**Univerzita Pardubice**

**Fakulta ekonomicko-správní**
**Ústav systémového inženýrství a informatiky**

# Tvorba prostředí pro robotické úlohy

**Markéta Jelínková**

**Bakalářská práce**
**2013**

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: **2012/2013**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Markéta Jelínková** |
| Osobní číslo: | **E10548** |
| Studijní program: | **B6209 Systémové inženýrství a informatika** |
| Studijní obor: | **Informatika ve veřejné správě** |
| Název tématu: | **Tvorba prostředí pro robotické úlohy** |
| Zadávající katedra: | **Ústav systémového inženýrství a informatiky** |

Zásady pro vypracování:

Cílem práce je tvorba prostředí, ve kterém se budou roboti pohybovat s následnou tvorbou úloh, které budou využity v předmětu Základy algoritmizace

1) Stavebnice Lego Mindstorm
2) Složení a tvorba prostředí pro robotické stavebnice
3) Tvorba úloh a jejich následná implementace pro robotické stavebnice

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

**TAUFER I. Algoritmy a algoritmizace - vývojové diagramy. Pardubice: Univerzita Pardubice, 2009. ISBN 978-80-7395-182-5.**
**PŠENČÍKOVÁ, J. Algoritmizace. 1. vyd. Kralice na Hané: Computer Media, 2007. ISBN 978-80-7402-034-6.**
**KNUTH, D. E., ERVIN, D. Umění programování I. Brno: Computer Press, 2008. ISBN 9788025120255.**

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Ústav systémového inženýrství a informatiky

Datum zadání bakalářské práce: **1. října 2012**
Termín odevzdání bakalářské práce: **30. dubna 2013**

L.S.

doc. Ing. Renáta Myšková, Ph.D.
děkanka

prof. Ing. Jan Čapek, CSc.
vedoucí ústavu

V Pardubicích dne 1. října 2012

## PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 30. 4. 2013 Jelínková Markéta

**PODĚKOVÁNÍ**:

Tímto bych ráda poděkovala svému vedoucímu práce Ing. Janu Panušovi Ph.D. za jeho odbornou pomoc, cenné rady a poskytnuté materiály, které mi pomohly při zpracování bakalářské práce.

## ANOTACE

*Tato práce bude sloužit studentům pro pochopení teorie předmětu Základy algoritmizace. Studenti se podle ní naučí pracovat s LEGO robotickými stavebnicemi a s programem Enchanting. Práce by měla sloužit jako předloha pro vypracování zadaných úloh. Tištěná verze je doplněna o CD s ukázkami řešených příkladů.*

## KLÍČOVÁ SLOVA

*Algoritmizace, robot, Lego Mindstorms, Enchanting*

## TITLE

Formation of environment for robotics exercise

## ANNOTATION

*This thesis will serve students for understanding the theory of the course Fundamentals of Algorithmization. Students will learn to work with LEGO robotic construction sets and with the program Enchanting. Thesis should serve as a model for the development of assignments. The printed version is complemented by a CD with examples of tasks.*

## KEYWORDS

*Algorithmization, robot, Lego Mindstorms, Enchanting*

# OBSAH

## SEZNAM ILUSTRACÍ

## SEZNAM ZKRATEK A ZNAČEK

| | |
|---|---|
| ČSN | Česká státní norma |
| ISO | Organization for Standardization |
| VD | Vývojový diagram |
| USB | Universal serial bus |
| OVP | Orientace v prostoru |

# ÚVOD

Cílem práce je tvorba úloh pro roboty, kteří se budou pohybovat v prostředí. Vytvořené úlohy budou využity v předmětu Základy algoritmizace, kde budou sloužit jako pomůcka pro studenty při řešení úloh.

Na začátku této bakalářské práce bude pojednáváno o základech algoritmizace. Budou popsány a zobrazeny základní prvky vývojových diagramů, které jsou v předmětu Základy algoritmizace nejvíce využívány.

V druhé kapitole bude charakterizována stavebnice Lego Mindstorms. Blíže bude popsána samotná řídící jednotka NXT, vstupní a výstupní moduly, které stavebnice obsahuje. Dále bude pokračovat popis sady na vytvoření „Zeleného města“, pro které jsou nabídnuty úlohy pro robotické stavebnice přímo od firmy Lego.

V další kapitole bude stručně popsán program Enchanting, ve kterém budou všechny úlohy vytvořeny. Existuje více aplikací pro práci s roboty, tento program byl však zvolen pro svou jednoduchost a přehlednost.

Poslední kapitola bude obsahovat jednotlivé vytvořené úlohy pro sestavené roboty. V kapitole budou postupně úlohy podle stupně obtížnosti na sestavení a vytvoření kódu. Na začátku bude seznámení s robotem a jeho základními funkcemi. Dále budou následovat souhrnné robotické úlohy.

# 1 ZÁKLADY ALGORITMIZACE

## 1.1 Algoritmy

S algoritmy se lidé setkávají v běžném životě, aniž by si to uvědomovali. Stačí, když se rozmýšlejí, zda přejít či nepřejít silnici. Dokonce i obyčejný kuchařský recept se dá nazvat algoritmem. Algoritmus je totiž určitý postup či návod, který vyřeší danou úlohu. Skládá ze z konkrétních kroků, které směřují k cíli. Jako algoritmus se můžeme počítat i například cesta přes město z bodu A do bodu B. Taková cesta má stanovený jasný cíl. Jejím začátkem je určen bod A, a jejím koncem bod B. Cesta jako taková, může být ovšem různá. Záleží na člověku, zda zvolí tu či onu ulici. V každém případě, výsledek bude vždy stejný. Na takovém příkladě je vidět, že pro jednu úlohu, může být i více správných algoritmů. To vše záleží na tom, kdo algoritmus vytváří a čím se řídí [1].

S algoritmy pracují především programátoři. Vytvořením správného algoritmu si mohou hodně usnadnit práci. Především se vyhnou určitým chybám, které by mohly vzniknout při programování. Je pět základních principů, které musí každý algoritmus splňovat. Pokud nějaký z těchto principů algoritmus nesplňuje, vyskytne se v programu chyba.

