZÁVĚR	59
	CONCLUSION	60
	SEZNAM POUŽITÉ LITERATURY	61
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	62
	SEZNAM OBRÁZKŮ	63
	SEZNAM TABULEK.....	65
	SEZNAM PŘÍLOH.....	66

ÚVOD

Již velmi dlouhou dobu si lidé snaží usnadnit si práci využitím strojů. S rozvojem informačních technologií se naskytla možnost stoje nevyužívat pouze jako manuální sílu, ale také jim dát určitou inteligenci, aby mohly pracovat samostatně bez nutnosti obsluhy člověkem. Než se ale k takovému stroji propracujeme, je třeba ho navrhnout, sestavit, oživit a dobře otestovat. Robotika a automatizace je nelehký vědní obor, pomocí kterého ale tuto myšlenku můžeme realizovat.

V současné době pracuji jako učitel ICT na střední škole. Rozhodl jsem se proto, že vytvořím příklady pro skupinu středoškolských studentů. Nebudu tedy v práci popisovat teoretické principy robotiky a automatizace, ale zaměřím se na řešení praktických úkolů. Myslím si, že NXT roboti jsou velmi vhodným nástrojem, a to jak z hlediska didaktického, tak i motivačního. Pro co největšímu přiblížení řešení problémů v praxi využiji možností projektové výuky. Předpokládám, že studenti budou více vtaženi do děje v hodinách a mimo jiné budou k řešení úkolů přistupovat zodpovědněji.

V samotné práci rozeberu a popíšu, jak se dá aplikovat projektová výuka na daný problém. Provedu stručnou charakteristiku Lego Mindstorms NXT, popíši jazyk a prostředí, ve kterém budeme programovat, a seznámím s řešením praktických příkladů.

S výukou pomocí robotů Lego Mindstorms NXT začneme příští školní rok. Přepokládám, že vše, co v rámci této práce popíši a vytvořím, prakticky využiji v hodinách.

I. TEORETICKÁ ČÁST

1 MOŽNOSTI PROJEKTOVÉ VÝUKY

Smyslem projektové výuky je přiblížení možností řešení problémů reálným situacím. Studentům se nabízí možnost spolupracovat na řešení konkrétního problému. Mohou být u jeho návrhu, analýzy, realizace i testování. V určitých případech by se dalo řešení realizovat i v praxi. Jako příklad mohu uvést systém elektronické burzy učebnic (<http://burzaucebnic.vassboskovice.cz/>), který pod mým vedením vytvořili a do praxe zavedli sami studenti. Z povahy problematiky, kterou řeším v této práci, se s praktickou realizací počítat nedá. Příklady, které v rámci řešení práce vzniknou, využijeme ve výuce, případně při řešení úloh do různých soutěží. Určitým dlouhodobějším výhledem je rozšíření výuky robotiky na naší škole a spolupráce s firmami. Tam by propojení s praxí bylo logickým vyústěním spolupráce.

1.1 Projektová versus tematická výuka

Projektová a tematická výuka mají řadu společných rysů. Nemůžeme ale říci, že oba přístupy jsou totožné. V následující části oba přístupy stručně popíšu a srovnám je.

1.1.1 Projektová výuka

Projekt bychom si mohli definovat, jako komplexní úkol. Ten máme vyřešit. Učitel by měl navrhnout konkrétní problematiku, ale ne vždy je nutné striktní zadání. Při dostatečné zkušenosti pedagoga mohou studenti navrhnout svoje vlastní zadání úkolu, jsou lépe motivováni a přistupují k řešení úkolu svědomitěji. Je poté na učiteli, jak přizpůsobí řešení projektu požadavkům RVP a ŠVP. Z hlediska učitele je tento přístup složitější, ale na druhou stranu kreativnější a může všechny strany zúčastněné v projektu posunout.

Každý projekt musí mít svůj začátek, průběh a konec. Vše by mělo být jasně definováno a to jak časově, tak i z hlediska funkčních požadavků na jednotlivé části. Hlavním „vedoucím“ je v tomto případě učitel, který koordinuje činnost studentů. Z vlastní zkušenosti mohu říct, že je vhodné studenty rozdělit na skupiny. Každá skupina může řešit část problému nebo celý projekt samostatně. Myslím si, že vhodnější je druhý přístup. Studenti mohou porovnávat svoje výsledky, lépe se jejich práce koordinuje a v každé skupině můžeme rozdělit role podle schopností a zaměření konkrétního studenta. Je třeba ale kontrolovat, aby nepracoval pouze ten nejlepší a ostatní se jenom „vezli“. [1],[3], [5]

Plánování a koordinace výuky je samozřejmě na straně učitele. Není ale od věci studenty do rozhodování o životě projektu zapojit. Opět je třeba brát na zřetel nutnost probrat elementární látku, ale ta se dá při vhodných metodách zakomponovat do řešení jednotlivých dílčích úkolů. Co se týká výuky ICT, je zde nutná návaznost na předchozí látku (splněný menší úkol). Jak se definují a k čemu jsou proměnné, je třeba vědět stále. Zde narážíme na jeden ze základních problémů (nejen) projektové výuky. Co se studenty, kteří nezvládnou nebo zapomenou dříve probíranou látku? Budeme ostatní „brzdit“? Nebudeme brát ohled na ty, kteří předchozí učivo nezvládli? Ani jedna z možností není správná. Proto v našich plánech musíme počítat s možností určitého zpoždění prací oproti navrhovanému postupu. Vždy je lepší něco do projektu přidat, než zjistit, že se celý projekt nerealizuje ani zčásti. Dalším aspektem je skutečnost, že ani žádné dvě třídy studentů nejsou stejné. Mnohdy to platí i v rámci ročníku, který je rozdělený na skupiny. Je proto vhodné mít rozmyšleno více náhradních scénářů. Vždy se dá něco vylepšit, přidat, prostě udělat nadstavbu. Dnes je již zřejmé, že se nedají hodnotit všichni žáci zcela stejně. Ten lepší by měl dostat například obtížnější část. Je členem týmu a někdy je třeba vyvinout větší úsilí oproti ostatním. Naopak např. ne úplně zdatný programátor může perfektně zpracovat projektovou dokumentaci nebo zoptimalizovat řešení problému. [2], [3], [4]

Projekt by se měl také rozvrhnout správně časově. V rámci školního roku je jistě vhodné členění po měsících a po pololetích. Nesmíme zde zapomenout na různé volné dny, jako jsou vánoční a jarní prázdniny, maturity atp. Opět je lepší mít menší cíl s výhledem na možná rozšíření řešení.

Dalším aspektem při časovém návrhu jsou technické možnosti. Budeme počítat s faktem, že máme odpovídající PC a programové vybavení. Co ale další technické požadavky? Problematiku popíšu na Lego Mindstorms ve spojení s naší školou. V současné době máme deset kusů stavebnic použitelných do výuky. V příštím roce, kdy se výuka začne realizovat, budeme učit jednu skupinu po šestnácti studentech. V plánu je jeden robot na dva studenty, což je zatím dostačující počet. Ve školním roce 2016/2017 ale přibude další skupina. Studenti by se potom o roboty museli dělit, což by mohlo být problematické. Ne vždy bude třeba stejný robot pro dva různé ročníky. Vzhledem k tomu, že stavba robota trvá cca minimálně dvě hodiny, je možnost přestavby robota na každou hodinu nerealizovatelná. Dále se zde setkáváme s problémem využití robotů pro propagaci školy nebo pro rozšířený kroužek Lego robotů. Tuto situaci částečně řešíme využitím RVW v rámci programu Ro-

botC, kde se dají nacvičit základní možnosti ovládání robota ve virtuálním prostředí. Do budoucna budeme nuceni z důvodu udržení kvality výuky nakoupit další sady.

1.1.2 Tematická výuka

Tematická výuka se typicky využívá na základních školách, například v souvislosti s ročními obdobími. Na podzim se dá učit o tom, proč padá listí, učit se anglická slovíčka vztahující se k podzimu nebo provádět typické činnosti vztahující se k danému ročnímu období. Oproti projektové výuce je tento typ volnější. Na začátku nejsou striktně určeny cíle, počítá se větším zapojením studentů, ale téma volí pouze učitel. Z hlediska inženýrského přístupu k řešení problémů, který je podle mého názoru vhodný při řešení problémů v informatice, je jistě vhodnější použití projektové výuky. Proto již provedu pouze srovnání těchto dvou metod. Tematickou výuku jsem zde uvedl, protože je často s projektovým přístupem zaměňována. [3]

1.1.3 Srovnání projektové a tematické výuky

Tab. 1 Srovnání projektové a tematické výuky

	Projektová výuka	Tematická výuka
Definice projektu	Učitel a žáci	Učitel
Výstup	Jasně dané na začátku	Nejsou striktně dané, důležitější je osvojení si základních dovedností
Činnost studentů	Určená z úkolů definovaných v projektu	Vyplývá ze zadaných úkolů
Činnost učitele	Pomocník při řešení	Jasný řídicí prvek
Náročnost	Organizace, řízení	Příprava úkolů

2 NÁVRH PROJEKTU FOTBAL ROBOTŮ

2.1 Volba projektu

Hlavním úkolem projektu, který jsem si vybral pro demonstraci možností využití robotické stavebnice, je fotbalové utkání dvou robotů. Roboti budou hrát v aréně o rozměrech 192 x 86 cm. Rozměr herního pole není určen náhodně. Je to třetina plochy, kterou jsme využili při přípravě na soutěž, která se konala v listopadu na Fakultě elektrotechniky ČVUT. V rámci přípravy jsem vyrobil velké bludiště s překážkami (celková velikost 258 x 192 cm), které robot procházel v časovém limitu. Již při návrhu jsem vyhodnotil, že z hlediska manipulace a dalšího využití bude plocha příliš velká. Vyrobil jsem proto tři spojitelné části, které se dají použít dohromady nebo samostatně. Toho se dá do budoucna využít např. pro realizaci turnajů. Možností je také rozšíření plochy o více modulů. Kromě flexibility spatřuji jako velkou výhodu fakt, že herní aréna má mantinely, které zabrání vypadnutí robota a jeho poškození.

Roboti budou hrát samostatně bez dálkového ovládání. Možnost dálkového ovládání bude v rámci projektu řešena jako bonusová úloha. Při využití dálkového ovládání bude možná hra dvou robotů ovládaných člověkem proti sobě nebo hra robota proti člověku. Herní aréna bude rozdělena příčně na poloviny. Každý robot se bude pohybovat pouze na své polovině. Hra bude časově omezena. Důležitou součástí budou branky, které poznají vstřelený gól.

Základním problémem je fakt, jak robot pozná, kde se nachází a kde je míč. Pro to využiji kompas, speciální míč a infračervený senzor pro příjem signálu, který se využívá právě ve spojení s míčem.

Před začátkem vlastní hry se bude muset robot kalibrovat. Kalibrace se bude realizovat umístěním robota na značku a načtením hodnot z kompasu. Tím si robot zapamatuje startovací polohu, která bude zároveň sloužit jako záchytný bod pro nastavení robota do původní polohy.

2.2 Plánování projektu

Jak jsem se již zmínil v kapitole o tematické výuce, časový harmonogram řešení projektu bude korespondovat s plánem školního roku. Standardně se počítá při dvouhodinové tý-

denní dotaci s třiceti třemi odučenými dvouhodinovkami za školní rok. Vzhledem k tomu, že nemůžeme počítat s celou hodinovou dotací, projekt rozdělím na dvacet výukových jednotek. Dále bude dělení pokračovat na dvakrát deset jednotek na každé pololetí. [4]

2.2.1 Možnosti RVW

V prvním pololetí se zaměřím na procvičení základních dovedností v prostředí RVW.

RVW je podle mého názoru velice vhodný právě pro první kroky s Lego roboty. V rámci kroužku, který vedu, jsem postřehl, že zpočátku jsou studenti mírně rozčarovani, protože robot nedělá, co chtějí. Občas jim ujede, spadne nebo se někde zasekne. To samozřejmě může i v RVW, ale je to vykompenzováno rozmanitostí scénářů, které se dají procházet. Dalším velkým plusem je fakt, že se robot nerozbije. Čeho si ale na tomto prostředí cením nejvíce, je sada příkladů, které jsou připraveny i s pracovními listy. Je zde návod, jak jednotlivé úkoly splnit a úkolem studenta je poté vše naprogramovat. Jedná se tedy také o vhodný motivační nástroj, kdy se studenti učí jistou formou hry. Rád bych ještě zmínil možnost přihlásit se do RVW na lokální PC nebo na server. Po přihlášení se uživatel může podívat, které úkoly má již splněny, případně jestli již v daném úkolu došel k některému z kontrolních bodů. To můžeme vidět na obrázku 4 v části „Achievements“.



Obr. 1. Možnosti RVW

2.2.2 Části RVW

Při spuštění prostředí máme na výběr několik možností:

2.2.2.1 Robots (roboti)

Zde si můžeme vybrat z několika typů robotů. Jsou to tzv. Remboti nebo dva modely aut. Rembot je základní typ Lego robota. Na jeho sestavení lze snadno získat návod, proto bude pro naše účely vhodný. Auta se používají v další části RVW, a tou jsou speciální scénáře, které jsou např. ve formě tropického ostrova. Je to možnost opět zajímavá, ale určená spíše pro děti na základní škole. Remboti jsou dva, jeden má v přední části dotykový senzor (obrázek 5) a druhý robotickou ruku (obrázek 6). Každý se hodí na jiný typ úkolu. Mírnou nevýhodou je, že si nemůžeme naimportovat vlastní roboty. Ve firmě, která vývojové prostředí realizuje, na tom úkolu pracují, ale problematika je natolik komplikovaná, že tato možnost v dohledné době zřejmě nebude realizovatelná. Je třeba ale uvést, že při tak velkém počtu navržených úloh a pro naše účely, je stávající stav plně dostačující. Kdo by i tak vyčerpal všechny možnosti, může využít modulu „Levelbuilder“, kde si může vytvořit vlastní scénáře.



