# VÝUKA PROGRAMOVANIA POMOCOU GRAFICKÝCH ROBOTICKÝCH PROGRAMOVACÍCH JAZYKOV PRE ZAČIATOČNÍKOV A POKROČILÝCH

**PAVEL PETROVIČ**

## ABSTRAKT

*Druhá generácia autonómnych robotických stavebníc od firmy LEGO je stabilizovanou výukovou platformou. Počty inštalácií v školách možno počítať na stovky. Na mnohých základných i stredných školách, v centrách voľného času, a v záujmových kluboch vznikli v posledných desiatich rokoch krúžky záujmovej mimoškolskej činnosti so zameraním na robotiku. Výukové materiály v slovenskom jazyku sú zatiaľ spočítateľné skôr na prstoch jednej, najviac dvoch rúk. V tomto článku opisujeme sadu cvičení, ktoré sme pripravili a sú vhodné pre začiatočníkov - učiteľov i žiakov, využívajúcich platformu na výuku programovania i základov robotiky. Úlohy sú verejne prístupné na webstránkach združenia Robotika.SK. Sadu sme úspešne overili na niekoľkých školeniach učiteľov. Evidujeme komunitu desiatok učiteľov zo Slovenska, Čiech a Moravy, ktorí ju využívajú. V druhej časti článku sa venujeme pokročilým funkciám jazyka NXT-G, ktoré používatelia zriedkavo poznajú. Ich účel demonštrujeme na ilustratívnych príkladoch.*

**Kľúčové slová:** *robotické stavebnice, programovanie, vizuálne programovacie jazyky*

## ÚVOD

Robotické stavebnice LEGO Mindstorms sú pôvodne určené pre hračkársky trh. Zodpovedá tomu aj ich konfigurácia: malé množstvo senzorov v základnej zostave, nízka kapacita pamäte, proprietárne a neštandardné hardvérové i softvérové rozhrania, orientácia predovšetkým na jednoduché a zábavné aktivity. Napriek tomu sa rovnaká zostava predáva aj v línii Education, kde je ponúkaná ako učebná pomôcka. Ako taká vykazuje najlepšie parametre spomedzi všetkých dostupných pomôcok z hľadiska flexibilnosti, jednoduchosti použitia, používateľskej podpory, množstva dostupných materiálov, rozšíriteľnosti, interoperability, aplikovateľnosti a didaktickej primeranosti. Ako učebnú pomôcku ju možno využiť na vyučovanie programovania, fyziky i matematiky a na medzipredmetovú projektovú výuku a na výuku základov robotiky a riadenia.

*Flexibilnosť* – vychádza zo základnej koncepcie stavebníc LEGO, čiže čo najväčšej modulárnosti. Preto je množina využití ohraničená iba hranicami tvorivosti a fantázie používateľa. Zo stavebníc je možné zostaviť kolesové a kráčajúce mobilné artefakty i statické automatizované modely s meraním procesov a veličín. Bohatá sada dielov umožňuje vytvárať jednoduché stroje – páky, kladky, podvozky s riadením, diferenciálový pohon, otočné robotické ramená, kĺbové spoje, závesy a prívesy.

*Jednoduchosť použitia* – systém je navrhnutý tak, aby s ním mohol pracovať bez akýchkoľvek predchádzajúcich skúseností alebo znalostí každý. Podrobné tutoriály, ktoré sú súčasťou stavebníc, prevedú používateľa cez všetky podstatné vlastnosti systému, vďaka čomu sa stane pokročilým používateľom.

*Používateľská podpora* – webstránky firmy LEGO poskytujú vždy čerstvé aktualizácie firmware, driverov i technickej dokumentácie, ale najmä projekty navrhnuté používateľmi (NXTLOG), komunitný portál, videoukážky, súťaže, obežník (newsletter), množstvo publikácií, diskusné fóra a ďalšie. Významná je i autonómna komunita používateľov, napr. okolo združenia Lugnet (LEGO Users group).

*Množstvo dostupných materiálov* – ako ku všetkým produktom z rady LEGO Education, aj k robotickým stavebniciam existuje sada návodov a didaktických materiálov, ktoré je spravidla možné dokúpiť samostatne, napr. [5]. Okrem toho existuje množstvo materiálov zverejnených na rozličných používateľských portáloch, alebo publikácií od rôznych autorov napr. [4,6,7,8].

*Rozšíriteľnosť* – stavebnice sú vďaka svojej modulárnosti veľmi dobre rozšíriteľné. Samotné LEGO i niekoľko ďalších firiem dodáva doplnkové senzory na meranie rôznych veličín: accellerometer, gyroskop, kompas, infračervený senzor na meranie vzdialenosti, RFID senzor, kamera, multiplexery pre pripojenie väčšieho množstva senzorov i motorov, radiče na servomotory a jednosmerné motory, senzory na robotický futbal a ďalšie. Elektrické rozhrania sú zdokumentované a preto je pre skúsenejších používateľov otvorená aj cesta vytvárania vlastných elektronických komponentov prepojiteľných so stavebnicami. Osobitnou kapitolou sú sady tuctov profesionálnych senzorov zo sérií LogIT spoločnosti DCP Microdevelopments [9] a NXT Vernier [10]. Stavebnice NXT je možné integrovať do zložitejších a náročnejších robotických výrobkov zostavených zo stavebnice Tetrix, ktorá priamo predpokladá riadenie pomocou programovateľnej jednotky LEGO NXT.

*Interoperabilita* – okrem uvedeného sú stavebnice kompatibilné s obrovským množstvom stavebníc LEGO, vrátane výukových zostáv zameraných na jednoduché stroje, obnoviteľné energie a pneumatiku. K týmto zostavám existujú ďalšie didaktické materiály s pracovnými listami pre študentov i učiteľov. Z hľadiska softvérovej interoperability je systém možné programovať v alternatívnych programovacích prostrediach – od systému RobotC vyvinutého na univerzite Carnegie Mellon v Pennsylvánii spolu s celým systémom výukových materiálov Robotics Academy, cez textovo-orientovaný jazyk NXC, tiež

so syntaxou jazyka C, až po jazyk Java pomocu prostredia Lejos. Štandardizovaný protokol BlueTooth™ dovoľuje robotom komunikovať s bežiacimi programami na PC, mobilných telefónoch a iných zariadeniach a takýmto spôsobom vytvárať rozsiahlejšie integrované a interaktívne výukové projekty.

*Aplikovateľnosť* – stavebnice sú použiteľné prakticky na všetkých stupňoch škôl, pohybujúce sa kocky zaujmú aj najmladšie deti, interaktívne modely s pripravenými programami sú vhodné pre deti na prvom stupni ZŠ, schopnosť abstraktne myslieť sa objavuje na druhom stupni ZŠ, kedy sa programovaním robotických modelov v grafickom programovacom jazyku žiaci zoznamujú so základmi programovania. Pre stredoškolákov sú vhodné interdisciplinárne alebo rozsiahlejšie integrované projekty a študenti vysokých škôl využívajú stavebnice na zoznámenie sa s princípmi robotiky, riadenia, umelej inteligencie, komunikácie a programovania v jazyku Java.

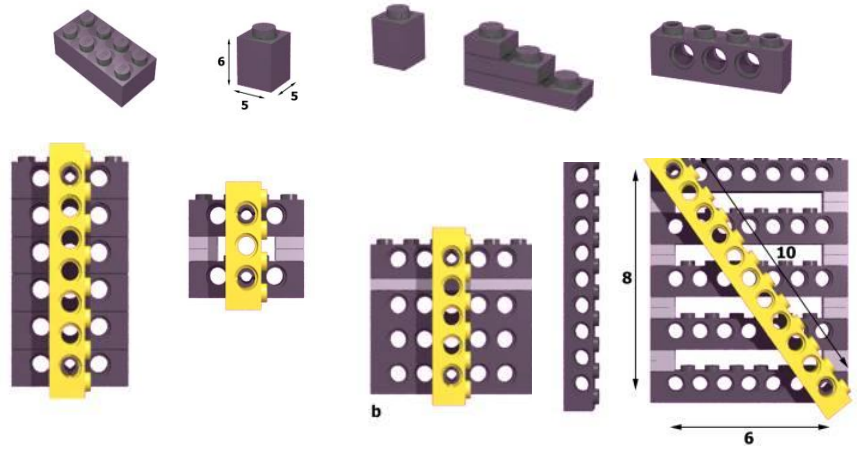
*Didaktická primeranosť* – z hľadiska didaktických trendov možno pozorovať, že do centra pozornosti sa dostáva žiak/študent, kým rola učiteľa sa mení na sprievodcu a partnera na ceste žiaka k poznaniu, to si však vyžaduje prítomnosť situácií vo vzdelávacom procese, kde má žiak možnosť objavovať. Presne to umožňujú robotické stavebnice a interaktívne robotické projekty. V našom vzdelávaní stále prevláda akademický spôsob výučby a to od najmladšieho veku. V mnohých prípadoch je nesporne nezastupiteľný, avšak ostáva tu veľký priestor, kde tieto tradičné metódy možno nahrádzať alternatívami, ktoré sú zároveň katalyzátorom záujmu žiakov o školu a vzdelávanie. Akademický spôsob vzdelávania totiž niektorým žiakom z takých, či iných dôvodov nevyhovuje a nevedia sa mu prispôsobiť. Výsledkom je ich vnútorná frustrácia, nechuť k učeniu a ku škole vôbec čo v neposlednej rade vedie aj k zhoršeniu návykov, správania a myslenia jednotlivcov. Z našej vlastnej skúsenosti vieme, že takíto žiaci vedia vyniknúť v alternatívnych formách výučby – napríklad v stavbe robotickým modelov a získať tak nesmierne dôležité sebavedomie, uznanie spolužiakov a učiteľa, ktoré ich pozitívne motivuje k dosahovaniu lepších študijných výsledkov aj v iných predmetoch.