**Vlastnosti algoritmů dle [1], [4]:**

- Konečnost – každý algoritmus musí mít začátek, konec a konečný počet kroků. Vždy se musí dostat ze začátku na konec po konečném počtu kroků.
- Obecnost – či hromadnost, univerzálnost nebo i masovost. Algoritmus by měl být tak obecný, aby mohl řešit úlohy s jinými vstupními parametry. Neměl by např. sčítat „4 + 5“, ale měl by spočítat součet jakýchkoli dvou čísel, které budou zadány.
- Determinovanost – každý krok musí být přesný, srozumitelný a jednoznačný. V každém kroku musí být jasné, jaký je jeho následující krok.
- Výstup – každý algoritmus má alespoň jeden výstup. Nějaký výsledek, který je řešením daného úkolu.
- Elementárnost – kroky algoritmu musí být jednoduché, snadno realizovatelné.

Mezi jazyky algoritmizace patří vývojové diagramy, strukturogramy, rozhodovací tabulky a programovací jazyky. Jelikož je oblast algoritmizace rozsáhlá, zmíním se pouze v základech o vývojových diagramech, které se používají v předmětu Základy algoritmizace [1].

## 1.2 Vývojové diagramy

Jak již bylo řečeno, vývojové diagramy jsou formou jazyka algoritmizace. V předmětu Základy algoritmizace se nejvíce používají, proto v této práci nebudou dále rozvedeny ostatní jazyky. V následujících řádcích jsou popsané základní prvky vývojových diagramů, které je třeba znát. V další části se již přistoupí ke stavebnici Lego Mindstorms, se kterou se bude nově v předmětu Základy algoritmizace pracovat [1].

Vývojové diagramy představují přesně definované symbolické značky, které jsou technicky normovatelné. Tyto grafické značky slouží k jednoznačnému zobrazení a je třeba dodržet pravidla, která jsou dána k jejich používání [1]. Pravidla jsou definována v české státní normě z roku 1995 – ČSN ISO 5807 „Zpracování informací. Dokumentační symboly a konvence pro vývojové diagramy toku dat, programu a systému, síťové diagramy programu a diagramy zdrojů systému“ [2]. Vývojový diagram je základ pro popis výpočetního algoritmu a jak již bylo řečeno, je to i podklad pro programátora při vytváření programu [1].

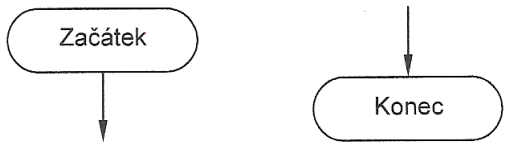
### 1.2.1 Symboly vývojových diagramů

Symboly vývojových diagramů jsou grafické značky, které jsou přesně stanovené. Jak tvarem, tak významem. Do značek jsou vpisovány operace. Není však stanoven způsob jejich zápisu. Pro dobrou čitelnost jak programátorem, tak i čtenářem neznalým programovacího jazyka, je lepší použít jednoduchý text a matematické značky [1].

Dále bude znázorněno několik nejvíce používaných symbolů vývojových diagramů.

**Mezní značky**

Mezní značky jsou Začátek a Konec. Je to jeden vstup z vnějšího prostředí do programu či podprogramu a jeden výstup z vnějšího prostředí do programu či podprogramu. Do značky Začátek nesmí vstupovat žádná jiná značka, a vystupuje z ní pouze jedna hrana. Do značky Konec vstupuje pouze právě jedna hrana, a nesmí z ní vystupovat žádná jiná. Po značce Konec již nesmí být žádná další jiná značka [3].

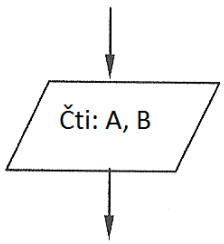



**Obrázek 1: VD - Mezní značky**

*Zdroj: [1]*

### Vstup a Výstup dat

Vstup nebo Výstup se řadí do sekvenčních bloků. Takové značky musí být uvnitř vývojového diagramu. Značka Vstup nebo Výstup dat slouží ke komunikaci s počítačem, kdy je třeba z vnějšího prostředí zadat nějaká data, či získat data pomocí zobrazení [3].

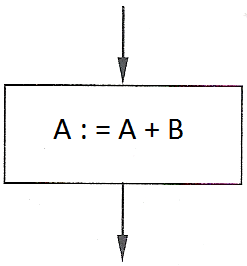



**Obrázek 2: VD - Vstup dat**

*Zdroj:upraveno podle [1]*

### Zpracování

Značka Zpracování udává činnost programu. Zde se provede operace s daty nebo jejich transformace. Může obsahovat i více instrukcí, které musí být řádně popsány. Zpracování musí mít právě jeden vstup a jeden výstup [3].

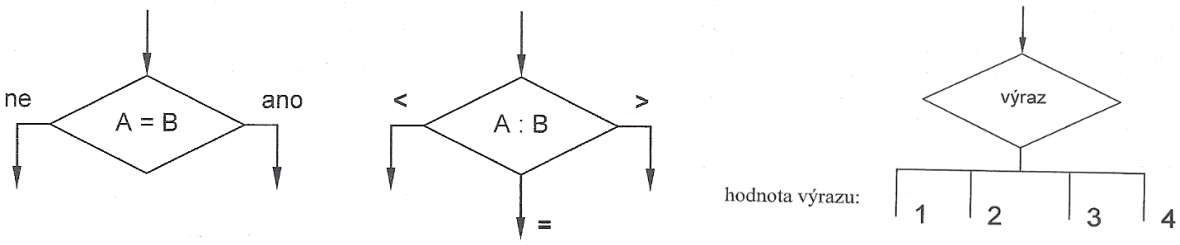



**Obrázek 3: VD – Zpracování**

*Zdroj:upraveno podle [1]*

### Větvení

Ne vždy může algoritmus pokračovat přímou cestou. Často vznikají otázky, u kterých je třeba se rozhodnout. Tato značka má funkci buď rozhodovací, nebo přepínací. Podle vyhodnocení podmínky se dále určuje cesta. Pokud je podmínka splněna, „půjde“ algoritmus jednou větví, pokud podmínka splněna není, bude pokračovat algoritmus druhou větví. Počet větví není omezen pouze na dvě, může jich být i více. Pokud je výsledných větví více, rozhoduje se podle příslušné hodnoty. Rozhodovací značka má vždy pouze jeden vstup [3].