Obr. 2. Rembot s dotykovým senzorem



Obr. 3. Rembot s robotickou rukou

2.2.2.2 *Movement (pohyb)*

V této části se řeší základní úlohy zaměřené na ovládání robota. Využívá se hlavně časování, nastavení otáček motoru nebo přímá jízda robota. I při nastavení stejného výkonu motoru robot mírně zatáčí. Na začátku to pro mě bylo mírně matoucí. Proč v ideálním virtuálním prostředí taková chyba? Po studiu diskuzí a důkladném pročtení návodů jsem pochopil, že je to záměr. Ani v reálných podmínkách roboti nejezdí naprosto rovně. Kromě světelného senzoru, pomocí kterého kontrolujeme, jestli robot nepřešel nepovolenou oblast, proto dobře využijeme gyroskop.

2.2.2.3 *Sensing (snímání hodnot senzorů)*

V této části se studenti naučí kombinovat hodnoty z více senzorů a prohloubí si základní znalosti z předchozí kapitoly.

2.2.2.4 *Variables (proměnné)*

Jak již název oddílu napovídá, stěžejním tématem bude práce s proměnnými. Definice, volba správných datových typů nebo výpis na obrazovku virtuální kostky. Pomocí výpisu dat na kostku se programy velmi dobře ladí a velmi zajímavé je porovnání s realitou. Například při použití světelného senzoru velmi záleží na světelných podmínkách okolí, což se ve RVW neprojeví.

2.2.2.5 Remotecontrol (dálkové ovládání)

Dálkové ovládání není podle mého názoru nejzajímavější částí při ovládání robota, ale na druhou stranu ne nepodstatnou. Robota můžeme ovládat pomocí speciálního dálkového ovladače, pomocí druhé kostky nebo joystickem, který je připojený k PC. Ovládání kostkou nebo joystickem můžeme využít při fotbalu robotů.

2.2.2.6 Utility (užitečné – nastavbové úkoly)

Tyto úkoly jsou nastavbové a mohou sloužit lepším studentům na prohloubení znalostí nebo jako příprava na různé soutěže.

2.3 Realizace projektu

Projekt fotbal robotů budu realizovat ve dvou částech. První fáze bude přípravná s pomocí RVW. Druhá fáze je řešení hlavního úkolu. Na tomto místě uvedu pouze názvy úkolů, které se budou realizovat, případně je doplním krátkým popisem.

2.3.1 Cvičné úkoly v RVW (1. pololetí)

Názvy úkolů korespondují s názvy v prostředí RVW

- LabyrinthChalange (movement)
 - Jízda ze startovní pozice do cíle s vyhnutí se překážce (čára na podložce).
- Robo 500 1 (movement)
 - Objetí středu testovací plochy. Střed je opět realizovaný pomocí černé čáry na podložce.
- Robo slalom (movement)
 - Jízda ze startu do cíle okolo překážek. Směr objetí překážek je přesně daný, překážku pro úspěšné zvládnutí nelze objet z druhé strany.
- Line runner 1 (sensing)
 - Jízda ze startu přes čáry a zpět.
- Firefly bot 1 (sensing)
 - Vyhledání nejsvětlejšího bodu na podložce.
- Obstaclecourse (sensing)
 - Komplexní úkol na jízdu rovně, otáčení, dotykový a světelný senzor a senzor na měření vzdálenosti.

- Auto attendance (variables)
 - Počítání náhodně rozestavených předmětů při průjezdu herním plánem.
- Pipebotlevel 1
 - Počítání přejetých čar v tunelu.
- Mine removalchallenge (remotecontrol)
 - Sbírání míčků a přesun do cíle na čas.
- Soccerchalange
 - Přesun míče do prostoru branky.

2.4 Fotbal robotů (2. pololetí)

- Stavba robota
 - Návrh výsledné sestavy může být klíčový pro úspěšnou realizaci celého projektu. Je dobré pečlivě zvážit všechny možnosti, případně se inspirovat již realizovanými projekty. Do značné míry je to věc zkušeností. První návrhy většinou bývají předělávány, případně je nutné vytvořit vše od začátku.
- Kalibrace robota
 - Po umístění na základní pozici by si měl robot zapamatovat směr, kterým je soupeřova branka, a uložit některé vstupní hodnoty.
- Jízda robota
 - Realizují pohyb robota po své polovině.
- Vyhledání míče
 - Robot bude muset najít míč a dojet k němu.
- Vytvoření branek
 - Branky bude třeba dobře ukotvit a bude se muset zamezit hluchým místům, odkud by robot nemohl míč dostat.
- Funkcionalita branek
 - Branky budou vracet míč zpět do hry a počítat góly.
- Střelba na bránu
 - Robot najde míč, postaví se směrem k bráně a vystřelí.
- Hra robotů proti sobě 1
 - Bude otestováno, jak budou roboti zvládat soupeření a jak je vhodné stanovit, kdy má hra skončit.
- Hra robotů proti sobě 2

- Hru bude jistě třeba otestovat a vyladit případné chyby.
- Vyhodnocení projektu
 - Kromě samotného hodnocení bude dobré navrhnout vylepšení, na kterém by se dalo pracovat v budoucnu.

2.5 Zakončení projektu

Logickým vyústěním zakončení projektu bude samozřejmě hodnocení. Nebude závislé pouze na konečném výsledku, bude také třeba nemalou měrou započítat průběžnou činnost. Se studenty se společně zamyslíme nad průběhem a samotnou realizací projektu. Popíšeme silné a slabé stránky realizovaných úkolů a pokusíme se z toho vyvodit možnosti dalšího zlepšení. Je možné, že se projekt nepodaří úspěšně dokončit na 100%. Není to sice ideální stav, ale i to se v životě děje. Neodpustím si poznámku, že i větší odborníci, než je učitel a skupina studentů, něco nedokončí. My tím naštěstí nemůžeme zkomplikovat průjezd Prahou, ani utratit několik miliard navíc. Bude také jistě vhodné vzájemně se pokusit vyhodnotit roli každého jedince na projektu. Vyhodnocení by se nemělo podcenit. Při správném uchopení sebereflexe to může být pro studenty velmi prospěšné. Je třeba říct, co se nám podařilo, co již méně, a vše umět shrnout do smysluplného výstupu.

3 POPIS STAVEBNICE LEGO MINDSTORMS NXT

Základní sada lego Mindstorms NXT 9797 je stavebnice, která obsahuje základní moduly a díly k sestavení robota.



Obr. 4. Sada dílů 9797 [7]

Tuto sadu lze rozšířit např. Sadou technických dílů 9695.



Obr. 5. Sada technických dílů 9695 [7]

Při spojení těchto dvou sad a přikoupení potřebných senzorů můžeme pro potřeby výuky sestavit prakticky libovolnou sestavu. Pro větší projekty je možné kombinovat více sad mezi sebou.

3.1 Základní stavební prvky a senzory

3.1.1 Řídící jednotka NXT (kostka)

Je to mozek celého robota. Základem je 32-bitový ARM7 procesor s 256 KB flash pamětí na uložení programů a souborů a 64 KB RAM.

Pro možnost práce s kostkou máme LCD display o rozlišení 100 x 64 bodů.



Obr. 6. Kostka NXT [7]

Kostka má čtyři porty pro připojení senzorů, které jsou označeny čísly 1 až 4 a tři porty pro připojení motorů, označenými písmeny A, B, C. Pro připojení do počítače slouží USB konektor typu B. Napájení je umožněno pomocí vložené baterie nebo pomocí monočlánků typu AA. Vložená baterie je samozřejmě nabíjecí. Na kostce jsou dále čtyři ovládací tlačítka. Oranžové slouží pro zapnutí kostky a potvrzování nabídek v menu, tmavě šedé pod oranžovým jako krok zpět a vypnutí a dvě světle šedá tlačítka umožňují pohyb v menu. S kostkou můžeme komunikovat také pomocí technologie bluetooth. Tímto způsobem můžeme do kostky z PC nahrávat programy, použít dálkový ovladač nebo využít vzájemné komunikace kostek mezi sebou. [6] [7]

3.1.2 Základní senzory a servomotor

V sadě Lego Mindstorms NXT najdeme tři servomotory. Co se týká senzorů, v základní výbavě najdeme světelný, ultrazvukový, dotykový a zvukový senzor. V sadách, které máme na naší škole, ještě můžeme počítat se senzorem na vyhledávání IR signálu, kompasem a gyroskopickým a RGB světelným senzorem.



Obr. 7. způsob připojení motorů a senzorů [7]

3.1.2.1 Dotykový senzor

Dotykový senzor reaguje na stisknutí tlačítka, které je umístěno v přední části. Při stlačení vrací hodnotu 1, v základní poloze má hodnotu 0. Můžeme ho použít k detekci překážky přímým kontaktem nebo jako čítač počtu stisknutí. Můžeme ho rovněž využít jako vypínač pro (de)aktivaci nějaké činnosti. V projektu fotbal robotů je senzor zabudovaný v brance pro detekci vstřeleného gólu. Tento senzor je ve své podstatě velmi jednoduchý a oproti např. světelným sensorům dává jasné hodnoty, které nejsou závislé na okolním prostředí. [6] [7]



Obr. 8. Dotykový senzor [7]

3.1.2.2 *Ultrazvukový senzor*

Používá se jako „oči“ robota. Oči i vizuálně připomíná, princip činnosti je ale založen na ultrazvukových vlnách. Senzor ultrazvukové vlny vysílá a také je umí přijímat. Na základě doby, za kterou se vyslaná ultrazvuková vlna vrátí, se vypočítá vzdálenost nějaké překážky. Je to tedy stejný princip, jaký používají pro orientaci v prostoru netopýři.

Vzdálenost můžeme měřit v centimetrech nebo v palcích v rozmezí 0 – 255 cm. Měření není úplně přesné. Udává se ± 3 cm. V praxi ale bývá rozptyl větší, např. při pohybu a otáčení robota. Dalším problémem mohou být malé nebo zaoblené předměty. Při použití více ultrazvukových senzorů v jednom prostoru se také mohou navzájem rušit. Při pokusu o využití ve fotbalu robotů se tento senzor neukázal jako úplně vhodný. [6] [7]



Obr. 9. Ultrazvukový senzor [7]

3.1.2.3 *Zvukový senzor*

Zvukový senzor měří intenzitu zvuku v decibelech nebo vrací procentuální poměr hlasitosti zvuku. Můžeme si tedy vybrat ze dvou režimů

- dB – vrácená hodnota je v decibelech, senzor snímá i zvuky nad a pod prahem slyšitelnosti pro lidské ucho
- bBA – procentuální poměr hlasitosti. Jako 100% je stanovena hodnota slyšitelnosti lidským uchem, tedy cca 90 dB.

Problémem při jeho použití je to, že se zvuky mohou různě odrážet, někdo nečekaně promluví nebo bouchnou dveře, což může ovlivnit snímanou hodnotu. Využít by se dal například jako detektor při ostraze nějakého objektu. [6] [7]



Obr. 10. Zvukový senzor [7]

3.1.2.4 Světelný senzor

Světelný senzor umí rozpoznat intenzitu světla ve svém okolí. Může ho použít v pasivním režimu nebo v režimu aktivním. U aktivního režimu se u senzoru rozsvítí červená dioda. Tím se dá měřit intenzita odraženého světla. Základní, a také často využívaná možnost, je orientace v prostoru pomocí detekce (většinou černé) čáry. Robot se po ní může pohybovat nebo čára vymezuje prostor, ve kterém se může robot pohybovat. Druhé možnosti jsem využil u fotbalu robotů. [6] [7]



Obr. 11. Světelný senzor [7]

3.1.2.5 Barevný senzor

Barevný senzor je podobný senzoru světelnému. Umí rozeznat 17 různých barev. Může pracovat v několika režimech. Může mít zapnutou nebo vypnutou osvětlovací diodu. Dioda může svítit červenou, zelenou nebo modrou barvou. Potom senzor vrací různé hodnoty na stejných barvách. Je třeba proto vyzkoušet, jaký mód je pro daný úkol nejvhodnější. Tento senzor jsem použil pro detekci míče ve fotbalu robotů. [6] [7]



Obr. 12. Barevný senzor [7]

3.1.2.6 Servomotor

Servomotor slouží k rozpořhybování robota. Můžeme ho využít jako hnací jednotku pro realizaci pohybu v prostoru nebo při konstrukci speciálních částí, kde je třeba pohyb jednotlivých elementů. Jako příklad mohu uvést uchopení předmětů, natočení kol při použití hřebenového řízení nebo otáčení nějakého ramene. U servomotoru můžeme nastavit rychlost (výkon od -100 do 100) nebo počet otáček s přesností na 1°. Při vyšších rychlostech má motor určitý dojezd, s čímž musíme při programování počítat. [6] [7]



Obr. 13. Servomotor [7]

3.1.3 Rozšiřující senzory

V předchozí části jsem popsal senzory, které jsou standardní součástí základní stavebnice. Nyní popíšu další senzory, které jsem využil při řešení praktických úloh.

3.1.3.1 Gyroskop

Je to vlastně jednoosý (hodnota se detekuje pouze na jedné ze tří os) gyroskopický senzor, který vrací otočení ve stupních za jednu vteřinu. Může tedy sloužit ke zjištění nebo regulaci rychlosti otáčení. Gyroskop se umísťuje na robota ve vodorovné poloze, protože osa měření je potom ve svislé poloze. Gyroskopický senzor může nabývat hodnot $\pm 360^\circ$.

Tento senzor jsem využil v prostředí RVW, a to pro přímou jízdu robota. Využit by se dal i pro orientaci v prostoru.