V nasledujúcich statiach opíšeme použitie robotických stavebníc ako užitočnú učebnú pomôcku, sadu úloh pre úvodné stretnutia s robotickými stavebnicami pre začiatočníkov, sadu úloh, pomocou ktorých je možné zoznámiť sa so základmi programovania robotických stavebníc a programovania vôbec a na záver sa venujeme pokročilejším témam.

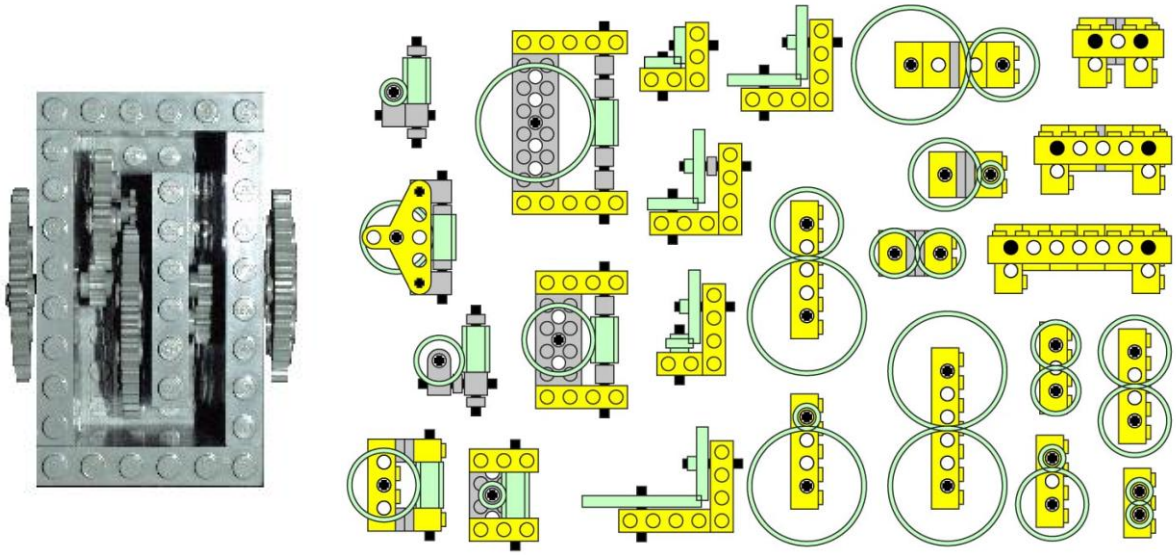
## 1 UČEBNÁ POMÔCKA

*Vyučovanie informatiky* – medzi kompetencie, ktoré by žiaci na vyučovaní informatiky mali získať, patrí a vždy bude patriť schopnosť abstraktne plánovať kroky nejakého procesu, napríklad výpočtu, čiže programovať. V prípade programovania robotov sa o výpočet jedná len skryte alebo priamo až v pokročilejších projektoch, skutočný zámer je spravidla nejaká činnosť robota v reálnom svete, ktorá však takisto pozostáva z jednotlivých krokov, podmienok, cyklov, systém si udržuje vnútorný stav pomocou premenných a na zložitejšie a opakované operácie si v programovacom jazyku môžeme vytvárať funkcie, resp. makrá. Rozdiel je však v tom, že programovanie činnosti v reálnom svete je omnoho konkrétnejšie a zrozumiteľnejšie ako napr. programovanie triedenia zadaných číselných prvkov, či vyhľadávanie podslov v reťazci znakov. Naviac, programovateľný jazyk je vytvorený tak, aby „skladanie činnosti“ bolo čo najzrozumiteľnejšie a tvorené prostredníctvom skladania grafických blokov do postupností. Štruktúrované príkazy ako vetvenie programu (podmienky) a cykly sú znázornené graficky (analogicky diagramom aktivít). Pochopením pojmov cyklus, podmienka a premenná v takýchto situáciách vytvárame pre žiaka ideálne východisko pre jeho budúce úspešné zvládnutie programovania v bežných programovacích jazykoch. Preto si dovolíme tvrdiť, že robotické stavebnice sú ideálnou platformou pre základy vyučovania programovania.

*Vyučovanie matematiky a fyziky* – hodnota robotických stavebníc spočíva v ich širokých možnostiach využitia. Už v samotnej konštrukčnej mechanike LEGO kociek sa skrýva množstvo zaujímavých závislostí, ktoré môžu obohatiť nejednu vyučovaciu hodinu matematiky. Na obrázku 1 hore je zobrazený pomer medzi výškovou a šírkovou vzdialenosťou LEGO kociek a medzi výškou nižších a vyšších kociek. To dovoľuje spájať kocky priečnymi prepojovacími kockami v rôznych konfiguráciách, ako ukazuje obrázok 1 dole.




**Obr. 1** Hore: základné druhy kociek – pomer medzi výškou a šírkou je 6:5, tri nízke tvoria výšku jednej vysokej. Zdroj:lugnet.com; dole: možnosti prepojenia kombinácií kociek tak, aby konektory presne zapadali (násobenie zlomkov), vpravo Pythagorejský trojuholník. Zdroj: lugnet.com.



**Obr. 2** Vľavo: skrinka s LEGO prevodmi (násobenie zlomkov). Vpravo: kombinácie prevodov, zdroj: lugnet.com.

Podobne zaujímavé úlohy získame pri kombinovaní rozličných prevodov. Obrázok 2 vľavo ukazuje príklad prevodovej skrine, ktorej pomer prevodu môžeme vypočítať z rozostavenia ozubených prevodových kolies vyskytujúcich sa vo veľkostiach 8, 16, 24 a 40 zubov. Obrázok 2 vpravo ukazuje väčšinu možných vhodných spôsobov prepojenia prevodových kolies a základné krížové prepojenia.

Ešte zaujímavejšie úlohy sa týkajú vyučovania fyziky. Zo stavebníc môžeme vytvárať jednoduché stroje – kladky, páky, ale môžeme ich prepojiť s ďalšími špecializovaným stavebnicami – pre prácu s obnoviteľnými zdrojmi, ktorá obsahuje energometer – zariadenie na meranie vykonanej alebo vygenerovanej práce v Jouloch, alebo so stavebnicami obsahujúcimi pneumatické prvky – piesty, pumpu, tlakovú nádobu, manometer. Platforma je ideálnou pomôckou pre tvorbu skupinových dlhodobejších projektov, v ktorých žiaci najskôr teoreticky analyzujú určitý problém a potom k nemu zostroja model. Takýto model potom môže ostatným žiakom slúžiť ako interaktívna učebná pomôcka s výukovými aktivitami, na ktorých pochopia fyzikálny pojem, závislosť, alebo fenomén.

## 2 EXISTUJÚCE MATERIÁLY

Ku stavebniciam existujú kvalitné didaktické materiály dodávané priamo firmou LEGO [5]. Obsahujú množstvo pripravených programov, pracovné zošity pre žiakov, plány učebných hodín pre učiteľov a kompletnú príručku pre učiteľov. Okrem toho vzniklo viacero kvalitných a rozsiahlych materiálov od tretích strán a autorov. Medzi najznámejšie patria Robotics Academy z Carnegie Mellon [13] a pracovné zošity Josef Lückinga [7,8], učebnice robotického vzdelávacieho programu zameraného na dievčatá, Roberta [14]. V slovenčine existuje materiál vytvorený v rámci projektu DVUI [15], niekoľko diplomových prác [16,17]. Od roku 2009 vylepšujeme sadu úloh s riešeniami, ktorá je k dispozícii na [4].

## 3 NAJJEDNODUCHŠIE ÚLOHY

Aby bolo možné so stavebnicou pracovať, je vhodné, aby sa používatelia: či žiaci, študenti alebo učitelia zoznámili s výukovou platformou. Ak by sme sa na tento proces pozreli filozoficko-didakticky, zamysleli by sme sa takto: naviesť používateľa na experimenty, ktoré mu pomôžu vybudovať si v mysli efektívne porozumenie základným princípom, je na jednej strane výhodné, na druhej strane to môže na prvý pohľad odporovať zásadám Konštrukcionizmu, kde žiak sám tvorí podľa svojho tempa a nápadov a pritom princípy postupne objavuje. Tento proces však môže byť usmerňovaný a plánovaný a to najmä v prípade, keď pedagóg má už dostatočne široký rozhľad a nadhľad nad problematikou, princípmi, súvislosťami a poznatkami, ktoré žiak len objavuje. V opačnom prípade môžu jednoducho objavovať spolu a zdieľať pritom svoje postupy myslenia, inšpirovať a obohacovať sa jeden druhého. Prax nám zatiaľ potvrdzuje – a malo by to byť predmetom ďalších diskusií a serióznych výskumov – že navedenie a usmernenie používateľa na konkrétne experimenty a úlohy vyvoláva u používateľov živý záujem a zároveň vedie k rýchlejšiemu a dôkladnejšiemu pochopeniu princípov.