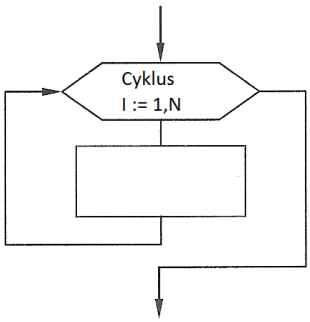



**Obrázek 4: VD – Větvení**

*Zdroj: upraveno podle [1]*

### Příprava

Přípravná značka označuje fázi programu, kde je vytvořen cyklus. Ve vývojových diagramech je více druhů cyklů, tenhle ale znázorňuje cyklus s pevným počtem opakování. Značka má dva vstupy – sekvenční a návratový, a jeden výstup [3].




**Obrázek 5: VD – Větvení**

*Zdroj:upraveno podle [1]*

### Spojka

Posledním častým symbolem je značka Spojka, která se používá u vývojových diagramů, které nemohli být nakresleny souvisle (na jeden papír). Symboly na Spojce musí být stejně označeny, aby nemohlo dojít k záměně. Spojka má pouze jeden vstup nebo pouze jeden výstup, podle toho, v které části vývojového diagramu se nachází [3].


**Obrázek 6: VD – Spojka**

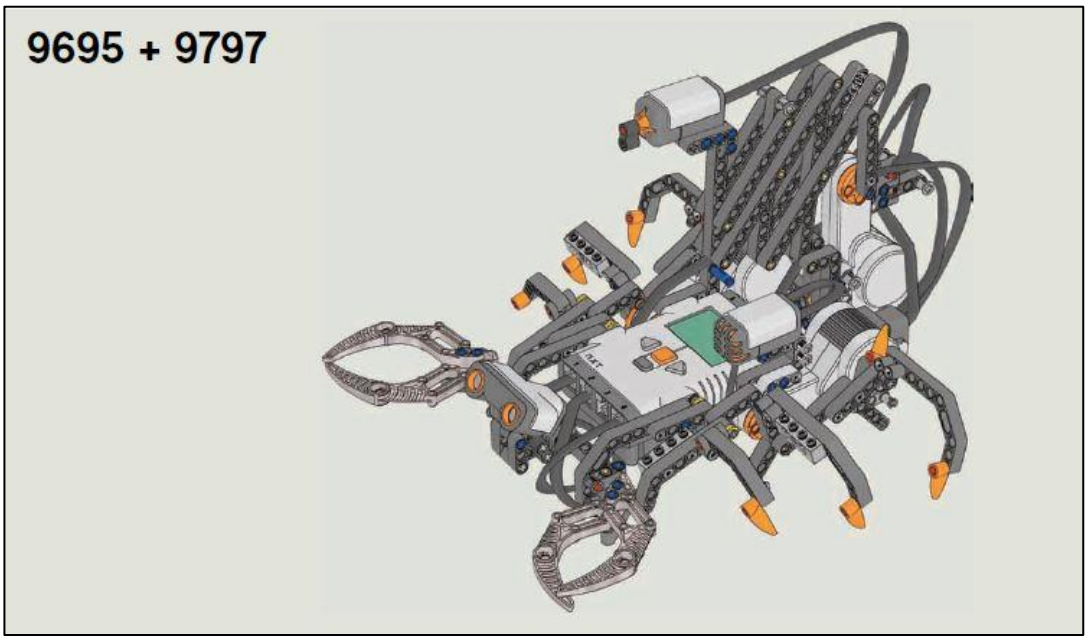
*Zdroj: [1]*

# 2 STAVEBNICE LEGO MINDSTORMS

Fakulta ekonomicko-správní Univerzity Pardubice zakoupila pro předmět Základy algoritmizace velké sety stavebnice Lego Mindstorms. Základní vzdělávací sadu (označení výrobcem pro sadu je číslo 9797), celou sadu náhradních dílů (9695) a sadu Zelené město (9594) [5].

Základní sada obsahuje řídící jednotku NXT, 3 servomotory, ultrazvukový senzor, zvukový senzor, světelný senzor a dva dotykové senzory. Dále obsahuje dobíjecí baterie, nabíječku a kabely o délkách 20 cm, 35 cm a 50 cm. V poslední řadě lze v setu nalézt červený a modrý míček, cd se softwarem, návod na sestavení robota a díly potřebné ke stavbě robota. Celkem stavebnice obsahuje 431 dílů [5], [6].

Rozšiřující sada obsahuje speciální díly, díky kterým lze sestavit větší množství různých stavebnic. Od tanků až po škorpiona či robotickou ruku. Mezi speciální díly stavebnice patří například pásy, unikátní konektory, šnekové převodovky, konstrukční prvky, nosníky náprav a další. Sada obsahuje celkem 817 jednotlivých dílků [5].




**Obrázek 7: Lego stavebnice 9695 + 9797 – Škorpion**

*Zdroj: [7]*

## 2.1 Robot

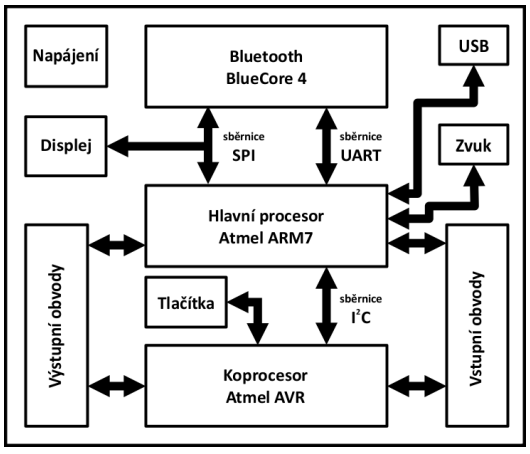
Jak již jistě každý zná ze základní školy, slovo „robot“ bylo poprvé použito v díle R.U.R. spisovatelem Karlem Čapkem. Pod tímto názvem si lze představit stroj, který něco vykonává, který zvládá činnosti jako lidé. V dnešní době roboti často mají i podobný tvar jako lidé. Dokonce i nová stavebnice od Lega s označením 8547 má tvar podoby člověka.