Dalším zajímavým využitím v praxi je jeho aplikace ve dvoukolovém vozítku Segway. Čím rychleji se vozítko nakloní dopředu, tím rychleji zaberou motory proti pohybu otáčení. Popis problému je celkem jednoduchý, ale z vlastní zkušenosti mohu říct, že realizace tak jednoduchá není. Důležité je najít vhodný poměr síly motorů vůči naklánění vozítka. [6] [7]



Obr. 14. Gyroskopický senzor [7]

3.1.3.2 Kompas

Digitální kompas určuje azimut podle magnetického pole země. Kompas měří hodnoty 0° - 359° . Ze své podstaty slouží k orientaci robota v prostoru. Měření probíhá stokrát za vteřinu s přesností 1° . Toho jsem úspěšně využil ve fotbalu robotů.

Při použití kompasu je třeba si dát pozor na okolní vlivy. Samozřejmě přesnosti měření vadí magnety, ale také servomotory. Proto je vhodné umístit senzor co nejdále od nich. Při

prvotním testování jsem se setkal s problémem, kdy jsem zkoušel funkčnost na stole, který měl pod dekou kovovou konstrukci. Robot se občas začal nesmyslně točit. [6] [7]



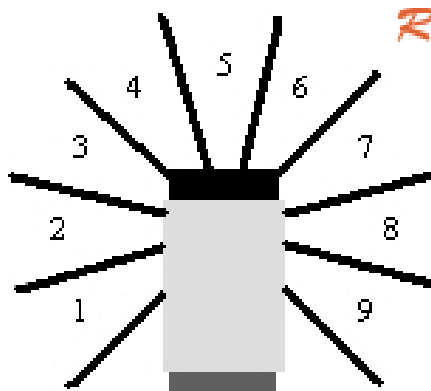
Obr. 15. Kompas [7]

3.1.3.3 Senzor vyhledávání IR signálu

Senzor slouží speciálně pro fotbal robotů. Umí přijímat infračervený signál, který vysílá speciální míč. Tím se dá zjistit směr, kterým má robot za míčem vyrazit. Senzor vrací hodnoty 1 – 9 v zorném poli 135° v závislosti na poloze míče (viz Obrázek 19). [6] [7]



Obr. 16. Senzor vyhledávání IR signálu [7]



Obr. 17. Hodnoty vrácené IR senzorem [7]

3.1.3.4 IR Fotbalový míč

Speciální míč, který vysílá infračervený signál, podle něj ho detekuje senzor pro vyhledávání IR signálu. Základem je 20 infračervených diod, které jsou rozmístěny tak, aby se míč dal vyhledat při jakémkoliv natočení míče. Míč může pracovat ve dvou základních módech, a to 600Hz a 1200Hz. Míč je napájen čtyřmi AAA bateriemi. [6] [7]



Obr. 18. Infračervený míč[7]

II. PRAKTICKÁ ČÁST

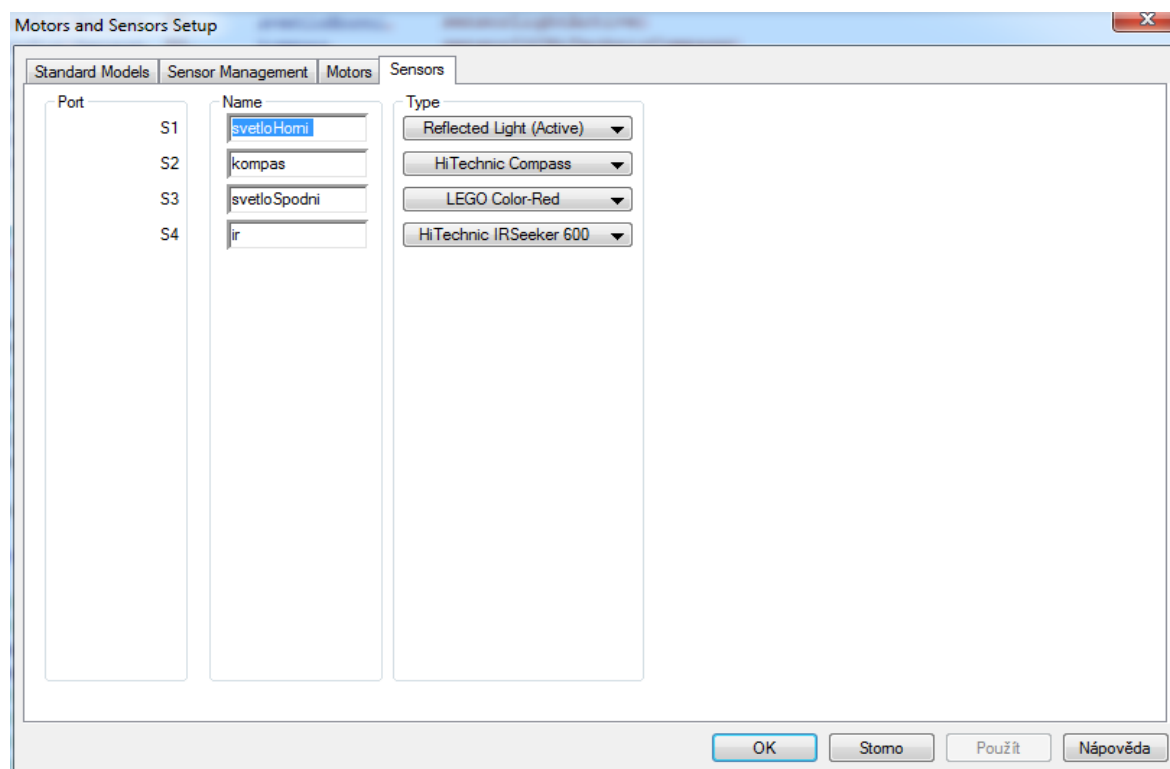
4 POPIS JAZYKA A VÝVOJOVÉHO PROSTŘEDÍ ROBOTC

Jako vývojové prostředí, a tím i programovací jazyk, jsem použil RobotC. Na tento program má naše škola zakoupenou licenci. Rozhodli jsme se pro něj pro jeho přívětivé uživatelské prostředí s velkou možností zobrazení různých informací o robotu. Co nás ale nadchlo nejvíce, bylo prostředí RVW. Nebudu se zabývat možnostmi a popisem dalších prostředí. Ne že bych je chtěl zavrhnout nebo je viděl jako méně dobré. Je to prostě proto, že pro výuku a další popis řešení příkladů to není třeba.

4.1 Vývojové prostředí

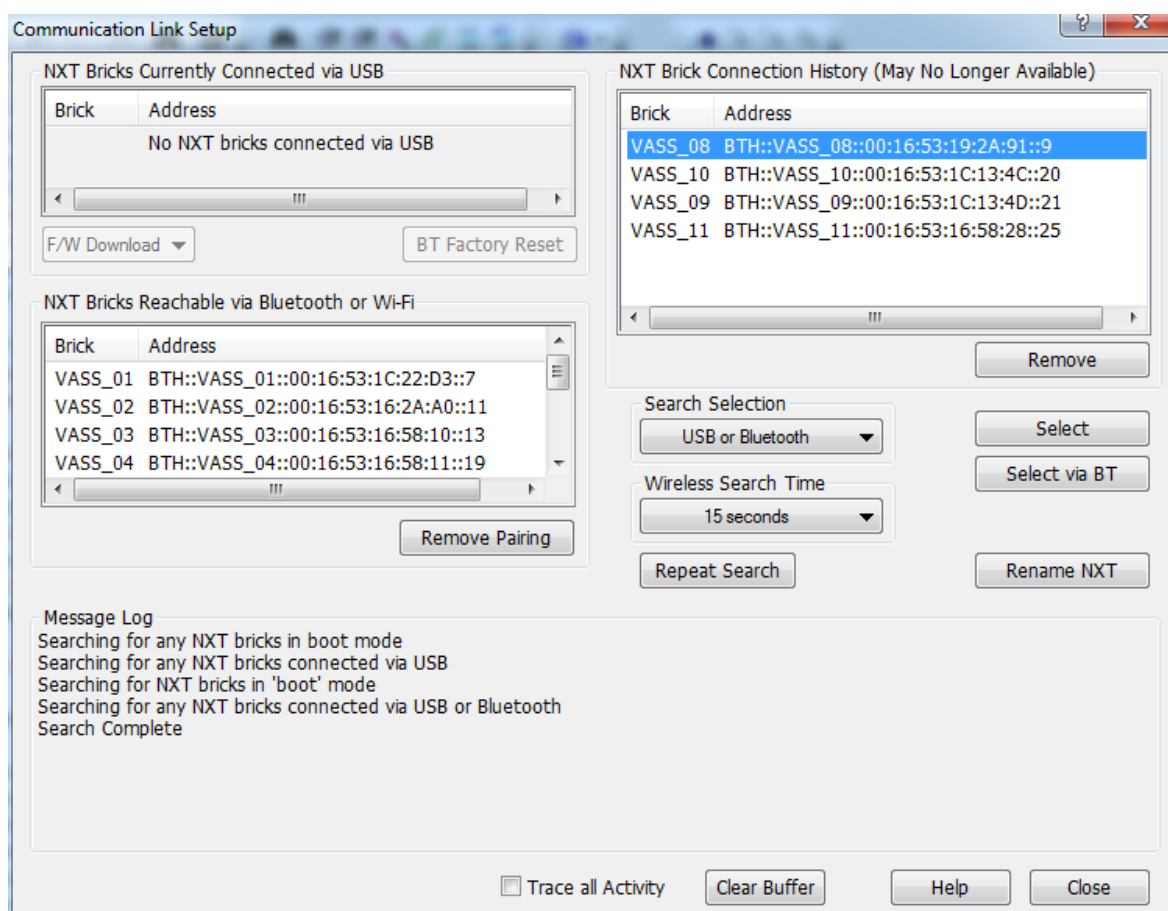
Okno vývojového prostředí má standardní rozhraní, proto nebudu popisovat jednotlivé nabídky a možnosti nastavení detailně. Zaměřím se pouze na specifické funkce, které lze využít při tvorbě a ladění programů.

Po vytvoření nového souboru je třeba udělat dvě základní věci. Nakonfigurovat senzory a vybrat prostředí, do kterého se program přeloží a nahraje. Konfiguraci senzorů vyvoláme z menu pomocí nabídky Robot | Motors and Sensors Setup. Konfiguraci lze provést také přímým zapsáním kódu na začátek programu. V prostředí RVW se dá využít možnosti přednastavených konfigurací v záložce Standard Models.



Obr. 19. Nastavení motorů a senzorů

Výběr prostředí se provede pomocí nabídky Robot | Compile Target. Pro nás mají význam nabídky Physical robot nebo Virtual Worlds. Dále je dobré provést kontrolu zvoleného typu jazyka v nabídce Robot | Platform Type. Vybereme možnost Lego Mindstorms NXT. Při připojení k fyzickému robotu je třeba vždy při prvním spuštění nahrát do kostky odpovídající firmware. Tuto možnost najdeme opět v menu v nabídce Robot | Download Firmware | Standard File. Připojení fyzické kostky se provádí pomocí nabídky Robot | NXT Brick W Communication Link Setup. Kostku můžeme připojit pomocí USB kabelu nebo Bluetooth. Při připojení pomocí USB můžeme kostku přejmenovat.



Obr. 20. Communication Link Setup

Pro ladění nebo zkoumání jednotlivých prvků robota využijeme okno, které se spustí z nabídky Robot | NXT Brick | Poll NXT Brick. Zde si můžeme otestovat nastavení jednotlivých komponentů.

NXT Sensor and Motor Display

Read Values from NXT

Motor	Spe...	PID	Mode	Regul...	Run St...	Tach User	Tach M...	Tach Limit	Tach Total

Sen... Type Mode Value Raw Ato-D

Sen...	Type	Mode	Value	Raw	Ato-D

Variable Value

Variable	Value
Sync Type	
Sync Turn	
Battery	
systemSl...	

Refresh Once
Poll Continuous
Poll Count
Less

Set Values into NXT

Motors

	Speed	Target Rot	Mode	Reg
A				<input type="checkbox"/>
B				<input type="checkbox"/>
C				<input type="checkbox"/>
D				<input type="checkbox"/>
E				<input type="checkbox"/>
F				<input type="checkbox"/>
G				<input type="checkbox"/>

Sensors

	Type	Mode
1		
2		
3		
4		

Encoder Reset
User Reset
Tacho Reset
Reset Devices

PID Factors

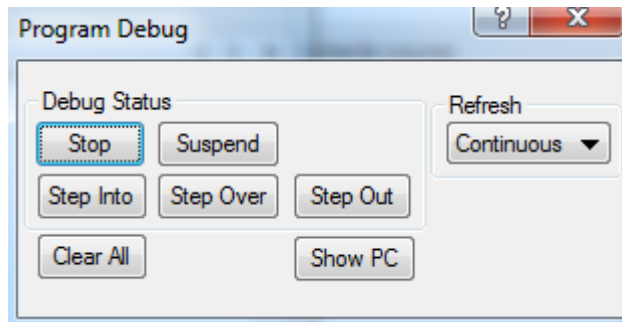
P	I	D

Factor Type
☒ Speed
☐ Encoder
☐ Trace Activity

Clear Buffer **Help** **Cancel**

Obr. 21. Poll NXT Brick

Pro kompilaci můžeme použít klávesu F5 (Robot | Compile and Download Program). Jak je zřejmé z anglického popisu, program se přeloží a nahraje do zařízení. To se dá poté ovládat z okna Program Debug. Mimo spouštění a zastavování programu zde máme možnost krokování.



Obr. 22. Nabídka Program Debug

4.2 Syntaxe jazyka

Jazyk RobotC je syntaxí podobný jazykům typu C, C++, PHP nebo JavaScript. Základní konstrukce, jako je zápis podmínky, cyklů nebo definice proměnných, je prakticky totožná a studenti již budou mít zvládnuté základy JavaScriptu. Nebudu zde proto popisovat základy jazyka jako takového, ale zaměřím se na speciální příkazy, které využijeme v následujících příkladech.