*Cvičenie 1* – zapnutie/vypnutie jednotky NXT, pohyb v menu pomocou ovládacích tlačidiel. V tejto fáze nevysvetľujeme, čo jednotlivé položky menu znamenajú, ale spôsob pohybu v menu. Napriek pomernej intuitívnosti aj skúseným používateľom chvíľku trvá, kým porozumejú, ako sa jednotka pomocou tlačidiel ovláda a ako je menu organizované. Žiaci môžu dostať za úlohu objaviť ako funguje pohyb v menu na kocke a prípadne nakresliť jeho stromovú štruktúru.

*Cvičenie 2* – pochopenie činnosti senzorov. Režim View na kocke NXT zobrazuje aktuálnu hodnotu načítanú zo senzora. Myslíme si, že kým je pozornosť a záujem používateľa na najvyššej úrovni, mal by sa zoznámiť s tým, ako senzory snímajú veličiny z prostredia a prevádzajú ich na číselnú formu. Treba pripraviť povrch rozličnej farby (napríklad papier s nalepenými farebnými plastovými páskami) a tiež ukázať ako sa dá dotykový senzor rozšíriť pomocou osky, na ktorej je

upevnený nárazník, prediskutovať ohraničenie ultrazvukového senzora na meranie vzdialenosti, vyskúšať ako sa správa svetelný senzor v rôznej vzdialenosti a pod rôznym uhlom natočenia, vysvetliť jeho dva rôzne režimy (je vhodné mať pripravený prenosný zdroj svetla, napríklad ručnú baterku). V tomto cvičení je vhodné zadať niekoľko motivačných cvičení, výsledkom by malo byť, že sa používatelia budú dobre orientovať v druhoch a rozsahoch hodnôt jednotlivých senzorov ako i v ich fyzikálnych vlastnostiach – napr. prečo svetelný senzor s červeným svetlom nevie rozlíšiť medzi bielym papierom a červenou páskou, či aké hodnoty ultrazvukový senzor nedokáže zmerať. Nemali by sme vynechať zabudované otáčkové senzory v motoroch.

*Cvičenie 3* – ak sa už používateľ orientuje vo funkcionalite senzorov, môže si ju overiť pomocou zabudovaných programov TryMe, ktoré využívajú zvukový výstup, aby ich význam znázornili efektom v reálnom svete. Zdá sa nám podstatné začať s meraním v režime View, pretože takto si používateľ skonštruuje presnejšie významy na základe skúsenosti, pochopí ohraničenia a presnosť senzorov dôkladnejšie. Režim View a TryMe totiž znázorňujú rovnakú funkciu, len používajú odlišnú modalitu. Nejde teda o analógiu ku kontrastu medzi porozumením zážitkom a teoretickým vysvetlením. Tu v oboch prípadoch dochádza k porozumeniu zážitkom, ale režim View presnejšie vystihuje, čo senzory dokážu zmerať. Opäť platí, že nezabudneme na funkcionalitu otáčkových senzorov zabudovaných v motoroch.

*Cvičenie 4* – v nasledujúcej fáze sa používateľ zoznámi so základným režimom programovania priamo na jednotke NXT bez použitia PC, čiže v režime NXT Program. Predtým ale problematiku môžeme uviesť tak, že budeme mať v kocke nahraté nejaké existujúce, avšak veľmi jednoduché, programy, vopred pripravené na PC a prenesené do NXT. V každom prípade však potrebujeme zhotovené modely robotov, ktoré sa dokážu pohybovať. Možno použiť buď základný model z priloženého návodu, alebo ešte jednoduchšie – rýchlo zostrojiť „5-minútového robota“ podľa návodu z komunity používateľov [3]. Najskôr sa teda používateľ zorientuje vo význame funkcie menu „Software files“, štartovaní a zastavovaní programov. A keďže komunikácia dnes už patrí k základným a kľúčovým vlastnostiam zariadení, nie je dôvod nevyužiť túto príležitosť na vysvetlenie komunikácie BlueTooth – párovania zariadení a posielania programov z jednej NXT na druhú. Môžeme postupovať napríklad tak, že na každú kocku pripravíme iný program. Používatelia v triede si potom programy môžu navzájom vymieňať posielaním cez rádiové rozhranie BlueTooth, pričom musia zistiť, ako sa rozhranie aktivuje a používa.

*Cvičenie 5* – Keď používatelia uvideli, čo je možné dosiahnuť jednoduchými programami, až teraz je vhodné dať im do rúk možnosť písať vlastné programy pomocou zabudovaného editora NXT Program. Zostrojíme napríklad programy, ktoré preskúmavajú prostredie a vyhýbajú sa prekážkam, hľadajú svetelný zdroj, alebo sa pretekajú, ktorý robot prejde medzi dvoma čiernymi čiarami a späť rýchlejšie. Pri snahe modifikovať programy postupne používateľ zistí, že režim NXT Program je veľmi obmedzený a na splnenie náročnejších úloh je potrebné naučiť sa vytvárať programy na PC.

## 4 PROGRAMOVANIE

V tejto stati opíšeme niekoľko úloh (mini-projektov), ktoré je možné využiť pri vyučovaní programovania pomocou grafického jazyka NXT-G. Ich kompletné zadania a riešenia sú dostupné na [4]. Programy opakovane využívame pri školeniach učiteľov zameraných na zoznámenie sa s platformou NXT ako i s deťmi na robotických krúžkoch. Aktivity sú zoradené postupne od jednoduchších k náročnejším, ale uvedená postupnosť nie je jediná možná, je to iba jedna z viacerých trajektórií cez sadu pomerne nezávislých úloh. Úlohy sú zostrojené tak, aby si vnímavý používateľ v mysli vybudoval čo najefektívnejším spôsobom dôkladný prehľad o konceptoch použitých v programovacom jazyku. Priemerný alebo málo vnímavý používateľ bude potrebovať pomalšie tempo a viac motivačných úloh a niektoré princípy vypustiť úplne. Na niekoľkých miestach v texte rozlišujeme doplňujúce úlohy a aktivity podľa stupňa pokročilosti používateľa. V úlohách používame robota, ktorý sa dokáže pohybovať vpred, otáčať na mieste a na ktorého môžeme modulárne pripájať rôzne typy senzorov. Je možné použiť základný model podľa návodov v stavebnici, 5-minútového robota menovaného vyššie, alebo ľubovoľný iný model, ktorý spĺňa tieto kritériá.

**Úloha1.** *Zadanie:* Vytvorte program, ktorý spôsobí, že robot prejde po hranách stredne veľkého štvorca. Robot sa v rohoch štvorca má otáčať na mieste. *Riešenie:* prvé riešenie, ktoré by používatelia mali vytvoriť pozostáva z postupnosti ôsmich pohybových blokov – striedavo pohyb vpred a otočenie na mieste. Ak nejde o najmladších žiakov, na tomto mieste je hneď vhodné zasvätiť používateľov do cyklického bloku so stanoveným počtom opakovaní (režim cyklu Count). Pokročilejších používateľov môžeme tiež zoznámiť s možnosťou definovania vlastného bloku – v tomto prípade na prejdenie jednej strany a otočenie sa o 90 stupňov na jej konci. Pokročilejším používateľom tu predvedieme možnosť zobraziť hodnoty otáčkových senzorov na displeji PC v nastaveniach bloku celkom v ľavom dolnom rohu obrazovky a možnosť vynulovania otáčkových senzorov kliknutím na písmenko 'R' v tejto oblasti (reset). Používatelia si vyskúšajú, že takýmto spôsobom vedia ručne zmerať počet stupňov otočenia motora potrebných na otočenie robota o 90 stupňov, resp. na prejdenie požadovanej vzdialenosti. Tu je niekedy potrebné robota pripojiť stlačením tlačidla „Connect“ v režime prezerania pamäte jednotky NXT, takže túto funkcionalitu je používateľom tiež potrebné ukázať. V prípade, že je PC vybavené zariadením BlueTooth, používateľom vysvetlíme spôsob pripájania robota cez bezdrôtové spojenie, pretože tým ušetria množstvo času a námahy pri riešení všetkých ostatných úloh a bol by hriech túto možnosť nevyužiť. *Komentár:* Cieľom tejto úlohy je zoznámiť sa s najjednoduchším použitím pohybového bloku – na priamy (regulovaný) pohyb vpred a otáčanie na mieste. Používateľ by si tu mal dôkladne a dôsledne uvedomiť množstvo nových informácií: prvýkrát zažije prípravu programu na PC a jeho prenos do jednotky NXT, pochopí čo je program, tok výpočtu, spôsob downloadovania, konfigurácie jednotlivých blokov, môžeme mu vysvetliť režim regulácie pri pohybe vpred – väčšina robotov sa nebude pohybovať vpred priamo, ale viditeľne regulovať svoju tendenciu vyjsť z priameho pohybu, v každom prípade by si mal uvedomiť (ne)súvislosť medzi údajom 'degrees' v pohybovom bloku a počtom stupňov, o ktoré sa robot skutočne otočí, význam posuvníka 'steering', ktorý vo svojich krajných polohách otáča robota na mieste. Je možné v úlohe ďalej experimentovať, pozorovať a uvedomovať si: aké hodnoty nastavenia 'power' vedú k akým skutočným správaniam? Ako by bolo možné „zaobliť“ rohy štvorca – t.j. robot sa neotáča na mieste, ale jedno koliesko stojí, druhé sa otáča, aký to má vplyv na počet otáčok potrebných na otočenie o 90 stupňov? (ak to

morfológia robota dovoľuje:) čo sa stane, ak zmeníme veľkosť kolies? Ako ťažké je vyladiť program tak, aby sa robot pohyboval naozaj presne do štvorca? Čo pre NXT znamená „pohyboval presne“? Ako sa robot správa na konci každého pohybu? Aký vplyv má zmena povrchu (napr. rozdiel medzi pohybom na zemi a na stole)? Úlohu môžeme koncipovať ako súťaž skupín – komu sa podarí kopírovať štvorec nakreslený/nalepený na stole čo najpresnejšie?