Aby mohl robot vykonávat činnosti, musí mít funkční části. Části, kde je nahrán program, který robota řídí a moduly s kterými vykonává jednotlivé operace. Nejprve se seznámíme s řídící jednotkou, která je mozkem robota Lego Mindstorms [8].

### 2.1.1 Řídící jednotka NXT

Řídící jednotka je největším modulem stavebnice. Obsahuje v sobě mikropočítač, který vykonává procesy a řídí ostatní napojené moduly. Do řídící jednotky se nahraje program přes USB kabel, nebo se může ovládat přes Bluetooth. Většinou se ale spíše používá USB kabel. Pokud je zapnutý Bluetooth, můžou řídící jednotky komunikovat i navzájem mezi sebou. Vzhledem k tomu, jak řídící jednotka vypadá, bývá nazývána Řídící kostkou NXT [8].

Na Obrázek 8 je vidět, jak je řídící jednotka sestavena. Je složena ze specializovaných modulů, které spolu komunikují pomocí několika sběrnic [8]. Hlavním „mozkem“ řídící jednotky je 32bitový mikroprocesor Atmel ARM7. Jádro tohoto procesoru navrhla firma Acorn, která přinesla jednotící prvek pro jádra a tím značně rozšířila své výrobky. V té době se společnosti právě snažily zvýšit výkon mikroprocesorů. Firma Acorn navrhla několik jader, z nichž právě nejrozšířenější je ARM7 [9].



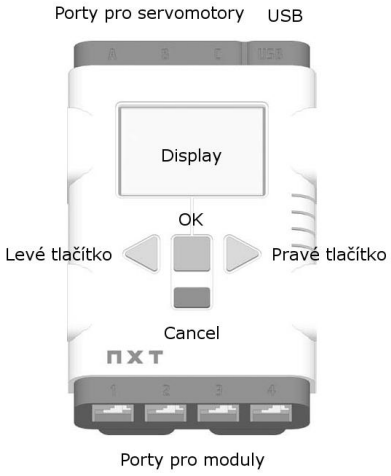

**Obrázek 8: Vnitřní propojení modulů řídící jednotky**

*Zdroj: [8]*

Jak již bylo řečeno, aby kostka něco dělala, je třeba do ní nejprve nahrát data a instrukce, které se mají zpracovávat a vykonávat. A aby vůbec komunikovala, je v ní od tvůrce nahrán vnitřní software, tzv. firmware. Pro správnou funkčnost ho je třeba aktualizovat na nejnovější verzi.

Kostka je dále vybavena 256 KB flash paměti, která není dále rozšiřitelná. Na Obrázek 9 je možné spatřit černobílý maticový displej o velikosti 100 x 64 pixelů a rozměrech 40,6 x 26 mm. Dále má čtyři funkční tlačítka – posun Vlevo a Vpravo, OK a Zrušení. Tlačítka je možné i naprogramovat. Např. pokud chcete, aby se při stisku tlačítka spustil či zastavil program.

Vpravo nahoře je možnost vidět USB port 2.0, označený USB. Dále jsou v horní části tři výstupní porty pro servomotory s označením A, B, C. Dole jsou označeny čtyři vstupní porty pro připojení modulů 1, 2, 3, 4. Při propojení modulů s řídící jednotkou kabelem je označení důležité. Především při programování, kdy se nastavují senzory, podle toho, v jakém portu se nacházejí [8].




**Obrázek 9: Řídící jednotka Lego Midstorms NXT**

*Zdroj: upraveno podle [10]*

### 2.1.2 Vstupní moduly

Vstupní moduly slouží ke vnímání. Mezi základní lidské smyslové vlastnosti patří sluch, hmat, čich, zrak a chuť. Roboti Mindstorms NXT mají také možnost některé z těchto vjemů „vnímat“ a to právě díky senzorům. Pomocí senzorů robot sbírá informace a data o svém okolí. Díky senzorům může robot jednat podle dané situace. Mezi základní senzory pro stavebnice Lego, které se dají připojit k řídící jednotce, patří dotykový senzor, světelný

senzor, zvukový senzor a ultrazvukový senzor. S těmito senzory bude pracováno při programování robotických úloh [8], [10].

### Dotykový senzor

Dotykový senzor je tlačítko vracející logickou hodnotu pravda/nepravda (true/false), podle toho, jestli je malé tlačítko na přední straně stlačené či nikoli. Nedokáže rozeznat intenzitu stlačení. Tím je jeho použití dosti omezené.

K senzoru lze přiřadit tři různé akce [8], [10]:

- Zmáčknutí (pressed) – pokud je tlačítko zmáčknuto, vrací hodnotu 1, pokud tlačítko zmáčknuto není, vrací hodnotu 0.
- Uvolnění (released) – uvolnění funguje opačně než zmáčknutí. Pokud je tlačítko zmáčknuto, vrací hodnotu 0 a pokud tlačítko není zmáčknuto, vrací hodnotu 1.
- Zmáčknutí a opětovné uvolnění (bumped) – počítá, kolikrát tlačítko narazilo. Přičemž naražením se myslí zmáčknutí a rychlé uvolnění.

**Obrázek 10: Dotykový senzor 9843**

*Zdroj: upraveno podle [6]*

### Světlený senzor

Světelný senzor umožňuje robotovi vidět. Dalším senzorem, který slouží k vidění je ultrazvukový senzor, o kterém bude více sděleno níže. Světelný senzor umožňuje rozlišovat intenzitu světla kolem, i intenzitu odráženého světla od blízkého objektu použitím diody. Senzor tedy neurčuje pouze světlo a tmu, dokáže dokonce i rozpoznávat barvy podle intenzity odstínů. Hodnoty, které senzor vrací, jsou od 0 do 100, kde nejtmavší hodnota je 0 a nejsvětlejší hodnota je 100.

V praktickém využití může světelný senzor určovat intenzitu světla v místnosti či venku. Nejčastěji se ale využívá spíše při sledování kontrastní čáry, umístěné na podlaze. To se využívá u zásobovacích robotů, kteří samostatně jezdí po pracovišti podle nakreslené vodící čáry [8].