Ještě před samotným popisem příkazů uvedu definici senzorů a motorů.

```

1  #pragma config(Sensor, S1,      svetloHorni,      sensorLightActive)
2  #pragma config(Sensor, S2,      kompas,          sensorI2CHiTechnicCompass)
3  #pragma config(Sensor, S3,      svetloSpodni,    sensorCOLORRED)
4  #pragma config(Sensor, S4,      ir,              sensorHiTechnicIRSeeker600)
5  #pragma config(Motor,  motorA,    strela,          tmotorNXT, PIDControl, encoder)
6  #pragma config(Motor,  motorB,    levy,            tmotorNXT, PIDControl, encoder)
7  #pragma config(Motor,  motorC,    pravy,           tmotorNXT, PIDControl, encoder)

```

Obr. 23. Definice senzorů a motorů

U definice senzorů (řádky 1-4) se zadávají jako parametry typ zařízení (Sensor), port, na který je senzor připojen (S1), název senzoru (svetloHorni) a typ senzoru (sensorLightActive). U motorů jsou první tři parametry stejné, u dalších se nastavuje typ motoru a směr otáčení.

Na začátku programu můžeme standardním způsobem nahrát data z dalších souborů, např. funkce ovladačů motorů a senzorů.

```

10 #include "drivers/hitechnic-colour-v2.h"
11 #include "drivers/hitechnic-compass.h"

```

Obr. 24. Načtení dat z externích souborů

Tab. 2 Vybrané příkazy jazyka RobotC

Příkaz	Popis	Příklad
motor[nazevMotoru]	Nastavení výkonu motoru	motor[motorA] = 100; Nastavení výkonu motoru na hodnotu 100.
bMotorReflected[nM]	Otočení motoru o 180°	bMotorReflected[mB]=true; motor se pootočí o 180°.
nMotorEncoder[nM];	Čítač otočení motoru.	nMotorEncoder[mC]=0; hodnota natočení motoru se nastaví na nulu.
nxtDisplayTextLine (nLineNumber, sFormatString, parm1,parm2, parm3)	Výpis textu na kostku NXT.	nxtDisplayTextLine(1,"%d", a); Na první řádek kostky se vypíše hodnota proměnné a.
SensorValue [nazevSenzoru]	Práce s hodnotou daného senzoru.	a=SensorValue[s10]; Do proměnné a se uloží hodnota ze senzoru s1.
wait1Msec(casVms)	Program čeká po dobu, která se zadává v milisekundách.	wait1Msec(200); Program počká 0,2 vteřiny.
TaskNazevVlakna()	Definice vlákna.	taskMereni() { //zápis kódu; }
StartTask(nazevVl)	Spuštění vlákna v hlavním programu.	StartTask(Mereni); Spuštění vlákna Mereni().
voidnazevProcedury()	Definice procedury nebo funkce	voidVystrel(){ //zápis kódu; }

5 SADY PŘÍKLADŮ V PROSTŘEDÍ RVW

Jak jsem již v práci popsal, prostředí RVW slouží jako cvičné hřiště, kde si studenti mohou osvojit základní dovednosti s Lego roboty. Jednou z mála nevýhod je nutnost zakoupení placené licence. Při hromadném nákupu se cena dá stlačit pod 500,- Kč, což mi nepřipadá jako velká částka. A investice do vzdělání se nemůže nikdy ztratit. Názvy podkapitol jsou shodné s názvy řešených úkolů. V závorce je vždy část, ve které se daný úkol nachází.

V kódu používám komentáře. Části kódů, které se vyskytují vícekrát (např. i v dalších příkladech), již komentované nejsou. Chci tím docílit toho, aby si student musel případně nastudovat předchozí příklad, který by při studiu přeskočil. Získá tím ucelený přehled o dané problematice.

5.1 Labyrinth Chalange (movement)

5.1.1 Zadání

Úkolem je projet podle plánu do cíle. Startovní poloha je v červeném čtverci, robot se musí dostat do černě vyšrafovaného pole. Na obrázku 21 jsou vidět žluté hvězdy, které znamenají splnění úkolu.

MOVEMENT

- ★ Labyrinth Challenge
- ★ Robo 500 1
- ★ Robo Slalom
- Line Painter Challenge
- Sumo Bot

Labyrinth Challenge

The robot must first begin at the starting point, and get to the goal area by completing turning and forward movement behaviors. Completion of this challenge is required for the Movement Mastery badge.

Specification Document

Achievements

- ★ Moving Forward
- ★ Turn Left
- ★ Turn Right
- ★ Labyrinth Completion

Current Robot: REMBot (w. Touch)

Fixed starting point:

START ACTIVITY ►

Obr. 25. Zadání úkolu Labyrinth Challenge

5.1.2 Řešení

Algoritmické řešení je velmi jednoduché a slouží k základnímu seznámení se s problematikou. Robot pouze jezdí rovně a otáčí se. Otáčení je realizováno pomocí gyroskopického senzoru, ale bylo by možné použít i časování.

Při řešení úkolu je vhodné zobrazovat si důležité funkce na display kostky. Velmi jednoduše si pro tento účel můžeme vytvořit samostatné vlákno. Tuto funkcionalitu budu používat i v dalších příkladech.

```
11 //vlakno pro kontrolu hodnot gyroskopu a otacek motoru na kostce
12 task Mereni()
13 {
14     while(true)
15     {
16         nxtDisplayCenteredTextLine(1, "%d", HTGYROreadRot(gyro));
17         nxtDisplayCenteredTextLine(2, "%d", nMotorEncoder[motorB]);
18     }
19 }
```

Obr. 26. Funkce pro výpis hodnot na kostku

Kostku si můžeme zobrazit virtuálně.



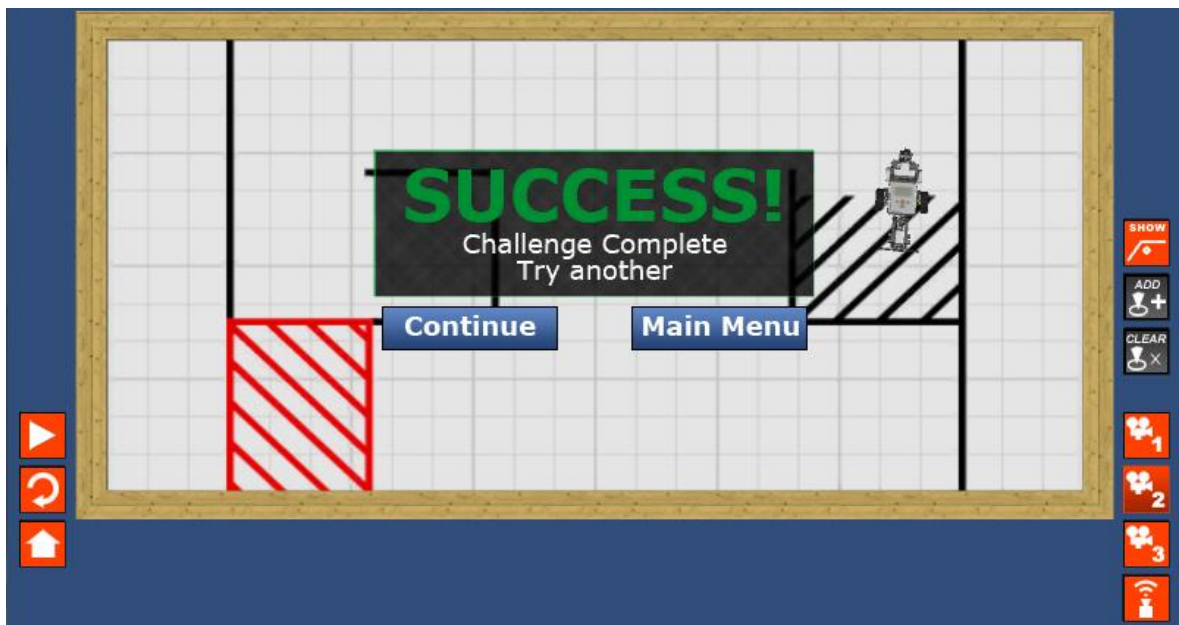
Obr. 27. Hodnoty na virtuální kostce NXT

Další základní funkcí je otočení robota. Využil jsem k tomu gyroskopický senzor.

```
21 void otoc(int stupen)
22 {
23     //kalibrace gyroskopu
24     HTGYROstartCal(gyro);
25     //kdyz je hodnota otoceni mensi nez 0, otaci se robot na jednu stranu
26     //do dosazeni zvoleneho stupne otoceni
27     if(stupen<0)
28         while(HTGYROreadRot(gyro)>stupen)
29         {
30             motor[motorB]=50;
31             motor[motorC]=-50;
32         }
33     else
34         //kdyz je hodnota otoceni vetsi nez 0, otaci se robot na druhou stranu
35         //do dosazeni zvoleneho stupne otoceni
36         while(HTGYROreadRot(gyro)<stupen)
37         {
38             motor[motorB]=-50;
39             motor[motorC]=50;
40         }
41     //motory je vzdy nutne zastavit, aby se robot neotcel stle
42     motor[motorB]=0;
43     motor[motorC]=0;
44 }
```

Obr. 28. Funkce pro otočení robota

Nakonec si ještě můžeme ukázat, jakým způsobem se studenti dozvědí o úspěšném splnění úkolu.

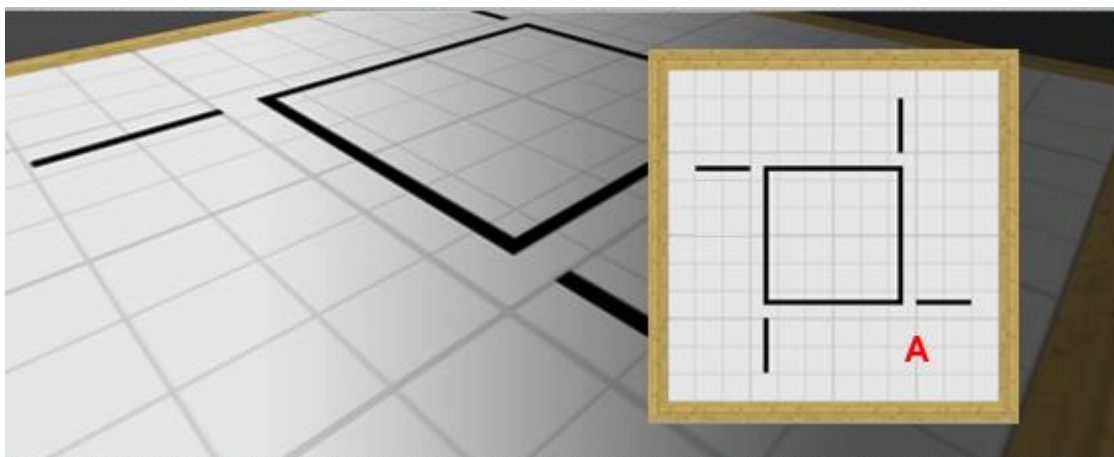


Obr. 29. Úspěšné splnění úkolu

5.2 Robo 500 1 (movement)

5.2.1 Zadání

Robot musí objet území, které je vymezeno černou čarou, a to dvakrát. Té se nesmí dotknout, naopak musí přejet všechny černé čáry, které jsou kolmo ke směru jízdy na konci každé strany.



Obr. 30. Zadání úlohy Robo 500 1

5.2.2 Řešení

Při řešení tohoto úkolu jsem musel zajistit jízdu robota v přímém směru. Využil jsem k tomu opět gyroskopický senzor. Po cca dvaceti měřeních odchylek při jízdě bez použití gyroskopu jsem dospěl k vhodnému poměru rychlosti motorů a odchylky robota od přímého směru. Jedním z faktorů, které mohou ovlivnit jízdu robota ve virtuálním prostředí, je reálné zatížení PC. Když jsem např. pustil záznam obrazovky na video, chování robota bylo ovlivněno. Program RVW prostě nestíhal. Je to třeba brát na zřetel a mít PC v dobrém softwarovém stavu a nemít na něm ve stejnou chvíli spuštěné další, výpočetně náročné aplikace.

```

45 //funkce pro jizdu primym smerem
46 void JizdaRovne(int v)
47 {
48     //vyrovnani jizdy do primeho smeru
49     motor[right]=v+2*SensorValue[gyro];
50     motor[left]=v-2*SensorValue[gyro];
51 }
52 task main()
53 {
54     StartTask(Mereni);
55     //kalibrace gyroskopu
56     HTGYROstartCal(gyro);
57     //po prejeti cary se robot otoci
58     //dokud se neotoci 8x
59     while((otacky++)<9)
60     {
61         HTGYROstartCal(gyro);
62         SensorValue[gyro]=0;
63         //casova prodleva je zde kvuli spravne kalibraci gyroskopu
64         wait1Msec(500);
65         JizdaRovne(90);
66         wait1Msec(2800);
67         Otoc(840);
68     }
69 }

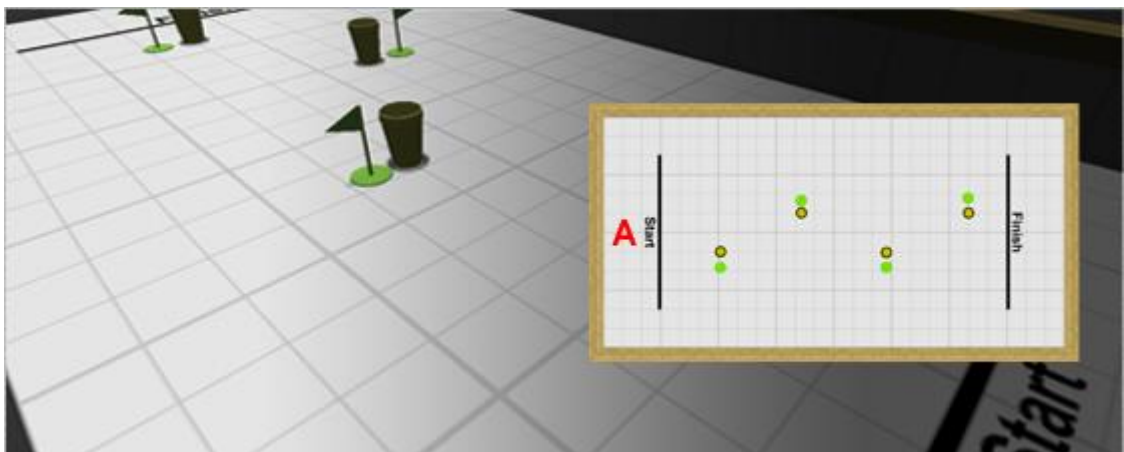
```

Obr. 31. Funkce pro jízdu rovně a hlavní část programu

5.3 Robo slalom (movement)

5.3.1 Zadání

Robot projíždí slalom, při průjezdu se nesmí dotknout překážky, shodit ji nebo se k ní přiblížit.