**Úloha 2.** *Zadanie*: Vytvorte program, ktorý spôsobí, že robot bude vykonávať sústavný pohyb v tvare osmičky. *Riešenie*: vzhľadom na symetriu je vhodné osmičku rozložiť na dve analogické časti (pravotočivá a ľavotočivá zatáčka rovnakej veľkosti) a to buď na dvojice príkazov: pohyb po kruhovom oblúku a pohyb vpred, alebo je možné vystačiť aj s jedným pohybom po kruhovom oblúku – získame guľatejšiu osmičku. Na nekonečný pohyb použijeme nekonečný cyklus. *Komentár*: používateľ zažije nasledujúce princípy: ako robot reaguje na rozličné nastavenie parametra 'steering' pri pohybe po kružnici, aký vplyv má jeho morfológia na tento pohyb (je výhodné, ak jednotlivé skupiny pracujú s rôznymi druhmi robotov, potom možno pozorovať, že otáčanie niektorých konštrukcií je náročnejšie), iné kritérium opakovania cyklu (nekonečný cyklus) a prípadne vnorené cykly, ak k úlohe pridáme požiadavku, že po každých troch osmičkách si robot musí dať chvíľu prestávku. Zároveň používatelia zistia, že presné vyladenie parametrov pohybu robota je niekedy náročnejšia úloha.

**Úloha 3.** *Zadanie:* Naprogramujte robota tak, aby preskúmaval svoje prostredie a pritom sa vyhýbal prekážkam. *Riešenie:* môžeme si vybrať, či úlohu vyriešime pomocou dotykového senzora a nárazníka, alebo pomocou ultrazvukového senzora na meranie vzdialenosti – rôzne skupiny môžu riešiť úlohu rôznymi spôsobmi. Pre najpokročilejších používateľov zadáme obmedzenie, že na robota nemôžu pripájať žiadne senzory! Budú tak odkázaní na použitie otáčkových senzorov, ktoré sú zabudované v robotoch, vychádzajúc z predpokladu, že kolieska robota sa po narazení do prekážky prestanú pohybovať (to docielime najmä vtedy, ak motory nastavíme na nízky výkon. V tejto úlohe pokročilejším používateľom (s mladšími a začiatočníkmi sa sem môžeme viackrát znovu vrátiť) predvedieme ekvivalenciu rozličných zápisov programu a pokúsime sa docieliť, aby pochopili v čom spočíva rozdiel: čakací blok, ktorý neumožňuje počas čakania na splnenie podmienky vykonávať žiadne iné testy a operácie, resp. cyklus s ukončovacou podmienkou, v ktorom môžeme pomocou bloku vetvenie/podmienka reagovať aj na iné podnety – napríklad na zvukový signál. Najpokročilejším používateľom však môžeme predviesť spracovanie nezávislých podnetov v dvoch paralelne vykonávaných vetvách programu obsahujúcich jednoduché čakacie bloky. *Komentár*: Táto úloha predstavuje konceptuálny skok, autonómny robot sa nielen pohybuje, ale už aj vníma svoje prostredie a stáva sa z neho plnohodnotný inteligentný agent. Ak je to možné, snažíme sa dosiahnuť, aby si používatelia uvedomili, že bloky, ktoré spracovávajú hodnotu zo senzora, v tom okamihu, keď sa práve vykonávajú, používajú vždy aktuálnu číselnú hodnotu, ktorú senzor v predchádzajúcom okamihu zaznamenal.

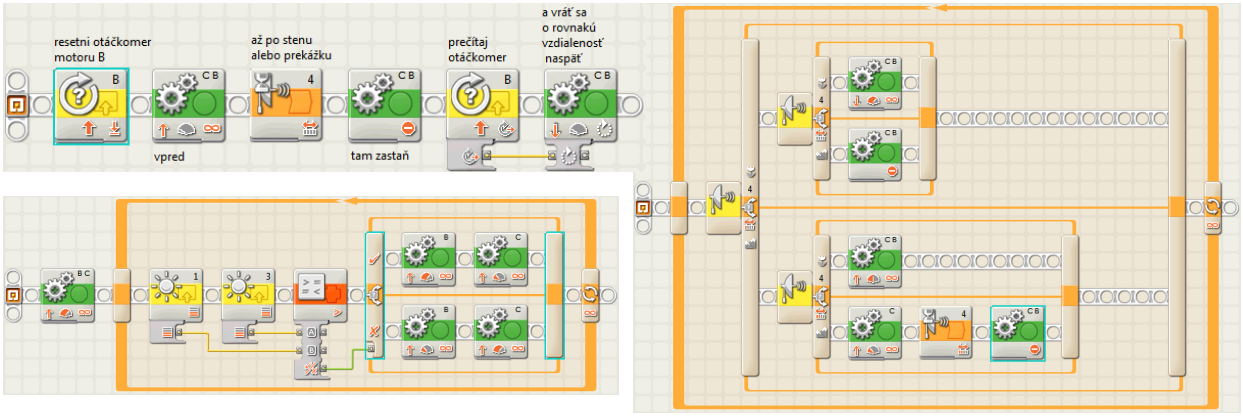
**Úloha 4**. *Zadanie:* Naprogramujte robota tak, aby vyštartoval smerom vpred, prišiel ku stene a zacúval naspäť presne na to miesto, kde vyštartoval. *Riešenie:* Táto úloha vyžaduje tri nové myšlienky: 1) účel senzorových blokov, ktoré nevykonávajú žiadnu akciu, iba poskytujú aktuálnu hodnotu zo senzora, 2) vytváranie dátových tokov medzi jednotlivými blokmi pomocou farebných prepojovacích káblikov, 3) spôsob vybalenia prípojek jednotlivých blokov – po kliknutí na stred dolnej strany štvorca, ktorým sa blok v programe zobrazuje. Riešenie je potom priamočiare – na začiatku programu vynulujeme otáčkový senzor v jednom z motorov, ktoré robot používa na pohyb vpred, pomocou dotykového alebo ultrazvukového senzora čakáme na priblíženie ku stene alebo prekážke, následne prečítame hodnotu z rovnakého otáčkového senzora (hodnota je v stupňoch) a prepojíme ju do ďalšieho pohybového bloku – spiatočný chod – do prípojky duration. *Komentár:* Používatelia overia činnosť programu z rôznych štartovných pozícii. Kreslenie dátových káblikov medzi blokmi si niekedy vyžaduje trochu trpezlivosti na strane používateľa, pretože programovacie prostredie sa snaží smerovať cestu káblika tak, aby prechádzal medzi blokmi optimálne. Preto je vhodné venovať 5-10 minút len trénovaniu kreslenia prepojek a ich mazaniu (kliknutím na cieľový terminál, alebo označením nefunkčného káblika a stlačením klávesu Del).

Takto si používatelia všimnú, že existujú tri druhy(farby) prepojek (o dátových typoch sa dozvedia viac neskôr) i to, že niektoré prípojky nie je možné prepojiť a vznikne tak nefunkčný šedý bodkovaný káblik, ktorý treba označiť a vymazať. Je úľavou, ak používatelia pochopia rozdiel medzi vstupnými a výstupnými prípojkami, pretože toto je v programovacom prostredí znázornené nedostatočne). Do pozornosti dávame materiál [11], ktorý spracoval autor a obsahuje popis všetkých programovacích blokov i ich prípojok v slovenčine. Pokročilí používatelia (alebo rôzne skupiny) môžu vytvoriť aj alternatívnu verziu použitím časovača namiesto otáčkového senzora. Zaujímavé je porovnať presnosť oboch prístupov. Pripravený pedagóg bude mať k dispozícii dve batérie: nabitú a takmer vybitú. Používatelia si overia, že s nabitou batériou sa robot pohybuje takmer dvakrát rýchlejšie ako s vybitou a uvedomia si, že používanie časovača, resp. časového intervalu vôbec pri navigácii robota je veľmi nespoľahlivé.