**Obrázek 11: Světelný senzor 9844**

*Zdroj: upraveno podle [6]*

### Zvukový senzor

Zvukový senzor slouží jako mikrofon, který zachycuje intenzitu zvuku. Jeho intenzitu vyjadřuje dvěma způsoby. V decibelech (dB) a procentuálním vyjádření hlasitosti (dBA). Decibely se určují podle velikosti frekvence. Takový zvuk může být pro lidské ucho neslyšitelný. Procentuální vyjádření hlasitosti zvuků odpovídá takové hlasitosti, která je pro lidské ucho slyšitelná [8]. Intenzitu zvuku zjišťuje v rozmezí od 0 do 100 % [10]. Zjištěné hodnoty lze při programování porovnávat pomocí podmínek „větší – menší“.

V praktickém životě může být zvukový senzor využit například v bezpečnostních zařízeních, kde při vloupání vzniká hluk, který může zvukový senzor zachytit [8].

**Obrázek 12: Zvukový senzor 9845**

*Zdroj: upraveno podle [6]*

### Ultrazvukový senzor

Ultrazvukový senzor je další senzor, který umožňuje robotovi vidět. Díky tomuto senzoru může robot hledat předměty, vyhýbat se překážkám, měřit vzdálenost a zaznamenávat pohyb. Rozsah vzdálenosti rozlišuje od 0 do 255 cm s ± 3 cm.

A jak vlastně senzor funguje? Vyšle akustickou vlnu o vysoké frekvenci a spočítá dobu, za kterou se vlna odražená od předmětu vrátí. Z vypočítané doby určí poté vzdálenost. Tento způsob orientace v prostoru využívají i netopýři [8]. Co se týče povrchů, zvukové vlny se lépe odrážejí od rovných pevných povrchů. Pokud je povrch zaoblený či kulatý, vlny se odrážejí do různých směrů. Navíc měkké předměty mohou zvukové vlny i absorbovat. Proto je lehčí detekovat těžké ploché objekty [10].

Při programování se opět využívají podmínky „větší – menší“. V praxi může ultrazvukový senzor měřit vzdálenost od překážky. S odhadováním vzdálenosti od přepážky je možné se setkat například u domácích vysavačů [8].

**Obrázek 13: Ultrazvukový senzor 9846**

*Zdroj: upraveno podle [6]*

**Rozšiřující moduly**

Při tvorbě složitějších robotických úloh je možné využít senzory od jiných firem. Mezi nejznámější výrobce patří HiTechnic a Verner [8], [11], [12] . Jelikož s nimi nebylo pracováno, nebudou podrobněji rozebírány.

- Slučovač tlakových senzorů
- Teplotní senzor
- Kompasový senzor
- Senzor zrychlení
- Gyroskopický senzor
- Optický senzor měření vzdáleností
- Infračervený míč
- Infračervený vyhledávač
- Barevný světelný senzor

### 2.1.3 Výstupní moduly

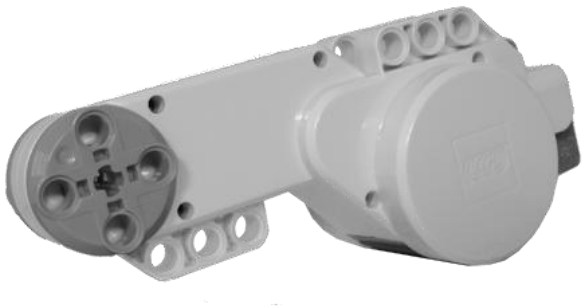
Mezi výstupní moduly řídící jednotky patří 8bitový zvukový výstup s reproduktorem, k použití pro výstrahy a signály. Obrazový výstup na černobílém displeji sloužící pro výpis hlášení a zobrazení obrázků. Dále mezi výstupní moduly patří světelné kostky využívané pro signalizaci a výstrahy, a interaktivní servomotor [8].

**Interaktivní servomotor**

V prvé řadě je třeba uvést, že servomotor není čistě výstupní modul, ale může být také použit jako vstupní modul. Častěji se ale používá jako modul výstupní. Servomotory jdou k řídící jednotce připojit celkem 3. Každý servomotor má rotační senzor, který měří otáčení motoru [8].

Rotaci servomotorů můžeme rozdělit na 4 doby trvání [8]:

- Neomezená doba – servomotor se otáčí až do skončení programu.
- Stupně – servomotor se otáčí podle daného počtu stupňů. Pokud jsou nastavené stupně kladné, servomotor se otáčí po směru hodinových ručiček, v případě že jsou nastavené stupně záporné, servomotor se otáčí proti směru hodinových ručiček.
- Rotace – je daný přesný počet rotací, kolikrát se servomotor otočí. Jedna rotace udává 360°.
- Sekundy – rotace servomotoru je dána na určitou dobu.