Obr. 32. Zadání příkladu Robo Slalom

5.3.2 Řešení

V příkladu je použita funkce pro přímý směr, otočení a časování, která jsou využita v předchozích dvou příkladech.

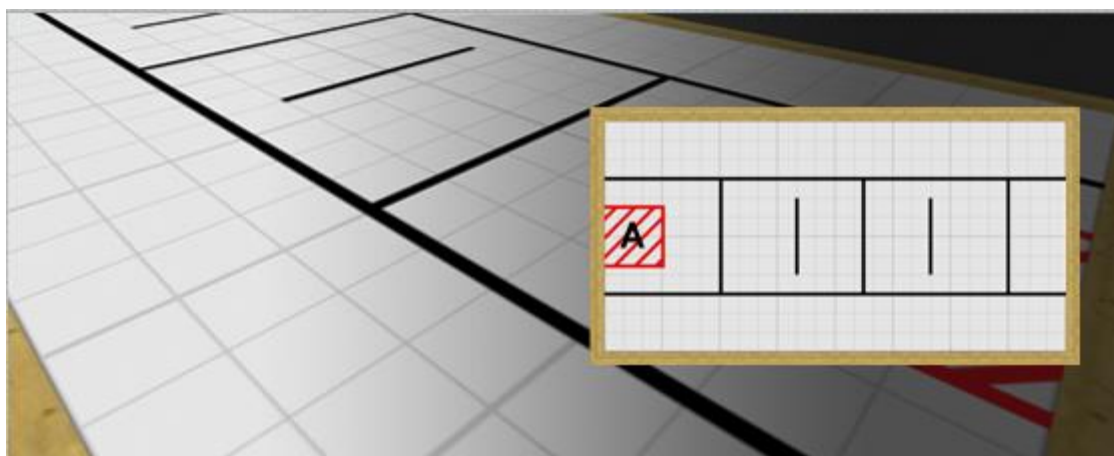
```
37 task main()
38 {
39     //promenna uhel není absolutní počet stupňů,
40     //ale počet stupňů za vteřinu
41     int uhel=440;
42     //stanovení základní časové jednotky
43     int cas=650;
44     //nastavení základní rychlosti
45     int v=90;
46     Otoc(uhel);
47     JizdaRovne(v);
48     wait1Msec(2*cas);
49     Otoc(-uhel);
50     JizdaRovne(v);
51     wait1Msec(cas/4);
52     Otoc(-uhel);
53     JizdaRovne(v);
```

Obr. 33. Část kódu robo slalom

5.4 Line runner 1 (sensing)

5.4.1 Zadání

Tímto příkladem se dostáváme do řešení úloh, které jsou zaměřené na práci se senzory. Robot má před sebou pět čar. Ze startu robot dojde k první čáře, na ní se otočí o 180° , poté se vrátí na start, otočí se opět o 180° , první čáru přejede, dojde ke druhé, otočí se atd., než se po přejetí všech čar vrátí na začátek. Robot nesmí vyjet mimo dráhu, která je vyznačena opět černou čarou. Kromě gyroskopického senzoru využijeme také senzor světelný.



Obr. 34. Zadání příkladu Line Runner 1

5.4.2 Řešení

Opět jsem zde využil funkce pro jízdu přímým směrem a otočení robota. Stěžejním problémem bylo přejetí čar, u kterých se již robot otočil, a zjištění, u kterých již byl, aby je

naopak nepřejel. Udělal jsem si výčet stavů podle počtu přejetí, tím se vyřešilo, kdy se má robot otočit a kdy má čáru přejet. Přejetí čáry je vyřešeno tak, že když robot čáru najde a má ji přejet, jede tak dlouho, dokud není hodnota senzoru pro světlo menší než hodnota naměřená na podložce.

```
29 //funce pro prejeti cary, kter byla jiz navstivena
30 void PrejedCaru(int vp)
31 {
32     //dokud nebude robot mimo caru, pojede rovne
33     while(SensorValue[light]<60)
34     {
35         JizdaRovne(vp);
36     }
37 }
```

Obr. 35. Funkce pro přejetí čáry

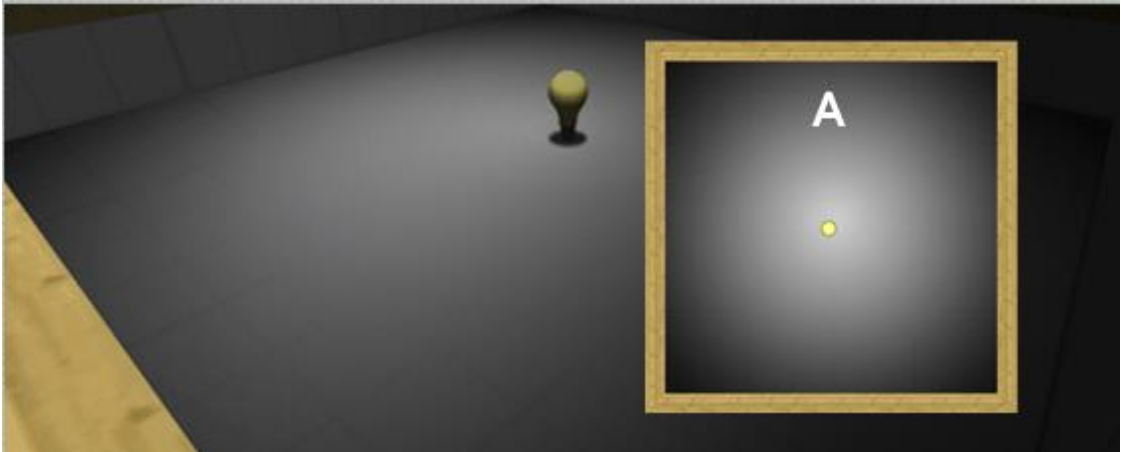
```
58 while(prejeto<=35)
59 {
60     while(SensorValue[light]>50)
61     {
62         JizdaRovne(v1);
63     }
64     //kontrola, kolik car jsme jiz prejeli
65     //kdy je jiz prejeta, robot se otoci
66     if(++prejeto==1 || prejeto==2 || prejeto == 4 || prejeto == 6 || prejeto == 9 ||
67     {
68         Otoc();
69     }
70     else
71         PrejedCaru(v1);
72 }
```

Obr. 36. Kontrola přejetí čáry

5.5 Firefly bot 1 (sensing)

5.5.1 Zadání

Robot je umístěn na kraji plochy. V jejím středu je žárovka a robot musí přijet co nejbližší k ní. Využívá se zde světelný senzor. Je třeba také kontrolovat, jestli robot nenarazil na kraj plochy. K tomu se využije dotykový senzor.



Obr. 37. zadání příkladu Firefly Bot 1

5.5.2 Řešení

Robot se na začátku natočí k žárovce. Poté vždy kousek popojede. Změří hodnoty na světelném senzoru před jízdou a po jízdě. Podle rozdílu naměřených hodnot se otočí k žárovce. Vše opakuje, dokud k žárovce nedojede.

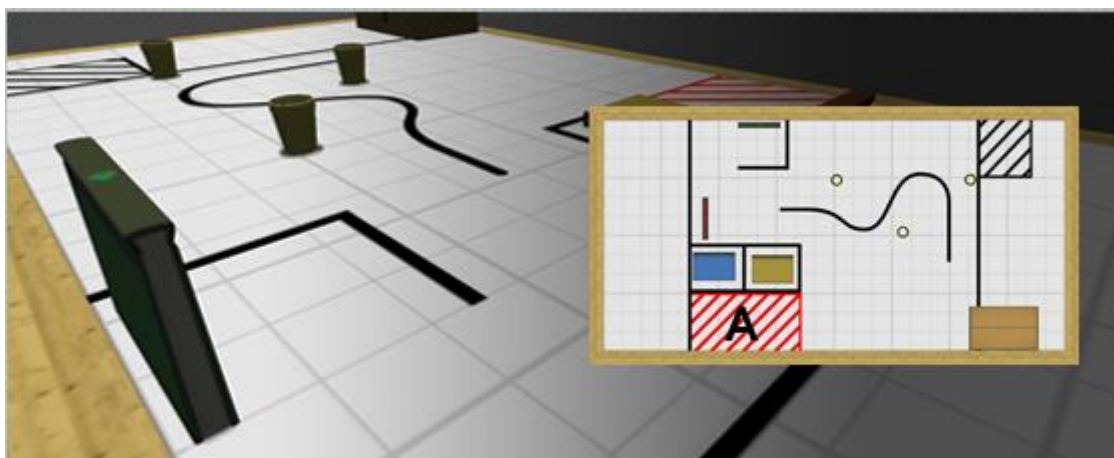
```
39 while(naraz==0 && maxLight<76)
40 {
41     //nacte se intenzita svetla
42     lightValuePred=SensorValue[light];
43     //robot popojede
44     motor[left]=50;
45     motor[right]=50;
46     wait1Msec(500);
47     //nacte se intenzita svetla po jizde
48     lightValuePo=SensorValue[light];
49     //vypocita se rozdil mezi puvodni a dalsi namerenou hodnotou
50     rozdil=lightValuePred-lightValuePo;
51     //podle rozdilu se robot natoci
52     if(rozdil>0)
53     {
54         motor[left]+=25*rozdil;
55         motor[right]-=25*rozdil;
56         wait1Msec(1000);
57     }
58 }
```

Obr. 38. Řešení příklad Firefly Bot 1

5.6 Obstaclecourse (sensing)

5.6.1 Zadání

Poslední příklad ze sekce sensing je komplexní a využijí se v něm všechny dovednosti a znalosti z předchozích příkladů. Robot musí pomocí dotykového senzoru najít první překážku. Potom najet na černou čáru, po které pojede až na její konec. Zde se využije světelný senzor. Poté robot musí pomocí senzoru na měření vzdálenosti najít další překážku. Poté, bez sražení dalších překážek, dojede na konec trasy. Při jízdě rovně se využívá gyro-skopický senzor. Z předchozích příkladů se využije funkce pro otočení a jízdu rovně.



Obr. 39. Zadání příkladu Obstacle Course

5.6.2 Řešení

Jednou ze základních úloh je jízda robota po čáře. Využil jsem zde algoritmu, kdy robot jede na jednu stranu, když je na čáře (hodnota senzoru je menší než v okolí čáry) nebo jede na stranu druhou, když je hodnota větší. Robot jede po hraně čáry mírně trhavým pohybem.

```
60 //jízda po care
61 //dokud se robot nedostane zpět na podložku
62 while(SensorValue(light)<60)
63     //kdyz je robot na care
64     if(SensorValue(light)<40)
65     {
66         motor(motorB)=90;
67         motor(motorC)=5;
68     }
69     //kdyz robot caru opusti
70     else
71     {
72         motor(motorB)=5;
73         motor(motorC)=90;
74     }
```

Obr. 40. Jízda robota po čáře

Dalším novým prvkem, který se v tomto příkladu objevuje, je využití senzoru na měření vzdálenosti. Zde je třeba věnovat pozornost faktu, že senzor měří hodnoty do 255 cm. Když je vzdálenost větší, v prostředí RVW vrací hodnotu -1, v reálné prostředí 255.

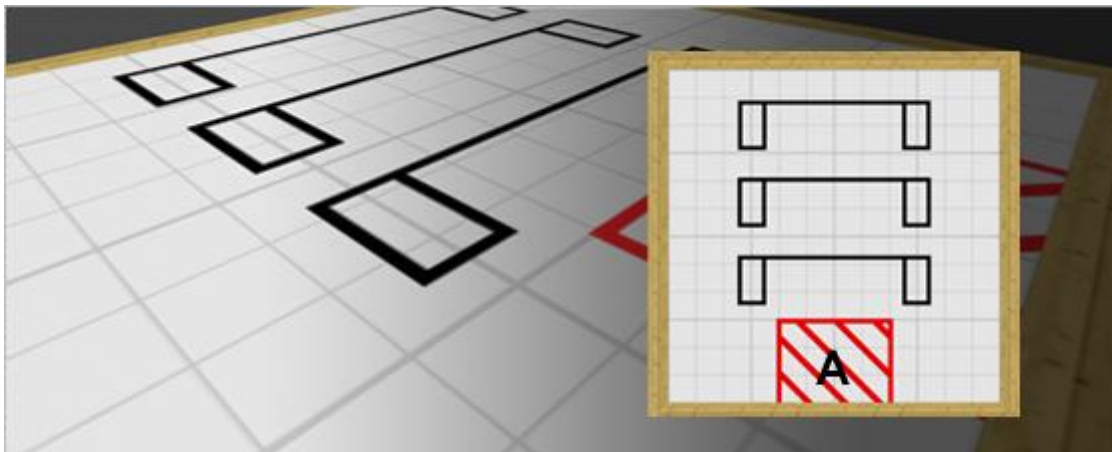
```
81 //dokud je vzdálenost mensi nez 30cm  
82 while(SensorValue[sonar]>=30)  
83     JizdaRovne(90);
```

Obr. 41. Využití ultrazvukového senzoru

5.7 Auto attendance (variables)

5.7.1 Zadání

Robot se pohybuje ve virtuální třídě a počítá studenty v ní. Lavice (celkem tři) jsou reprezentovány černou čarou a studenti knihami. Na začátku se náhodně vygeneruje, jestli je student na svém místě. Jestliže ano, umístí se zde kniha. Robot je mezi lavicemi, u každé se zastaví (využití světelného senzoru), rozhlédne se a pomocí ultrazvukového senzoru zjistí, jestli je student (např. Hliník) přítomen. Poté se robot přesune k další lavici. Počítají se přítomní studenti. Aktuální stav studentů se průběžně vypisuje na kostku.