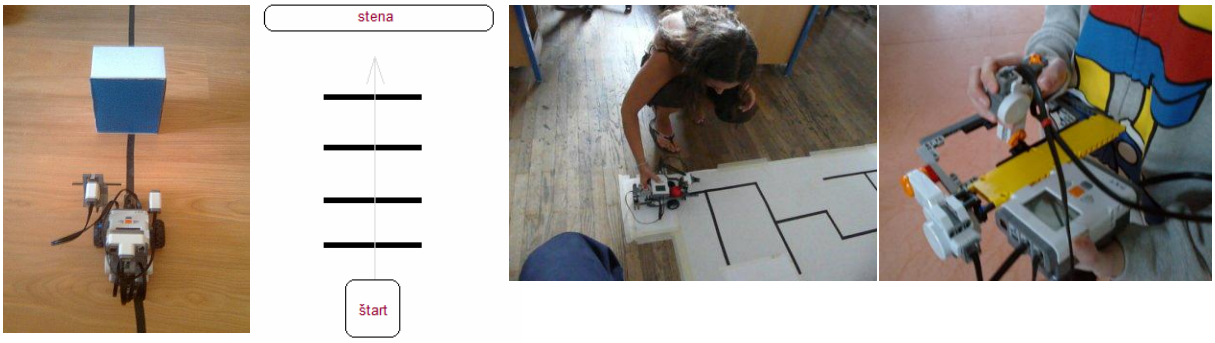
**Úloha 5.** *Zadanie:* Naprogramujte robota, ktorý v miestnosti nájde svetlo. *Riešenie:* Na robota pripojíme dva svetelné senzory, ktoré smerujú vpred, jeden natočený mierne vľavo, druhý mierne vpravo. Program zostavíme tak, že hodnoty oboch svetelných senzorov porovnáme porovnávacím blokom a pomocou vetvenia programu budeme robota smerovať jedným alebo druhým smerom pri jeho súčasnom pohybe vpred. *Komentár:* treba mať pripravený vhodný zdroj svetla – baterka, čelovka, svetlo z bicykla, alebo stolná lampa. Namontovanie senzorov na robota šikmo je dobré mechanické cvičenie pre začiatočníkov, ktorí už zvládli stavať robota podľa návodu, treba vyvinúť trochu trpezlivosti, cieľavedomosti, ale dajte pozor, aby boli senzory primontované pevne a v žiadnom smere sa neotáčali. V programe sa prvýkrát stretávame s tým, že podmienka vetvenia nie je špecifikovaná v atribútoch bloku, ale je cez vstupný káblik privedená ako logická hodnota. Kontrolná úloha: upraviť program tak, aby hľadal tmavý kút.

**Úloha 6.** *Zadanie:* Naprogramujte robota tak, aby nasledoval človeka. *Riešenie:* Pokúsime sa napísať program tak, aby sa robot pohyboval za človekom alebo robotom, ktorý je pred ním: keď robot človeka dostihne a priblíži na malú vzdialenosť, robot zastane, ak sa človek začne vzďaľovať, robot sa znovu rozbehne a nasleduje ho, ak sa človek priblíži ešte bližšie, robot začne cúvať, ak sa človek stratí z dohľadu, robot sa začne otáčať na mieste, až kým človeka nenájde, potom ho začne znova sledovať. Pokročilejší používatelia sa môžu usilovať, aby pohyby boli plynulejšie a nie trhané. Úloha je názornou ukážkou využitia vnorených vetvení v programe na spracovanie jednotlivých prípadov – intervalov vzdialenosti pred robotom. Robot vystačí aj s jedným ultrazvukovým senzorom. *Komentár:* V tejto úlohe si treba uvedomiť, že zostrojiť robota, ktorý dokonale

sleduje človeka je práca pre profesionálnych výskumníkov, ktorí sa zaoberajú komunikáciou človeka s robotmi. Môžeme používateľom ukázať niekoľko motivačných videí a vysvetliť im, že naším cieľom je skôr vytvoriť jednoduchý model takého robota a porozumieť náročnosti úlohy. Potom sa môžeme spokojne pustiť do práce, ak očakávania nie sú privysoké.




**Obr. 3** Vľavo: Program v NXT-G k úlohe 4 – využíva otáčkový senzor (hore); program v NXT-G k úlohe 5 – hľadanie svetla (dole). Využíva porovnávací blok a logickú hodnotu ako vstup pre vetvenie; vpravo: program ku sledovaniu človeka s viacnásobným vetvením.

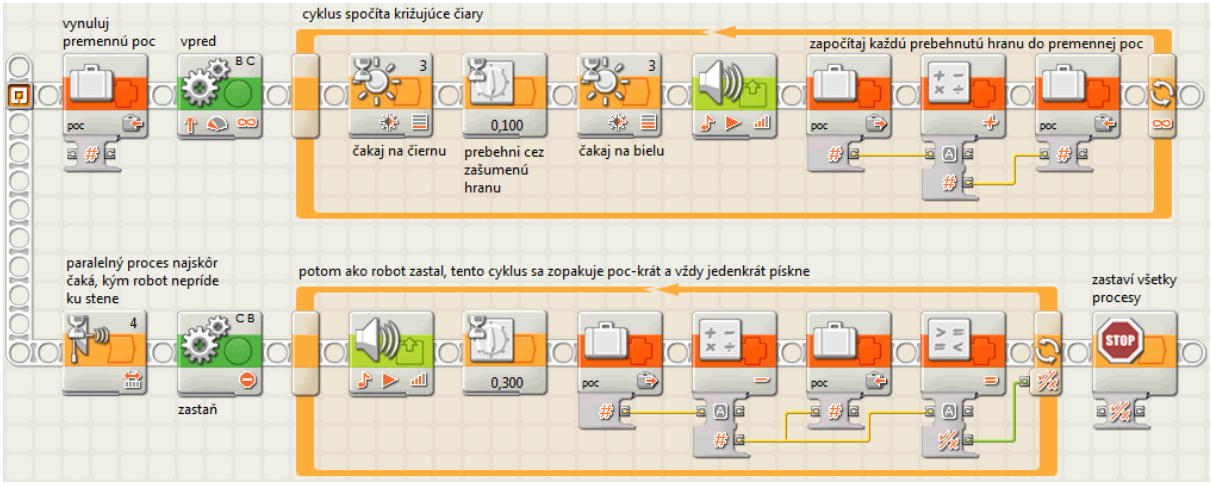



**Obr. 4** Zľava: robot s nárazníkom, svetelným a ultrazvukovým senzorom; situácia pri štarte v úlohe 8; robot v bludisku (námety); diaľkové riadenie pomocou „joysticku“.

**Úloha 7.** *Zadanie:* Zostrojte a naprogramujte robota, ktorý bude sledovať čiernu čiaru na svetlom podklade (napr. na stôl so svetlým povrchom nalepíme 1.5cm širokú čiernu izolačnú pásku, ktorú zakúpime napr. v železiarstve) a obchádzať prekážky umiestnené na čiare. *Riešenie:* Použijeme ľubovoľný robot, ktorý sa dokáže pohybovať vpred a otáčať na mieste a namontujeme naň svetelný senzor tak, aby bol na stredovej osi poháňanej nápravy kolies a od osi otáčania poháňaných kolies bol vzdialený niekoľko centimetrov. Na zaregistrovanie prekážky použijeme ultrazvukový senzor alebo nárazník namontovaný na dotykovom senzore. Algoritmus na sledovanie čiary by mali používatelia objaviť sami – na výber máme z viacerých možností: 1) algoritmus zig-zag, kde sa robot pohybuje smerom vpred a mierne zabáča striedavo vľavo a vpravo tak, aby senzorom prechádzal čiarou raz z jednej raz z druhej strany, t.j. pohyb senzora bude nasledujúci: smerom vľavo: svetlá – tmavá – svetlá – smerom vpravo: svetlá – tmavá – svetlá – smerom vľavo: svetlá... 2) algoritmus udržania sa na čiare: robot sa na čiernej čiare pohybuje vpred, až kým sa nedostane von z čiary – následne sa otáčaním pozrie pár desiatok stupňov vpravo, buď čiaru nájde, alebo sa vráti a hľadá čiaru pár desiatok stupňov vľavo, pre zvýšenie spoľahlivosti môžeme opakovať so zväčšujúcim sa uhlom, kým čiara nie je nájdená. 3) algoritmus sledovania hrany/zjednodušený zig-zig – robot sa pohybuje vpred, striedavo nad čiarou a napravo od čiary, v okamihu keď je nad čiarou z nej hneď začne vychádzať von a v okamihu keď vyjde z čiary sa na ňu začne vracať. Výsledný pohyb je pomerne efektívny a plynulý. Obchádzanie prekážky doprogramujeme až po otestovaní a odladení sledovania čiary (inkrementálny vývoj). Riešime ho buď vetvením v hlavnom cykle (potom však nesmieme použiť na sledovanie čiary čakacie bloky, ale neustále opakujúce sa vetvenie/podmienku), alebo prípadne – ak pracujeme s pokročilými používateľmi – pomocou paralelných procesov, kde jeden sa stará o sledovanie čiary a druhý o obchádzanie prekážky. *Komentár:* sledovanie čiary je základná úloha, ktorou by mal prejsť každý používateľ, ktorý so stavebnicami pracuje. Algoritmus je poučný prítomnosťou emergencie – z jednoduchej senzori-motorickej interakcie vzniká na prvý pohľad neočakávané správanie sledovania čiary. Pripravený pedagóg bude mať pripravený robot, na ktorý môže umiestniť svetelný senzor do rôznej vzdialenosti od osi otáčania kolies a demonštrovať tak závislosť správania od morfológie robota: rovnaký program sa na rôznych robotoch bude správať odlišne. Je dôležité, aby si používateľ uvedomil, že robotika nie je len informatika a programovanie a teda podstatnú úlohu hrá stavba robota, fyzikálne vlastnosti motorov a senzorov a ich umiestnenie. S touto úlohou je možné stráviť viacero niekoľkohodinových stretnutí – naprogramovať robota, ktorý sleduje čiaru pomocou dvoch alebo troch senzorov, robota, ktorý dokáže sledovať čiaru s prudkými, až 90º ostrými zákrutami, robota, ktorý dokáže sledovať čiaru aj pri pohybe po šikmej rampe nahor alebo nadol, alebo robota, ktorý sa dokáže pohybovať v bludisku vytvorenom z čiar a nájsť východ, alebo zostrojiť jeho mapu. Podobne možno požadovať, aby

robot prekážku obchádzal rôznou stranou podľa farby prekážky, S čiernou páskou a svetlým podkladom vystačíte minimálne na pol roka zaujímavých aktivít na krúžku alebo v triede. V každom prípade je táto úloha veľmi vhodná ako prvý väčší projekt, na ktorom si používateľ vyskúša dotiahnuť väčší projekt do úspešného konca – pre mladších používateľov to môže byť prvý podobný zážitok.