**Obrázek 14: Servomotor 9842**

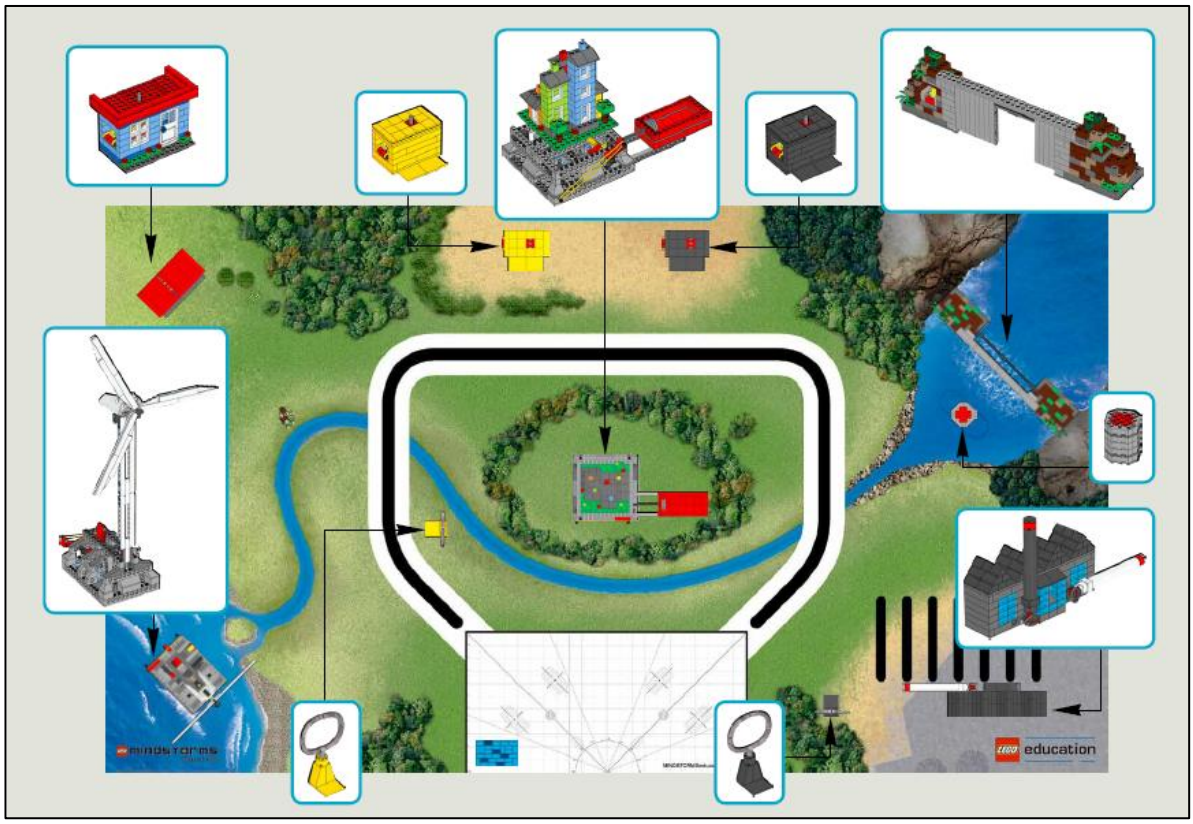
*Zdroj: upraveno podle [8]*

## 2.2 Zelené město

Sada Zelené město slouží k vybudování energeticky úsporného města, ve kterém můžou roboti plnit spoustu úkolů. Při splnění každého úkolu student obdrží „Energetickou kostku“, která po splnění daného úkolu vypadne ze stavby. Zelené město slouží pro studenty jednak pro rozvoj myšlení a naučení se programování, ale také pro ekologický cit. Úlohy jsou totiž zadány na způsob rozvoje obnovitelných zdrojů energie, úspory energie, snížení znečištění, recyklace materiálů, řízení nakládání odpadu a přístup k pitné vodě. Sada obsahuje tři tréninková plátna, cd se softwarem, a kostky stavebnice. Celá sada má celkem 1365 dílů [5].

Na plánu města (viz Obrázek 15) lze spatřit rozmístění jednotlivých objektů a jak jednotlivé předměty vypadají po sestavení. K jednotlivým objektům budou dále popsány úlohy, které by měl robot učinit. Za splnění každého úkolu student obdrží body, které se nakonec můžou sečíst a porovnat s ostatními studenty. Studenti by se postupně měli vypořádat se všemi úlohami Zeleného města. Poslední úlohou je Napájení města vodou. Pro splnění této úlohy musí robot zvládnout nejméně 4 úlohy předchozí. Při úspěchu se mu podaří napojit celé město [5], [14].

Na plánu lze také spatřit černou čáru, která je použita v úloze Jízda po černé čáře.




**Obrázek 15: Plán Zeleného města**

*Zdroj: [13]*

### 2.2.1 Větrná elektrárna

Robot musí aktivovat větrnou elektrárnu. Poté obdrží Energetickou kostku. Elektrárna je aktivována stiskem přepážky, která dá do chodu listy rotoru [14].

### 2.2.2 Umístění solárního panelu

Robot musí umístit solární panel na střechu domu. Pokud se mu to podaří, z domu vypadne Energetická kostka. Robot při úkolu nesmí shodit žádný jiný objekt [14].

### 2.2.3 Vyhnout se zalévající dámě

Během pohybu po plánu se robot nesmí dotknout postavičky dámy, která zalévá květiny.

### 2.2.4 Třídění odpadků

V úloze třídění odpadků robot pracuje se dvěma kroužky znázorňujícími odpadky a musí je umístit na odpadkový koš. Ty jsou rozděleny podle barev na žlutou a černou. Při odkládání odpadků na koš musí barvy korespondovat. Pokud nejsou odpadky umístěný na správný odpadkový koš, student neobdrží žádný bod [14].

### 2.2.5 Uzavření přehrady

Robot musí nalézt otvor v přehradě a nedaleký předmět, kterým přehradu uzavře. Otvor v přehradě lze různě nastavit podle potřeby. Po splnění úkolu vypadne z přehrady Energetická kostka [14].

### 2.2.6 Nasazení nového komínu

Úloha s továrnou je obtížnější než ostatní. Skládá se ze dvou kroků. Nejprve musí robot zvednout bílý komín, a teprve potom musí položit černý komín. Pokud robot vykoná vše ve správném pořadí, obdrží Energetickou kostku. Pokud robot úkol nesplní podle zadání, Energetická kostka nevypadne z továrny.

### 2.2.7 Napájení Zeleného města

Robot musí nejprve získat nejméně 4 Energetické kostky, které jsou nezbytné ke splnění úkolu. Energetické kostky musí vložit do červeného držáku. Pokud se mu to povede, dalším krokem je stlačení držáku a očekávání úspěchu. V případě výhry se domeček na vršku roztočí dokola [14].

# 3 SLOŽENÍ A TVORBA PROSTŘEDÍ PRO ROBOTICKÉ STAVEBNICE

Složení stavebnic a robotů proběhlo s pomocí přiložených návodů. V návodech je přesně popsán krok za krokem pomocí obrázků. U každého kroku jsou zobrazeny potřebné součástky někdy i ve velikosti 1:1, které jsou potřebné k dokončení aktuálního kroku. Veškeré stavebnice byly sestaveny do poslední kostičky. Jejich stavba zabrala poměrně hodně času.