Obr. 42. Zadání příkladu autoattendance

5.7.2 Řešení

Při řešení opět využijí funkce pro jízdu v přímém směru, otočení a přejetí čáry.

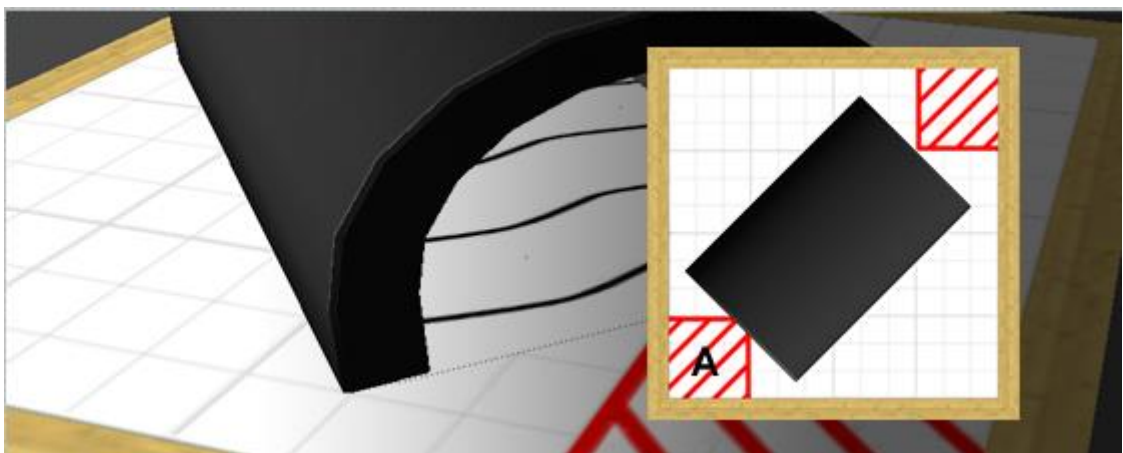
```
64 task main()
65 {
66     StartTask(Mereni);
67     int v1;
68     v1=90;
69     JizdaRovne(v1);
70     wait1Msec(500);
71     //dokud robot neprejde vsechny cary
72     while((prejeto++)<=2)
73     {
74         //kdyz je robot mezi lavicei
75         while(SensorValue[light]>50)
76         {
77             JizdaRovne(v1);
78         }
79         //rozpoznavani studenta
80         Pocitej();
81         PrejedCaru(v1);
82     }
83     JizdaRovne(v1);
84     wait1Msec(500);
85     motor[motorB]=0;
86     motor[motorC]=0;
87 }
```

Obr. 43. Hlavní část programu Auto attendance

5.8 Pipebotlevel 1 (variables)

5.8.1 Zadání

Robot projíždí tunelem, ve kterém jsou náhodně generovány černé čáry, které jsou umístěny kolmo ke směru pohybu robota. Pro kontrolu se počet čar objeví v levém horním rohu virtuálního prostředí. Robot má za úkol spočítat, kolik čar v tunelu je. Počet čar se průběžně objevuje na kostce NXT.



Obr. 44. Zadání příkladu Pipebot Level 1

5.8.2 Řešení

V příkladu se opět sumarizují funkce a poznatky z předchozích úkolů. Využijí funkce pro jízdu přímým směrem a přejíždění čáry. V potaz se musí brát jiné světelné podmínky v tunelu.

```
36 task main()
37 {
38     StartTask(Mereni);
39     int v;
40     v=90;
41     JizdaRovne(v);
42     wait1Msec(1000);
43     motor[right]=0;
44     motor[left]=0;
45     wait1Msec(1000);
46     //dokud neni robot na konci
47     while(SensorValue[light]>80)
48     {
49         if(SensorValue[light]>=90)
50             JizdaRovne(v);
51         else
52         {
53             motor[right]=0;
54             motor[left]=0;
55             ++prejeto;
56             PrejedCaru(v);
57         }
58     }
59 }
```

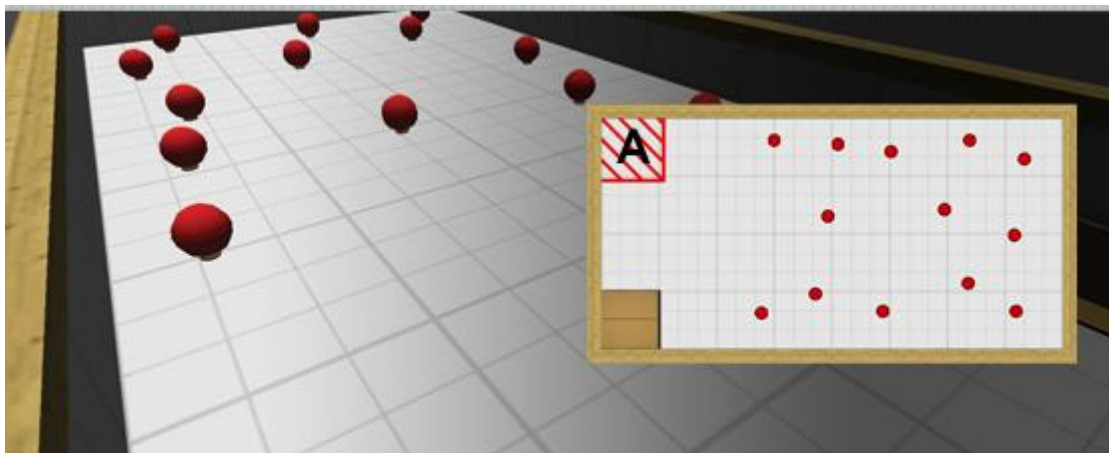
Obr. 45. Řešení úkolu PipebotLevel 1

5.9 Mine removalchallenge, Soccerchalange (remotecontrol)

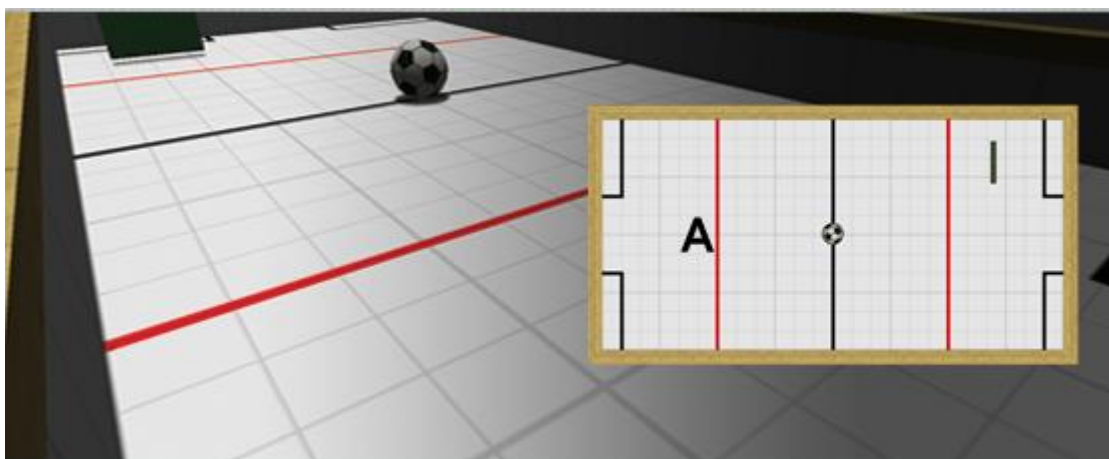
5.9.1 Zadání

Poslední dva úkoly jsem spojil do jednoho. Robot se ovládá joystickem, proto programová část není nijak náročná a pro oba příklady můžeme použít stejný kód. Na rozdíl od předchozích příkladů použijí robota s mechanickou rukou (viz. Obr. 6). Problém může nastat při nastavení joysticku.

V příkladu MinMine removalchallenge má obsluha robota posbírat míčky a dopravit je do cílového kontejneru. V úloze sockerchalange je úkolem dopravit míč do branky. Kromě statických jsou zde také pohybující se překážky.



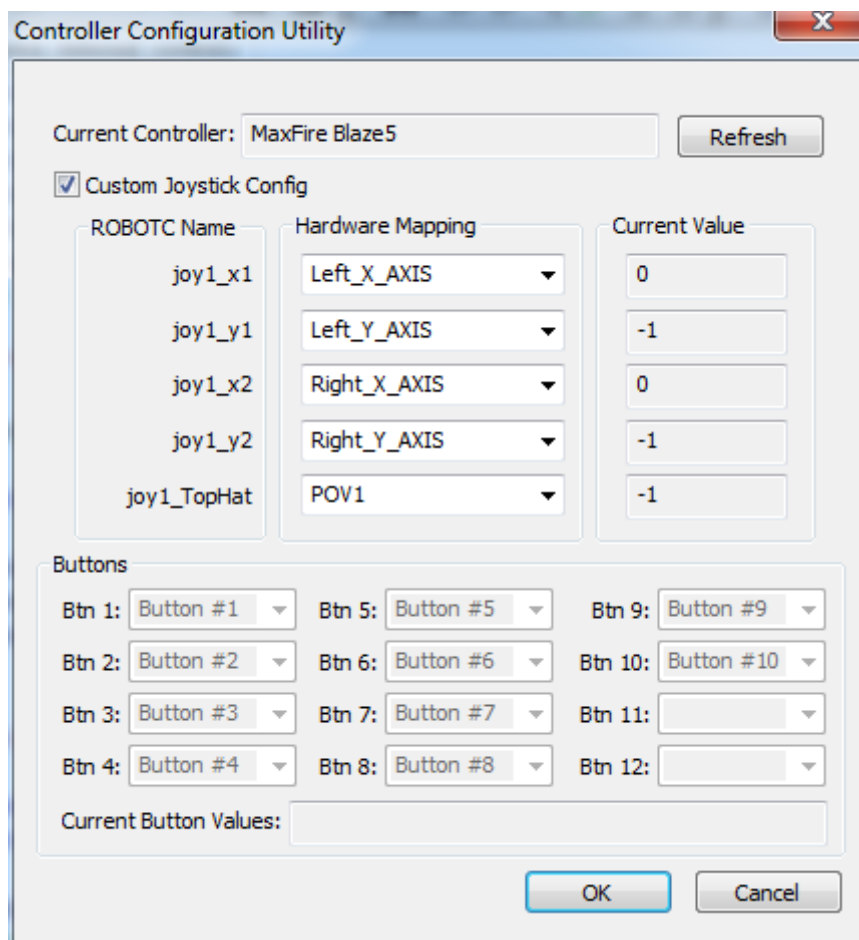
Obr. 46. zadání příkladu Mine Removal Challenge



Obr. 47. Zadání příkladu SoccerChallenge

5.9.2 Řešení

Když chceme robota ovládat dálkově pomocí joysticku v prostředí RVW, musíme ho mít správně nakonfigurovaného. Konfiguraci provedeme v nabídce menu Window|Configure Joystick. V první řadě je třeba provést kontrolu, jestli jsou aktivovaná potřebná tlačítka.



Obr. 48. Konfigurace joysticku

Dále je třeba na začátek programu vložit kód: `#include "JoystickDriver.c"`. To nám zajistí možnost práce se zařízením. Kdyby knihovna `JoystickDriver.c` nebyla v počítači dostupná, je třeba ji stáhnout ze stránek www.robotc.net.

Po základním nastavení je třeba zapnout grafické rozhraní, které je nutné pro to, aby joystick v programu opravdu fungoval, a také v něm můžeme vidět hodnoty, které jednotlivá tlačítka vrací při jejich stisknutí. Můžeme zde zjistit i názvy jednotlivých tlačítek. Okno zobrazíme opět výběrem z menu `Robot | Debugger windows | Joystick Control – Basic`.

Jak je vidět z následujícího obrázku, i když se mluví obecně o joysticku, je možné použít jakékoliv herní zařízení. Zajímavé by jistě bylo použití volantu s pedály na ovládání směru a rychlosti robota. V samotném kódu se kontroluje, jaké tlačítko je stisknuté, a podle toho se nastavuje síla motorů.