**Úloha 8.** *Zadanie:* Zostrojte a naprogramujte robota, ktorý zistí počet čiar pred stenou. Čiary môžu označovať ulice, cez ktoré má robot krížom prejsť, aby sa dostal do cieľa. Pri prechádzaní každej čiary robot vydá zvukový signál. Po dosiahnutí cieľa (prekážky) robot zastaví a vydá zvukový signál toľkokrát, koľko čiar počas jazdy prekrižoval. *Riešenie:* Použijeme ľubovoľný robot, ktorý sa dokáže pohybovať vpred a má namontovaný svetelný senzor smerom nadol a ultrazvukový senzor smerom vpred. Počet prejdených čiar budeme ukladať do číselného „kufríka“, čiže do premennej. V neustále sa opakujúcom cykle budeme reagovať na čiary krátkym zapískaním (napr. tón dĺžky 0,01s) a zvýšením obsahu kufríka o jeden pomocou vetvenia, podmienka ukončenia cyklu bude test na prítomnosť steny. Po skončení cyklu pohyb motorov zastavíme a v ďalšom cykle budeme hodnotu kufríka znižovať o jeden, vydávať zvukový signál a čakať krátky čas medzi signálmi – podmienka ukončenia cyklu bude tento raz vypočítaná logická hodnota určujúca, či obsah kufríka klesol na nulu. *Komentár:* Táto úloha je veľmi užitočná na pochopenie použitia premenných, na jej vyriešenie sú nevyhnutné. Výpočet logickej hodnoty určujúcej ukončenie druhého cyklu zavádza použitie porovnávacieho bloku a logických káblikov. V prvej verzii možno nahradiť zvukový signál po zastavení výpisom počtu čiar na displej, čo vyžaduje a umožňuje pochopiť význam konverzného bloku. Hoci v tejto úlohe vystačíme s jedinou preddefinovanou číselnou premennou **`Number1`**, vhodnejšie je pomocou menu *Edit – Define Variables* zadefinovať novú premennú s iným názvom, napr. **`pocet`**. Typický problém, ku ktorému dochádza pri riešení úlohy: pri započítavaní čiary dochádza k viacnásobnému započítaniu, keďže cyklus s vetvením sa zopakuje niekoľkokrát, pokiaľ robot opustí čiaru a pokračuje nasledujúcou svetlou oblasťou. Predísť tomu môžeme buď tak, že po započítaní každej čiary pomocou čakacieho bloku necháme robota automaticky sa pohybovať vpred po dobu približne pol sekundy, aby opustil priestor čiary, prípadne použijeme čakací blok so svetelným senzorom. S tým súvisí aj voľba „wait for completion“ zvukového bloku. Ďalší zaujímavý aspekt je overiť, či program bude fungovať správne aj vtedy, keď sa pred stenou nenachádza žiadna čiara – či po zastavení zapíska raz, či bude pískať do nekonečna, alebo (správne) nezapíska ani raz. Úprava si pravdepodobne vyžiada doplňujúcu podmienku. Pokročilí používatelia môžu úlohu riešiť pomocou paralelných procesov, ako ukazuje obrázok 5.

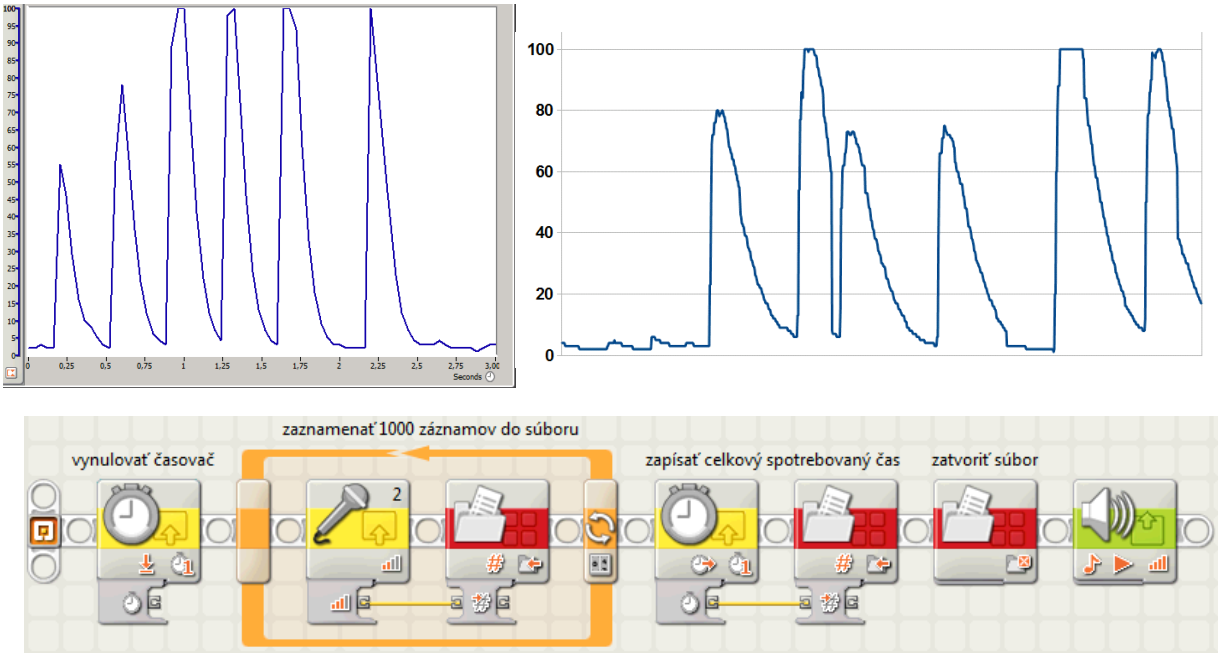



**Obr. 5** Riešenie k 8. úlohe – počítanie čiar – využitie kufríkov (premenných) a paralelných procesov.

**Úloha 9.** *Zadanie:* Zostrojte kalkulačku pomocou NXT kocky. *Riešenie:* Na kocku NXT pripojíme niekoľko dotykových senzorov, ktoré budú slúžiť ako tlačidlá na zadávanie vstupných hodnôt a matematických operátorov. Kalkulačka bude s používateľom komunikovať prostredníctvom displeja a prípadne zvukových signálov. Po odštartovaní programu používateľ pomocou tlačidiel (senzorov a tlačidiel priamo na kocke NXT) zadá dve hodnoty a matematickú operáciu, kalkulačka vypočíta a zobrazí výsledok. *Komentár:* V tejto úlohe si používateľ dôkladne precvičí prácu s premennými a displejom, kde môže využiť blok na spájanie textových reťazcov, musí porozumieť nastavenie 'Clear' v zobrazovacom bloku a vyskúša si možnosť zadať viac ako dve vetvy podmienky po vypnutí režimu vetvenia 'flat view'. Úlohu môžeme zaradiť, keď sa chceme zamerať viac na programovanie alebo keď nemáme čas na stavbu robota. Na prácu s tlačidlami je vhodné zadefinovať vlastné nové bloky. Nemali by sme sa snažiť napísať celý program naraz, ale budovať ho postupne, avšak premyslene (inkrementálny vývoj).

**Úloha 10.** *Zadanie:* Zostrojte robota, ktorý sa bude otáčať na mieste rôznou rýchlosťou podľa toho, ako rýchlo používateľ rytmicky tlieska. *Riešenie:* Použijeme ľubovoľného robota, ktorý sa vie otáčať na mieste, pripojíme zvukový senzor. Po spustení programu sa robot začne veľmi pomaly otáčať, v cykle budeme pomocou časovača merať interval medzi dvoma zvukovými vstupnými impulzmi. Nameraný údaj preškálujeme a otočíme, aby sme vypočítali číselnú hodnotu, ktorú privedieme na prípojku 'power' pohybového bloku. Do programu môžeme doplniť druhý proces, ktorý bude rýchlosť robota plynule a veľmi pozvoľne znižovať tak, aby po ukončení rytmického tlieskania. *Komentár:* Okrem toho, že si používateľ vyskúša činnosť časovača, lepšie sa zoznámi s povahou snímania pomocou zvukového senzora. Jedno tlesknutie vyvolá zvukový impulz, ktorý je na senzore prítomný určitú dobu (okolo 100 ms) a preto je potrebné zabezpečiť, aby sa cyklus neopakoval častejšie, napríklad pomocou čakacieho bloku. Na lepšie pochopenie fenoménu môžeme zaznamenať hodnoty zvukového senzora do datalogu a vizualizovať ho v grafe pomocou programu NXT Data Logging a odsledovať dĺžku pulzov

pri tlieskaní. Takto okamžite získame záznam s maximálnou frekvenciou 25 vzoriek za sekundu. Podrobnejší záznam získame, ak napíšeme program zapisujúci údaje do súboru. V takom prípade je frekvencia až okolo 345 vzoriek za sekundu. Všimneme si tiež miesta, keď došlo k premodulovaniu signálu a počas niekoľkých desiatok milisekúnd senzor vracal hodnotu 100 dB.