Samozřejmě je také možné sestavit si robota podle svého gusta. Na to je ovšem potřeba řádný kus fantazie a velká představivost. Nelze opomenout ani zručnost.

Po sestavení robotů již následuje programování. Všechny úlohy v této práci byly tvořeny v programu Enchanting verzi 0.2.3.

## 3.1 Enchanting

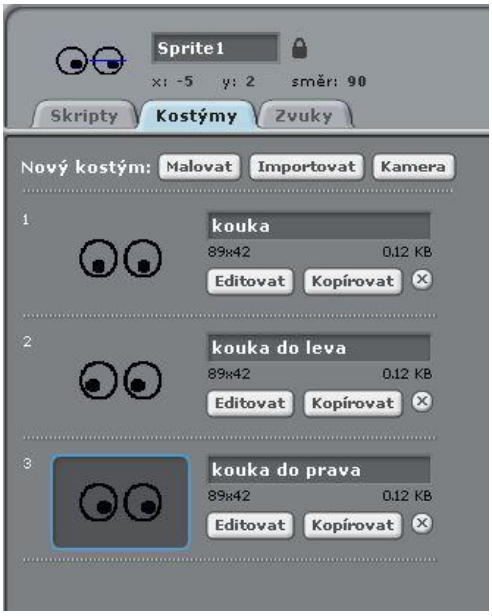
Enchanting je nástroj k programování, ke kterému není třeba znát žádný složitý programovací jazyk. Lze ho stáhnout pro operační systémy Windows, Mac OS X a Linux [15]. Pracovala jsem s programem v české verzi. Vše ale nebylo přeložené, některé části zůstali v angličtině. Enchanting je snadno pochopitelný a dobře se s ním dělá. Je založený na programovacím jazyku Scratch, který umožňuje snadno vytvářet interaktivní příběhy, animace, hry a hudbu [16]. Enchanting je také založen na programovacím jazyku Snap/BYOB (build your yown block), který je rozšíření Scratche. Tento jazyk je na bázi drag-and-drop (vzít a položit) [17]. Do řídící jednotky je nahrána java leJOS NXT [18].

Tvoření úloh pomocí Enchantingu bude popsáno především v další kapitole. Ta bude obsahovat jednotlivé příklady s vysvětlením a obrázkem z programu. Nejprve se ale seznámíme s editorem malování a vytvářením nových bloků. Tyto editory budou využity v některých úlohách.

### 3.1.1 Malování v Enchantingu

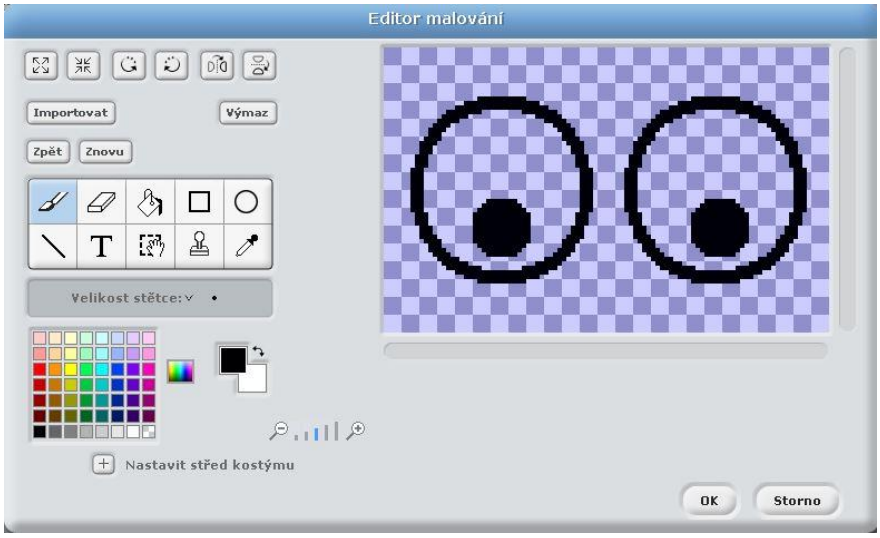
Jak již bylo řečeno, řídící jednotka obsahuje displej, na kterém je možné vykreslovat různé texty či obrázky. Enchanting zobrazuje tři záložky, ze kterých lze vybírat – Skripty, Kostýmy a Zvuky. Záložka Kostýmy lze ještě přepínat na záložku Pozadí. Na Pozadí se zobrazují různé scény, jak by mělo být z názvu jasné. V programu je přednastavených několik možností na výběr. Buď může být scéna čistě bílá, anebo lze nastavit scénu uvnitř místnosti, vně místností či sportovní hřiště. Scénu je možné i samostatně namalovat v editoru.

Ve složce Kostýmy se zobrazuje obrázek Spirit. Těch může být v jednom programu i více. Stejně jako u Pozadí, i Kostýmy jsou některé již přednastavené a to z oblastí zvířat, fantasy, písmen, lidí, věcí a dopravy. Následující obrázek (viz Obrázek 16) slouží ke tvorbě robotické úlohy Jízda ovládaná tlačítky + koukání různými směry. Obrázky očí byly vytvořeny v editoru malování (viz Obrázek 17). Lze zde kreslit štětcem či gumovat. Malovat kolečka, čtverce, rovné čáry, také i psát. Používat různé barvy (které nám při práci s roboty nijak užitečné nejsou, jelikož displej řídící kostky je černobílý). Obrázek lze také kopírovat, otočit, přetočit a měnit velikost.