Obr. 49. Joystick control

```

23 while(true)
24 {
25     getJoystickSettings(joystick);
26     //nacteni hodnot z leveho D-padu (maly levy joystick)
27     //ovadani pohybu celeho robota
28     x=joystick.joy1_x2;
29     y=joystick.joy1_y2;
30     //kdyz neni D-pad uprostred
31     if(abs(y)>1 || abs(x)>0){
32         if(y>0)
33         {
34             //vypocet rychlosti motoru podle polohy d-padu
35             motor[motorB]=y-x;
36             motor[motorC]=y+x;
37         }
38         else
39         {
40             //vypocet rychlosti motoru podle polohy d-padu
41             motor[motorB]=y+x;
42             motor[motorC]=y-x;
43         }
44     }
45     else{
46         motor[motorB]=0;
47         motor[motorC]=0;
48     }
49     //vyber ze stisknutych tlacitek pro robotickou ruku
50     if(joystick.joy1_TopHat==0 && nMotorEncoder[motorA]>-20)
51         motor[motorA]=-30;
52     else
53         if(joystick.joy1_TopHat==4 && nMotorEncoder[motorA]<0)
54             motor[motorA]=30;
55         else
56             motor[motorA]=0;
57 }

```

Obr. 50. Řešení příkladů Mine RemovalChallenge a SockerChallenge

6 FOTBAL ROBOTŮ

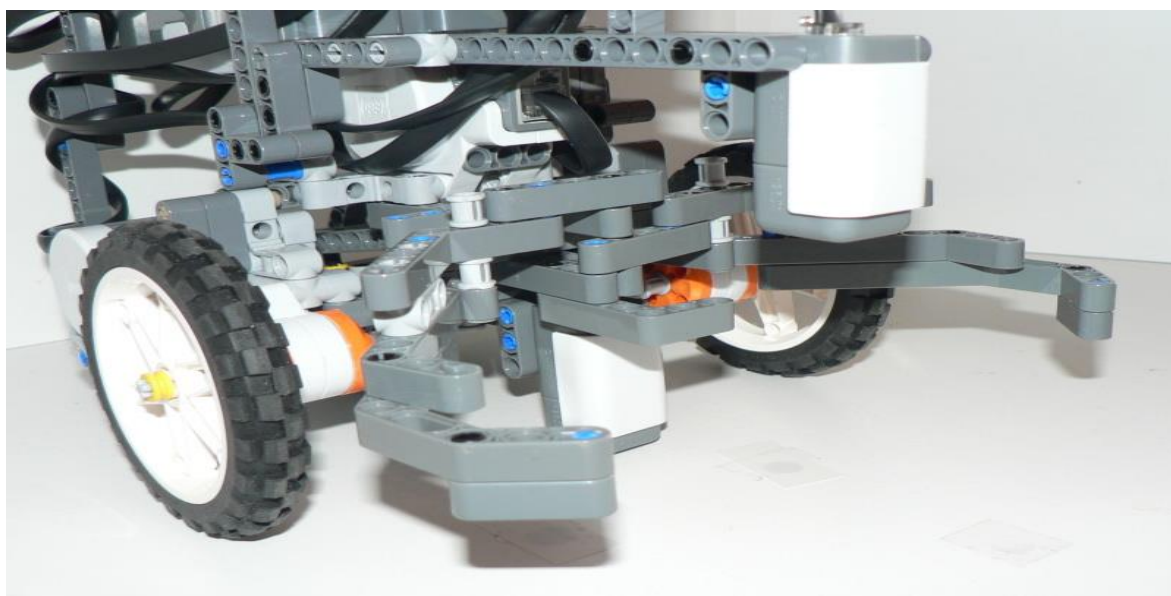
Poslední částí této práce je popis realizace fotbalu robotů. Princip jsem již popsal na začátku práce, proto nyní provedu pouze stručné shrnutí:

- Hrají dva roboti proti sobě
- Hřiště je rozděleno na dvě poloviny, na každé z nich je černou čarou vymezen prostor, kde se roboti mohou pohybovat
- Roboti hrají se speciálním míčem, který mohou „vidět“
- Branky jsou interaktivní, po vstřelení gólu navýší skóre a míč vrátí do hry

Popis problému bude rozdělen do kapitol podle navrhnutého harmonogramu projektu, přičemž sloučím hru robotů proti sobě 1 a 2 a vynechám poslední část vyhodnocení projektu. Hodnotit budeme se studenty až po realizaci celého projektu, k čemuž zatím nemohlo dojít.

6.1 Stavba robota

Při konstrukci robota jsem vycházel především z vlastních zkušeností ze stavby modelů v minulosti. Inspiroval jsem se také v knize [2]. Jako první část jsem vytvořil robotickou ruku, která uchopí míč. Její součástí je barevný senzor, pomocí kterého robot pozná, jestli je míč v prostoru ruky. Na začátku jsem použil ultrazvukový senzor na měření vzdálenosti, který ale nevracel přesné hodnoty, a robot míč vždy nerozpoznal. Při použití světelného senzoru může dojít k nepřesnostem při měnících se světelných podmínkách. V běžně osvětlené místnosti, kam přímo nesvítí slunce, ale chytání míče probíhá spolehlivě.



Obr. 51. Robotická ruka

Jako důležité se ukázalo vhodné umístění kompasu a senzoru pro vyhledávání infračerveného signálu. Musely být umístěny minimálně deset centimetrů od motorů. Udělal jsem pro ně proto samostatné rameno, kde jsou umístěny.

Robot samozřejmě nebude postavený nastalo. Vytvořil jsem proto model v prostředí Lego Digital Designer (LDD). Model je vytvořený i pro branku. Výhodou je, že po seskládání ve virtuálním prostředí je možné vyexportovat návod na sestavení robota krok po kroku.



Obr. 52. Fotbalový robot

6.2 Kalibrace robota

Před samotnou hrou je zapotřebí načíst hodnoty z kompasu a ze světelného senzoru. Robota umístíme zhruba do středu jeho poloviny hřiště čelem k soupeřově brance. Poté robot provede výstřel na prázdko, aby se nastavila ruka pro možnost chytání míče. Tato část je časově naddimenzovaná. Bude v ní proto prostor pro další úkoly, případně pro úpravu konstrukce robota.

```
60 //kalibrace robota
61 void Kalibrace()
62 {
63     //nacteni hodnoty svetelneho senzoru
64     kalSvetloHorni=SensorValue[svetloHorni];
65     //nacteni hodnoty kompasu
66     kalKompas=SensorValue[kompas];
67     //kalibracni vystrel
68     Vystrel();
69 }
```

Obr. 53. Kalibrace robota

6.3 Jízda robota

Robot může jezdit libovolně po své polovině. Nesmí však přejít černou čáru. V této fázi je proto důležité určit správnou vzdálenost od mantinelů, aby do nich zbytečně nenarážel a také aby se roboti kolem poloviny hřiště nesráželi, případně se nezachytili rukou. Dále je třeba zjistit hodnotu světelného senzoru, který je ve spodní části robota a je určen pro detekci černé čáry. Poslední veličinou, kterou je třeba vhodně zvolit, je rychlost robota.



Obr. 54. Hřiště

6.4 Vyhledání míče

Je to jedna z nejdůležitějších funkcionalit. Při použití IR míče a IR vyhledávače signálu danou problematiku značně ulehčí. Funkci obou zařízení jsem již popsal v teoretické části, proto opět pouze stručné shrnutí:

- IR míč vysílá infračervený signál tak, aby byl senzorem vždy k nalezení
- IR senzor přijímá signál z míče a vrací hodnoty 1 – 9 podle polohy míče

Po zvolení vhodné rychlosti jsem si nadefinoval dvě pole o deseti prvcích. V každém jsou uloženy hodnoty pro aktuální výkon motorů pro pohon robota při aktuální hodnotě IR senzoru. V prvcích s indexem 0 jsou hodnoty opačné. Robot se potom točí dokola, tím poznáme, že míč není v dosahu. Stane se tak většinou, když míč zapomeneme zapnout nebo má slabý zdroj napájení.

```

16 //definice pole hodnot pro levý motor
17 int levyIr[10]={-30,-30,-15,0,15,30,30,30,30,30};
18 //definice pole hodnot pro pravý motor
19 int pravyIr[10]={30,30,30,30,30,30,15,0,-15,-30};

```

Obr. 55. Definice polí

Robot tedy jezdí za míčem. Po detekci černé čáry se vrátí mírně zpět a vyhledávání míče pokračuje znovu. Problém vznikl, když byl míč u mantinelu a robot na něj najížděl pod úhlem menším než 45°. Protože byl míč v přímém směru, robot se již nenatáčel, najížděl na míč stále rovně, ale nedosáhl na něho. Proto si robot počítá, kolikrát narazí na černou čáru, aniž by našel míč. Po čtyřech pokusech poodjede dozadu, náhodně se otočí o 90° a kousek popojede. Tím se dostane do lepší pozice a k míči má lepší dostupnost. Jestli má robot míč v dosahu pro chycení, pozná podle změny hodnoty na světelném senzoru.

```

75 //nactení aktualmi hodnoty senzoru
76 int irHodnota=SensorValue[ir];
77 //promenna pro vypocet uhlu otoceni
78 int kamOtocit;
79 //kdyz neni robot na care, hodnoty indexu pole nejsou mimo rozsah
80 //a nema chyceny mic
81 if(SensorValue[svetloSpodni]>20 && irHodnota<10 && chyceno==false)
82 {
83     //nastaveni hodnoty vykonu leveho motoru
84     motor[levy]=levyIr[irHodnota];
85     //nastaveni hodnoty vykonu praveho motoru
86     motor[pravy]=pravyIr[irHodnota];
87 }

```

Obr. 56. Jízda robota za míčem

```

91     //zastaveni robota
92     motor[levy]=0;
93     motor[pravy]=0;
94     wait1Msec(200);
95     //robot odjede dozadu
96     motor[levy]=-20;
97     motor[pravy]=-20;
98     wait1Msec(1200);
99     //kdyz nebyl u cary uz 3x
100    if(++pocetPokusu>3)
101    {
102        //vypocet smeru otoceni
103        kamOtocit=random(10);
104        //podle vygenerovane hodnoty se robot natoci
105        if(kamOtocit<5)
106        {
107            motor(levy)=20;
108            motor(pravy)=-20;
109        }
110        else
111        {
112            motor(levy)=20;
113            motor(pravy)=-20;
114        }
115        wait1Msec(1000);
116        //robot se zastavi
117        motor[levy]=0;
118        motor[pravy]=0;
119        //vynulovani otacek motoru
120        nMotorEncoder[motorB]=0;
121        //dokud robot neodjede bokem nebo nenarazi na caru
122        while(nMotorEncoder[motorB]<270 && SensorValue[svetloSpodni]>20)
123        {
124            //robot popojede (kola se natoci o 270°)
125            motor[levy]=20;
126            motor[pravy]=20;
127        }
128        //vynulovani poctu pokusu o nalezeni mice
129        pocetPokusu=0;
130        //robot se zastavi
131        motor[levy]=0;
132        motor[pravy]=0;

```

Obr. 57. Změna polohy robota při nenalezení míče

```

50 task ChytMic()
51 {
52     //vynulovani neuspesnych pokusu o nalezeni mice
53     pocetPokusu=0;
54     while(true)
55     {
56         //kdyz je mic nalezen a neni chycen
57         if(SensorValue[svetloHorni]>(kalSvetloHorni+5) && chyceno==false)
58         {
59             //uzavreni ruky
60             motor[strela]=-20;
61             wait1Msec(500);
62             //robot vi, ze ma mic
63             chyceno=true;
64             //ruka zustane uzavrena
65             motor[strela]=0;
66         }

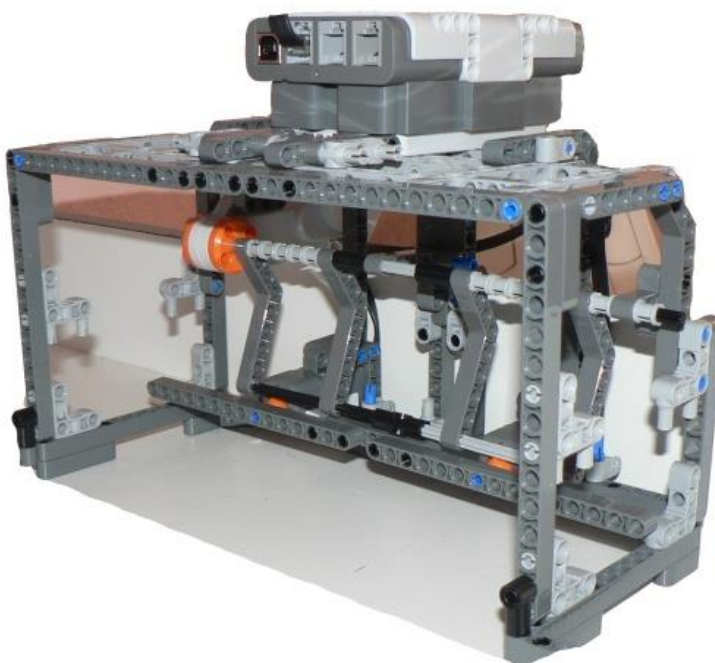
```

Obr. 58. Chycení míče

Posledním problémem, který bylo třeba vyřešit, byl fakt, že robot míč obtížněji chytal u mantinelů. Před mantinely jsem umístil zábrany, které míč nepřekoná, ale robotovi nevadí. Zábrany jsem vytvořil z rovných dílů stavebnice na výšku dvou dílů.

6.5 Vytvoření branek

Při konstrukci branek jsem musel vzít v potaz rozměry míče a hřiště. Pro fixaci jsem využil otvorů, do kterých se původně umísťovali překážky při plnění úkolu průchod robota bludištěm.