**Obr. 6** Porovnanie presnosti merania bežným datalogom (vľavo hore) a pomocou zápisu do súboru (vpravo hore a dolu).

## 5 NÁMETY NA ĎALŠIE ÚLOHY A POKROČILÉ TECHNIKY V NXT-G

Až po zvládnutí funkcionality programovacieho jazyka sa otvárajú dvere do sveta skutočného konštrukcionizmu a vzrušujúcich poznávacích projektov. V tejto stati uvádzame niekoľko námetov na väčšie projekty, mnoho ďalších je k dispozícii na [4].

*Námet 1:* Zostrojte a naprogramujte robota, ktorý nájde východ z bludiska. Bludisko je tvorené pravouhlou sieťou ciest reprezentovaných čiernou páskou na bielom podklade. Bludisko môže obsahovať križovatky všetkých druhov i slepé uličky. Východ z bludiska môžeme označiť predmetom, ktorý robot rozpozná pomocou ultrazvukového senzora. Po celý čas smie robot jazdiť iba pozdĺž čiernej pásky – prejsť krížom cez biele územie nie je dovolené. V jednoduchšej verzii bludisko neobsahuje okruhy (na to isté miesto sa nedá vrátiť z iného smeru), v mierne náročnejšej verzii obsahuje cykly, ale zo štartu do cieľa sa dá dostať pravidlom pravej, resp. ľavej ruky. V najnáročnejšej verzii robota umiestnime do stredu bludiska, kde by pravidlo pravej i ľavej ruky viedlo k zacykleniu. Riešením môže byť náhodný pohyb a v prípade pokročilých používateľov mapovanie ihriska.

*Námet 2:* Animácia fyzikálneho deja na displeji NXT. Ku kocke pripojíme nejaký jednoduchý stroj – páku, alebo kladku a na displeji animujeme stav stroja, priebeh experimentu môžeme riadiť tlačidlami na NXT alebo dotykovými senzormi. Na nakreslenie obrázkov, ktoré je možné zobraziť na displeji zobrazovacím blokom možno použiť program nxtRICeditV2 [11].

*Námet 3:* Zostrojte a naprogramujte triediacu linku, ktorá rozpoznáva predmety podľa ich tvaru. Z gumených pásov zostrojíme linku, po ktorej sa automaticky posúvajú rozličné predmety. Ultrazvukovým senzorom umiestneným na bočnej alebo hornej strane linky meriame profil predmetu. Pomocou svetelného senzora a svetla (fotobunka) štartujeme záznam ultrazvukovým senzorom, ktorý trvá určitú dobu. Niekoľko rôznych predmetov zosnímame viackrát a ich profily uložíme do súborov. Súbory prenesieme na PC a pomocou tabuľkového editora po odfiltrovaní extrémnych hodnôt vypočítame priemer pre každý druh predmetu. Napokon naprogramujeme triedičku, ktorá počas snímania predmetu porovnáva údaje so priemernými profilmi a počíta súčet kvadratických chyb pre každý druh natrénovaného predmetu. Na základe toho vyhodnotí o ktorý predmet išlo a zobrazí koeficient istoty (podobnosti).

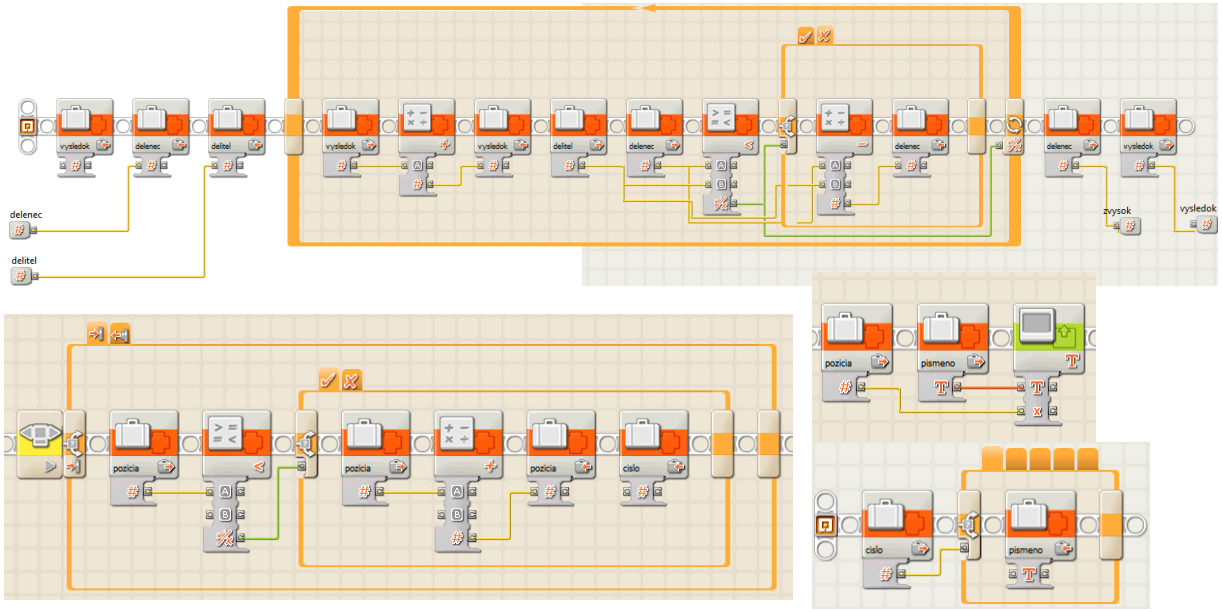
*Námet 4:* Diaľkové riadenie robota inou kockou. Diaľkový ovládač môžeme vytvoriť buď pomocou dotykových senzorov, alebo pomocou otáčkových senzorov zabudovaných v motoroch, ktoré fungujú ako joystick – jeden sníma dopredné naklonenie a je umiestnený v ráme, ktorý sa celý nakláňa, pričom druhý sníma bočné naklonenie rámu, pozri obrázok. Diaľkový ovládač komunikuje s robotom posielaním správ. Ovládaný robot na prijatie každej správy zareaguje príslušným pohybom – vpred, vzad, vľavo, vpravo, pričom rýchlosť pohybu je určená stupňom naklonenia „joysticku“. Otvára sa tak zaujímavý problém – ako odoslať viacero hodnôt z ovládača do ovládanej kocky súčasne? Potrebujeme odoslať smer pohybu a stupeň naklonenia. Prirodzené riešenie je využitím rozličných schránok (mailbox), avšak zamyslime sa, ako by sme úlohu vyriešili použitím jedinej schránky. Jednou možnosťou by bolo vyslať dve hodnoty za sebou, to však naráža na možnú stratu prvej poslanej hodnoty, keďže posledná hodnota odoslaná do schránky vždy nenávratne prepíše predchádzajúcu. Iným riešením je skombinovať viacero hodnôt do jednej pomocou súčiniteľa `vysledok = hodnotaA * sucinitel + hodnotaB`, pričom sucinitel musi byt väčší ako maximálna hodnotaB, ktorú chceme vyslať. Spätne môžeme jednotlivé

hodnoty získať jednoducho: **`hodnotaA = vysledok div sucinitel, hodnotaB = vysledok mod sucinitel`** (operátory celočíselného delenia div, mod). V tom spočíva ale výzva: jazyk NXT-G pozná násobenie i delenie, ale nepozná celočíselné delenie a zvyšok po delení. A práve na tento účel môžeme vytvoriť vlastný blok – postupným odčítavaním. Na tomto príklade vidieť, že používateľom definované bloky môžu mať aj svoje vstupné a výstupné argumenty. Tie sa potom v programe zadávajú rovnakým spôsobom ako argumenty ostatných blokov.