Obr. 59. Fotbalová branky

6.6 Funkcionalita branek

Branky mají dvě funkce. Počítají vstřelené góly a vrací míč do hry. Obě činnosti na sebe navazují. Branka pozná, že byl vstřelen gól, zvětší počet vstřelených gólů o jedničku a odráží balon ven z branky. Po několika pokusech jsem přistoupil k použití dvou dotykových senzorů. Všechny ostatní se ukázaly jako nespolehlivé. Ultrazvukový senzor si s míčem moc „nerozumí“ (ne vždy ho detekuje), světelné někdy počítají za branku, když robot nebo míč projede kolem branky, a IR vyhledávací senzor se nedá nastavit, aby míč nezjišťoval i mimo branku nebo zase nepoznal gól. Může se stát, že míč do branky doputuje příliš pomalu a dotykový senzor se nestiskne. Budeme to brát tak, že míč nepřešel brankovou čáru celým objemem.

```

23     //kdyz je stisknutý jeden ze senzor (padl gol)
24     if(SensorValue(dotyk1)==1 || SensorValue(dotyk2)==1)
25     {
26         //motor se otoci o 50 stupnu
27         while(nMotorEncoder[vystrel] < 50)
28         {
29             motor[vystrel] = 40;
30         }
31         //motor se zastavi
32         motor[vystrel] = 0;
33         //motor se vrati zpet
34         while(nMotorEncoder[vystrel] > 30)
35         {
36             motor[vystrel] = -100;
37         }
38         //motor se zastavi
39         motor[vystrel] = 0;
40         //pripocita se dalsi gol
41         gol++;

```

Obr. 60. Funkcionalita branky

6.7 Střelba na bránu

Poslední dovedností, kterou musíme robota naučit, je vystřelení na branku. Na branku robot střelí po chycení míče. Nenatáčí se vždy přímo na branku, protože před ní stává soupeř. Vypočítá se proto hodnota $(-5) - 5$, ta se přičte k zapamatované hodnotě při kalibraci kompasu. Poté se přepočítá do hodnot $0 - 359$, robot se na danou pozici natočí, rozjede se, dojde k černé čáře a vystřelí. Poté opět začíná s hledání míče.

```

38     //strelba na branu
39     void Vystrel()
40     {
41         //uvolni se ruka a mic je vystrelen vpred
42         motor[strela]=60;
43         wait1Msec(200);
44         //ruka se zastavi, aby mic vyletel bez problemu
45         motor[strela]=0;
46         wait1Msec(500);
47         //ruka se vrati do polohy pro chytani mice
48         motor[strela]=-40;
49         wait1Msec(50);
50         //robot vi, ze nema mic chyceny
51         chyceno=false;
52         //ruka se zastavi
53         motor[strela]=0;
54     }

```

Obr. 61. Střelba na bránu

```

150 void Zamir()
151 {
152     //nacteni aktualni hodnoty na kompasu
153     int aktKompas=SensorValue[kompas];
154     //promenna pro vypocet, kam robot pojede
155     int pozadovanyKompas;
156     //odchylka od smeru primo na branu
157     int uhel=random(10)-5;
158     //vypocet pozadovane hodnoty kompasu
159     pozadovanyKompas=kalkompas+uhel;
160     //prepocitani hodnoty kompasu do rozmezi 0°-359°
161     if(pozadovanyKompas<0 || pozadovanyKompas>=360)
162         pozadovanyKompas=abs(360-pozadovanyKompas);
163     //Dokud se robot nenatoci do pozadovane pozice +/- 1°
164     while(abs(SensorValue[kompas]-pozadovanyKompas)>1)
165     {
166         nxtDisplayCenteredTextLine(3,"pozKompas: %d",pozadovanyKompas);
167         nxtDisplayCenteredTextLine(4,"Rozdil: %d",pozadovanyKompas);
168         //robot se otaci
169         motor[levy]=-20;
170         motor[pravy]=20;
171     }
172     //robot se zastavi
173     motor[levy]=0;
174     motor[pravy]=0;
175     //dokud robot nenaravi na cernou caru
176     while(SensorValue[svetloSpodni]>20)
177     {
178         //robot jede vpred
179         motor[levy]=30;
180         motor[pravy]=30;
181     }
182     //robot se zastavi
183     motor[levy]=0;
184     motor[pravy]=0;
185     //robot vystreli
186     Vystrel();
187 }

```

Obr. 62. Míření na bránu

6.8 Hra robotů proti sobě

Podle mého názoru je hra robotů plynulá a nepostrádá určitý náboj. Já osobně jako tvůrce vidím některé nepřesnosti nebo možnosti zlepšení. Výsledek jsem proto ukázal několika přátelům a známým a hlavně studentům. Ti, jak známo, bývají největšími kritiky práce učitelů. I tato zkouška ohněm dopadla dobře. Největší pochvalou byl zřejmě nezapomenutelný výrok: „To je hustý!“.

Občas se stane, že robot nechytne míč, ale odstřelí ho při pokusu o zachycení. Většinou míč letí na polovinu soupeře, takže to není velký problém. Také se míč výjimečně dostane do brány velmi pomalu a branka to nepozná. Jak jsem již ale uvedl dříve, z pohledu pravidel je vše v pořádku. Všechny tyto nepřesnosti ale z celkového hlediska považuji za nepodstatné.

ZÁVĚR

Smyslem práce bylo s pomocí metod projektové výuky vytvořit sadu příkladů pro výuku základů robotiky a automatizace na střední škole. Po prostudování možností projektové výuky jsem navrhnul příklady pro úvodní seznámení s problematikou Lego robotů. K tomu jsem využil prostředí RVW. Dalším krokem byl návrh a řešení komplexního projektu na reálných robotech ze stavebnice Lego Mindstorms NXT. Tím je fotbal robotů. Při návrhu a realizaci jsem přihlédl ke správné motivaci studentů. V potaz jsem také musel vzít materiální a časové možnosti jak studentů, tak situaci na naší škole. Všechny úkoly jsem naprogramoval a zdrojový kód jsem okomentoval.

Myslím si, že jsem úkol splnil. Jako každý správný vývojář již teď ale vidím možná vylepšení. Dalo by se zapřemýšlet nad vzájemnou komunikací robotů, nakoupením dalších senzorů (např. senzor na přesné měření vzdálenosti) nebo vytvořením e-learningového kurzu. Kurz určitě vznikne v rámci přípravy na příští školní rok.

I když by jistě vylepšení byla celá řada, z vlastní zkušenosti mohu říci, že často méně je více. Není vhodné studenty zahltnout množstvím nových technologií nebo různými složitými řešeními. Je jistě lepší zvládnout méně náročné úkoly, ale tak, aby studenti vše ve větší míře pochopili, a hlavně, aby se dostavil reálný výsledek jejich práce. Musíme si uvědomit, že pracujeme s studenty střední školy.

Z didaktického hlediska je výuka pomocí robotů vhodná do škol technického typu, což naše škola splňuje. Rozvíjí u studentů kreativitu, zručnost, trpělivost a v neposlední řadě je nutí spolupracovat. Řada studentů se dostane k prostředkům, které by si za běžných okolností nemohli dovolit. To by se dalo považovat za slabší článek ve výuce, protože studenti nebudou mít možnost práce s roboty doma. Částečně je to eliminováno prostředím RVW.

S roboty pracujeme na naší škole v rámci kroužku již zhruba rok. Problematika proto pro mě nebyla nová, ale vypracování výše zmíněných úkolů mě posunulo opět o kus dále a utvrdilo mě v přesvědčení, že výuka pomocí Lego robotů může být zajímavá, inspirující a pro studenty prospěšná.

CONCLUSION

The purpose of this work was using project-based learning methods to create a set of examples for teaching the basics of robotics and automatization in high school. After studying the possibilities of project teaching I designed examples as an introduction to the issue of Lego robots. I used RVW environment to accomplish it. The next step was designing and implementing of complex project aimed at solving real robots from Lego Mindstorms NXT. The project is robot soccer. I considered the proper motivation of pupils during the design and implementation phase. I also had to consider the material and time demandingness for both students and the situation at our school. I programmed all the tasks and I commented source code.

I think I have fulfilled the task. I can already see possible improvements, like any good developer. You could think about the mutual communication of robots, purchasing of additional sensors (eg. A sensor for accurate measurement of distance), or creating e-learning course. Course will be certainly created in preparation for the next school year.

As we can see many possible improvements, based on my own experience, I can say that less is often more. Students should not be overwhelmed by count of new technologies or different complex solutions. It's certainly better to handle less demanding tasks, but in the way, that all students can fully understand issue and most importantly, to get real result of their work. We must realize that we work with high school students.

From the didactic point of view, teaching by the use of robots is suitable for schools of technical type, what fulfil our school. It develops student's creativity, skill, patience and it force them to cooperate. Many students will get to the resources that would normally be unable to afford. This could be considered a weak link in the classroom because students will not have the opportunity to work with robots at home. It's partially eliminated by RVW environment.

With robots at our school, within the robotics club, we have been working for about a year. The issue, therefore, was not new to me, but developing the above mentioned tasks moved me further on and reassured me in the belief that teaching using Lego robots can be interesting, inspiring and beneficial for students.

SEZNAM POUŽITÉ LITERATURY

- [5] KRATOCHVÍLOVÁ, Jana, *Teorie a Praxe projektové výuky*. Brno: MU 2006, ISBN: 89-210-4142-0.
- [2] NOVÁK, Daniel, *Elektrotechnické stavebnice v technické výchově*, Praha, 1997, ISBN: 80-86039-37-4.
- [3] SKALKOVÁ, Jarmila, *Obecná didaktika*. Praha: ISV, 1999, ISBN: 80-85866-33-1.
- [4] PAVELKOVÁ, Isabella, *Motivace žáků k učení: perspektivní orientace žáků a časový faktor v žákovské motivaci*. Praha: Universita Karlova, 2002, ISBN: 80-729-0092-7.
- [5] W. H. KILPATRICK, *The Project Method: The Use of the Purposeful Act in the Educational Process*, University of California, Eleventh Impression March, 1929.
- [6] *Robotické vzdělávání LEGO Mindstorms* [online]. ©2012 [cit. 2015-04-28]. Dostupné z: <https://lego.zcu.cz/web/>
- [7] *EDUXE s.r.o., Velké Pavlovice – distributor učebních pomůcek*, [online]. ©2015 [cit. 2015-04-28]. Dostupné z: <http://www.eduxe.cz/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

RVP	Rámcový vzdělávací plán.
ŠVP	Školní vzdělávací plán.
ČVUT	České vysoké učení technické
RVW	Robot VirtualWorld
LDD	Lego Digital Desinr
IR	Infračervený

SEZNAM OBRÁZKŮ

Obr. 4. Možnosti RVW	15
Obr. 5. Rembot s dotykovým senzorem	16
Obr. 6. Rembot s robotickou rukou	17
Obr. 7. Sada dílů 9797 [7].....	21
Obr. 8. Sada technických dílů 9695 [7]	21
Obr. 9. Kostka NXT [7]	22
Obr. 10. způsob připojení motorů a senzorů [7]	23
Obr. 11. Dotykový senzor [7]	24
Obr. 12. Ultrazvukový senzor [7]	24
Obr. 13. Zvukový senzor [7].....	25
Obr. 14. Světelný senzor [7]	25
Obr. 15. Barevný senzor [7].....	26
Obr. 16. Servomotor [7].....	26
Obr. 17. Gyroskopický senzor [7]	27
Obr. 18. Kompas [7]	28
Obr. 19. Senzor vyhledávání IR signálu [7]	28
Obr. 20. Hodnoty vrácené IR senzorem [7].....	29
Obr. 21. Infračervený míč[7]	29
Obr. 22. Nastavení motorů a senzorů	31
Obr. 23. Communication Link Setup	32
Obr. 24. Poll NXT Brick.....	33
Obr. 25. Nabídka Program Debug	34
Obr. 26. Definice senzorů a motorů.....	34
Obr. 27. Načtení dat z externích souborů	34
Obr. 28. Zadání úkolu Labyrinth Challenge	36
Obr. 29. Funkce pro výpis hodnot na kostku	37
Obr. 30. Hodnoty na virtuální kostce NXT	37
Obr. 31. Funkce pro otočení robota	38
Obr. 32. Úspěšné splnění úkolu	38
Obr. 33. Zadání úlohy Robo 500 1	39
Obr. 34. Funkce pro jízdu rovně a hlavní část programu	40
Obr. 35. Zadání příkladu Robo Slalom.....	40

Obr. 36. Část kódu robo slalom	41
Obr. 37. Zadání příkladu Line Runner 1	41
Obr. 38. Funkce pro přejetí čáry	42
Obr. 39. Kontrola přejetí čáry	42
Obr. 40. zadání příkladu Firefly Bot 1	43
Obr. 41. Řešení příklad Firefly Bot 1	43
Obr. 42. Zadání příkladu Obstacle Course	44
Obr. 43. Jízda robota po čáře	44
Obr. 44. Využití ultrazvukového senzoru	45
Obr. 45. Zadání příkladu autoattendance	45
Obr. 46. Hlavní část programu Auto attendance	46
Obr. 47. Zadání příkladu Pipebot Level 1	46
Obr. 48. Řešení úkolu PipebotLevel 1	47
Obr. 49. zadání příkladu Mine Removal Challenge	48
Obr. 50. Zadání příkladu SockerChallenge	48
Obr. 51. Konfigurace joysticku	49
Obr. 52. Joystick control	50
Obr. 53. Řešení příkladů Mine RemovalChallenge a SockerChallenge	50
Obr. 54. Robotická ruka	51
Obr. 55. Fotbalový robot	52
Obr. 56. Kalibrace robota	53
Obr. 57. Hřiště	53
Obr. 58. Definice polí	54
Obr. 59. Jízda robota za míčem	54
Obr. 60. Změna polohy robota při nenalezení míče	55
Obr. 61. Chycení míče	55
Obr. 62. Fotbalová branky	56
Obr. 63. Funkcionalita branky	57
Obr. 64. Střelba na bránu	57
Obr. 65. Míření na bránu	58

SEZNAM TABULEK

Tab. 1 Srovnání projektové a tematické výuky	13
Tab. 2 Vybrané příkazy jazyka RobotC.....	35

SEZNAM PŘÍLOH

- P I: Ukázka pracovního listu z programu RobotC
- P II: CD s textem práce a zdrojovými kódy

PŘÍLOHA P I: UKÁZKA PRACOVNÍHO LISTU Z PROGRAMU ROBOTC

ROBOTC

Challenge

Labyrinth Challenge

Challenge Description

To complete this challenge, program the robot to move to the end (the black striped area) of the Labyrinth. The robot must NOT cross any of the black lines.

Board Specifications