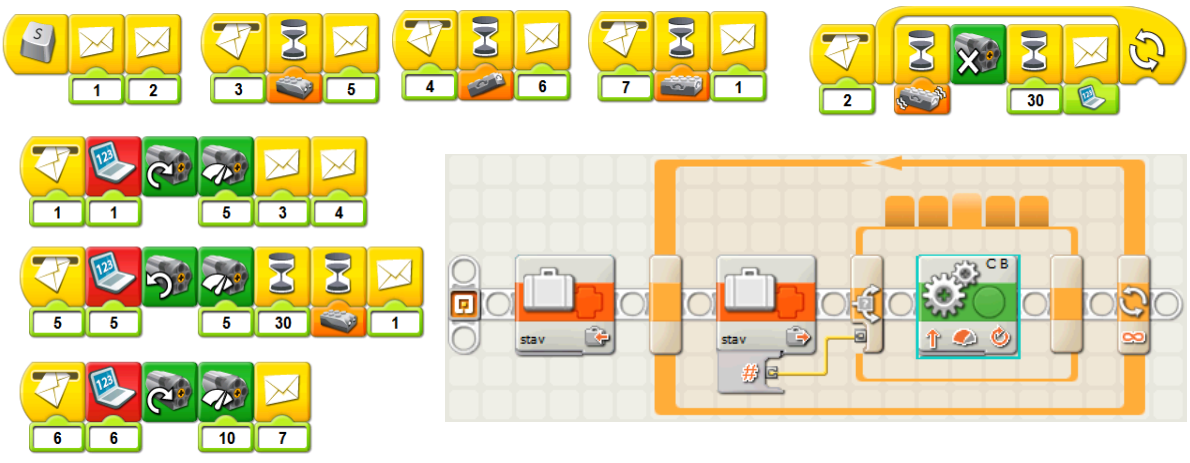
*Námet 5:* Jednoriadkový editor. Zostrojte program, ktorý umožňuje editovať text na displeji NXT pomocou gumovej klávesnice: šípky posúvajú kurzor, enter mení znak nad kurzorom. Keďže jazyk NXT- G nemá aritmetiku na úrovni znakových kódov a ani nevie priamo konvertovať číselné znakové kódy (napr. v kódovaní ASCII) na znaky, potrebujeme si takýto blok vytvoriť, obrázok 7 vpravo dolu. Potom stačí naprogramovať pohyb šípkami, zobrazovanie znaku na pozíciu. Pokročilí používatelia naprogramujú aj zobrazovanie kurzora na aktuálnej pozícii a budú skladať výsledný text z jednotlivých písmen, aby ho mohli v každom okamihu celý znova prekresliť (vzhľadom na to, že na displej možno iba dokresľovať, nie však mazať bez toho, aby sme nezmazali všetko).

## 6 VYSOKÝ STROP

Stavebnice NXT ponúkajú široké spektrum rôznych spôsobov konštrukcie, programovania a použitých postupov, nízku štartovnú latku, keďže začať môže každý používateľ bez akýchkoľvek predchádzajúcich vedomostí a vysoký strop – prakticky neohraničené možnosti zložitosti. Medzi rozšírenia patria aj rozličné doplnkové senzory – kompas, akcellerometer, gyroskop a senzor na hľadanie infračervenej futbalovej lopty (IR Seeker2). Využijeme ho na vyučovanie programovania aj bez toho, aby sme vysielali tím na súťaž. Senzor oznamuje hodnotu 1-9 podľa toho, ktorým smerom „vidí“ loptu. Vhodným cvičením je napísať program, ktorý túto hodnotu spracuje a prepočíta na smer otáčania (steering) pre pohybový blok tak, aby sa robot pohyboval za loptou. Na to si používateľ musí uvedomiť vstupný a výstupný interval prevodu a navrhnúť funkciu, ktorá hodnotu správne prepočíta.




**Obr. 7** Delenie so zvyškom (hore), úryvky z jednoriadkového editora (dolu).




**Obr. 8** Programovanie stavovými automatmi WeDo (hore) a NXT-G (dolu).

Inou zaujímavou oblasťou je implementácia stavových automatov v jazyku NXT-G. Program navrhnutý ako stavový automat je rozčlenený do množstva nezávislých častí – stavov, v ktorých vykonáva nejakú konkrétnu činnosť. Za istých okolností prejde do iného stavu a začne vykonávať inú činnosť, až kým sa nedostane znovu do iného stavu, resp. na koniec do konečného stavu, keď program skončí. Stav reprezentujeme číselnou alebo textovou premennou. Program potom pozostáva z jedného, typicky nekonečného cyklu a širokého vetvenia, pričom každá vetva spracuje príslušný stav. Obrázok 8 vpravo dolu znázorňuje vetvenie podľa aktuálneho stavu. Do pozornosti treba dať aj súvis tejto myšlienky so štruktúrou programov v programovacom jazyku pre stavebnice WeDo, ktoré sú určené pre nižšie stupne. Zo stavebníc WeDo sa nedajú stavať autonómne modely, iba modely neustále riadené z PC a pripojené pevným spojením cez USB kábel. Jazyk WeDo neobsahuje vnorené cykly a vôbec neobsahuje vetvenie, obsahuje ale zaujímavý spôsob, ako dosiahnuť zložitejšiu funkcionalitu: prostredníctvom posielania správ, čiže využitím stavových automatov. Obrázok 8 vľavo hore znázorňuje typickú situáciu.

## ZÁVER

V článku sa zaoberáme využitím robotických programovateľných stavebníc vo vyučovaní programovania na základných a stredných školách. Opísali sme rozličné aspekty a predpoklady použitia stavebníc pre tento účel. Predstavili sme existujúce materiály, vhodné postupy a úlohy pre začiatočníkov i pokročilých a sadu úloh, ktoré sme viacnásobne úspešne otestovali na školeniach učiteľov a na krúžkoch. Rozoberáme pokročilé námety, témy a techniky programovania a uvádzame príklady projektov, ktoré vytvorili žiaci. Jednotlivé úlohy sa dajú použiť v rozličných verziách podľa požadovanej náročnosti a odborných skúseností skupiny. V opisoch úloh uvádzame jednotlivé varianty, rozšírenia a zjednodušenia. Z filozofického a didaktického hľadiska náš prístup otvára širšiu diskusiu o spôsobe využitia stavebníc v edukačnej praxi. Jeden možný spôsob spočíva v striktnej filozofii konštrukcionizmu, čiže ponechania detí poobjavovať úplne všetko, vrátane funkcii jednotlivých programovacích blokov, ich prípojok, významov premenných, vetvení, cyklov, atď. na základe ich vlastnej tvorivej činnosti inšpirovanej vlastnými nápadmi a predovšetkým bez jednoznačne stanoveného plánu kurikula, to znamená dieťa sa vydáva tým smerom a objavuje také princípy a koncepty, ktoré sú potrebné na zvládnutie úlohy, ktorú si samé stanovilo. Druhý spôsob, ktorý obhajujeme v článku, vychádza z konkrétnejšieho plánu. Čo chceme, aby si deti (resp. účastníci kurzu) osvojili, ktoré princípy a pojmy zvládli? Za tým účelom im poskytneme konkrétne úlohy (resp. množinu alternatívnych úloh, z ktorých si vyberajú) a navádzame ich tak k objavovaniu poznatkov svojou konštrukcionistickou hrou. Na jednej strane je to negatívne, pretože my určujeme, čo dieťa bude objavovať a v akom poradí, na druhej strane, v prípade, že cieľom kurzu je vyučovanie konkrétne stanoveného kurikula, považujeme tento spôsob za nevyhnutný. Fundamentalistický konštrukcionizmus považujeme v tomto kontexte za priveľký luxus a neefektívnu metódu bez možnosti dosiahnutia stanovených pedagogických cieľov. Na druhej strane, po zvládnutí základných princípov a pojmov sa otvárajú dvere do ozajstného otvoreného sveta konštrukcionizmu - interdisciplinárnych tvorivých projektov, ktoré už nepotrebujú presné smerovanie a postupnosti predpripravených úloh.

## LITERATÚRA


[1] Kratochvílová, E.: Pegagogika voľného času - Výchova v čase mimo vyučovania v pedagogickej teórii a v praxi. TYPI Universitas Tyrnaviensis, 201, ISBN 978-80-8082-330-6.
[2] Průcha, J : Moderní pedagogika, čtvrté, 4. vyd. Portál 2009. ISBN 978-80-7367-503-5.
[3] Five Minute Bot. Nxtprograms.com, Fun Projects for your LEGO MINDSTORMS NXT. Dostupné online: www.nxtprograms.com/five_minute_bot/index.html pristupované: 1.2.2012.
[4] Stavebnice LEGO MINDSTORMS NXT vo vyučovaní. Robotika.SK. Dostupné online: robotika.sk/nxt, pristupované: 1.2.2012.
[5] Widger, R. : NXT User Guide and ICT Curriculum Scheme of Work. Dacta Ltd, United Kingdom, 2006.
[6] Nxtprograms.com, Fun Projects for your LEGO MINDSTORMS NXT. Dostupné online: www.nxtprograms.com/five_minute_bot/index.html pristupované: 1.2.2012.
[7] Lücking, J.: LEGO MINDSTRORMS EDUCATION NXT – Eine Einfürung für die Schule, 2009.
[8] Lücking, J.: LEGO MINDSTRORMS EDUCATION NXT – Unterrichtsmaterialien für fortgeschrittene, 2009.
[9] LogIT World. DCP Microdevelopments. Dostupné online: www.logitworld.com, pristupované 1.2.2012.
[10] NXT Sensor Adapter. Vernier. Dostupné online: www.vernier.com/products/interfaces/bta-nxt, pristupované: 1.2.2012.
[11] Referenčná príručka ku grafickému jazyku NXT-G, dostupné online: http://robotika.sk/events/09KurzNXT/nxtg.pdf resp. http://robotika.sk/events/09KurzNXT/nxtg2.pdf, 2009.
[12] Dreier A.: NxtRICeditV2, Tool for creation or modification of RIC files for the Mindstorms NXT. Dostupné online: http://ric.dreier-privat.de/Docu/index.htm pristupované 1.2.2012.


## AUTOR


**PETROVIČ, PAVEL**

Katedra aplikovanej informatiky, Fakulta matematiky, fyziky a informatiky, Univerzita Komenského Bratislava.
Mlynská dolina, 842 48 Bratislava
ppetrovic@acm.org