**Obrázek 16: Prostředí Enchanting – záložka Kostýmy**

*Zdroj: vlastní*




**Obrázek 17: Editor malování**

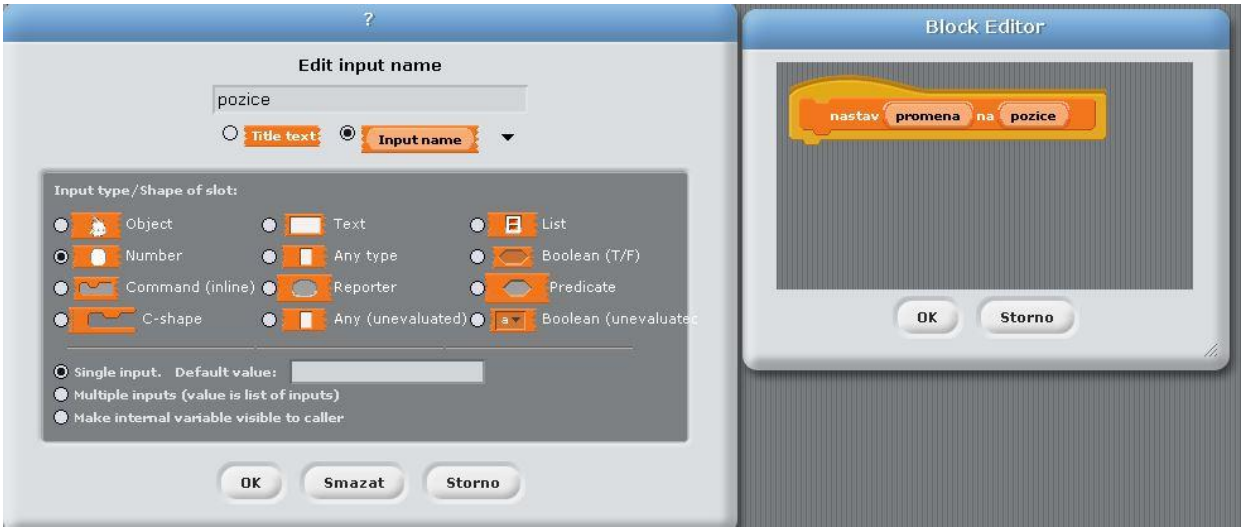
*Zdroj: vlastní*

### 3.1.2 Vytváření nových bloků v Enchantingu

Dalším užitečným editorem Enchantingu je editor na vytváření nových bloků. Nejdříve je nutné vybrat kategorii, do které bude nový blok příslušet, jaký bude typ, jeho název a zda bude použit pro všechny programy či pouze pro tento program (viz Obrázek 18). Poté se zobrazí další tabulka (viz Obrázek 19). Zde se již blok upravuje do konečné podoby. Vkládá se titulní text a různé typy vstupu. Na pravé straně obrázku je možné vidět konečnou podobu bloku.


**Obrázek 18: Vytvoření nového bloku**

*Zdroj: vlastní*




**Obrázek 19: Nastavení nového bloku**

*Zdroj: vlastní*

# 4 TVORBA ÚLOH A JEJICH NÁSLEDNÁ IMPLEMENTACE PRO ROBOTICKÉ STAVEBNICE

Všechny úlohy jsou vytvořeny pomocí programu Enchanting 0.2.3. Následné úlohy budou sloužit pro výuku předmětu Základy algoritmizace, a proto jsou podle toho i tak koncipovány. Každá úloha bude mít vždy určité části, a to:

- název úlohy
- zadání pro studenty
- popis, co by měl robot tedy vykonávat
- moduly, které jsou k robotovi připojeny a v jakých portech jsou zapojeny
- popis programu
- shrnutí, poznámky a případné doplňující úlohy pro studenty
- obrázek vytvořeného programu

## 4.1 Seznámení s robotem

Nejprve se seznámíme s robotem, než začneme vytvářet složitější úlohy. Začneme s jednoduchými úkony, při kterých se robot pohybuje. Tyto úkony jsou rozděleny podle způsobu naprogramování a to buď přes konfiguraci Driving, anebo přes konfiguraci Motors. Následně se bude zkoušet, jak fungují různé přídavné moduly. Byly vybrány 4 základní moduly, které obsahovala stavebnice, a to dotykový senzor, světelný senzor, zvukový senzor a ultrazvukový senzor.

### 4.1.1 Pohyb robota

**Pohyb Driving**

Prvním způsobem pohybu robota, který si ukážeme, je přes konfiguraci Driving. Tento způsob slouží přímo ke konfiguraci dvou servomotorů, které budou robota udávat do pohybu. Můžeme říci, že se jedná o konfiguraci pro „jezdítko“. Do editoru se dostanete tak, že kliknete na Driving a následně na Configure Drive Type. Poté se zobrazí okno Configuration editor. Je třeba z levé části editoru přesunout „nastavení řízení“ do pravé části editoru a připnout ho k Drive type. Je tu pět oblastí, které se musí nakonfigurovat. První z nich je velikost kol. Většinou je údaj napsán přímo na kole, proto ho stačí jenom přečíst. Pokud by na kole napsán

nebyl, je třeba tento údaj zjistit pomocí pravítka. Druhým údajem je šířka mezi koly. Tento údaj již je třeba změřit pomocí pravítka. Měří se od vnější strany pravého kola k vnitřní straně levého kola. Třetím a čtvrtým údajem je připojení servomotorů k řídící jednotce. Zde je na výběr jestli se servomotor připojí do portu A, B či C. Pátým a posledním údajem je nastavení otáčení servomotorů buď po směru hodinových ručiček, nebo obráceně (dopředu nebo opačně). To záleží pozici servomotorů, jakým směrem byly umístěny při stavbě robota.


**Obrázek 20: Konfigurace Drive**

*Zdroj: vlastní*

**Název:** **Jízda do čtverce**

**Zadání:** Pomocí Driving vypracujte program pro jízdu robota do čtverce. Robot by měl skončit na stejném místě, na kterém začal.

**Popis:** Po spuštění programu se robot rozjede a ujede určitou vzdálenost. Když ujede zadanou vzdálenost, otočí se doprava o 90°. Tyto dva úkony by se měli 4 opakovat. Po posledním otočení by měl být robot opět na místě, odkud začínal při spuštění programu.

**Moduly:** Při této úloze jsou zapotřebí dva moduly a to servomotory. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B.

**Program:** Po spuštění programu se robot rozjede a ujede vzdálenost 25 cm pomocí dvou servomotorů. Až tuto vzdálenost ujede, otočí se doprava o 90°. Jízda dopředu a

otočení se o 90° se bude opakovat celkem čtyřikrát. Až robot vykoná všechny úkony, program skončí.

**Shrnutí:** V této úloze je jasně vidět, jak je důležitá správná konfigurace motorů. Pokud totiž motory nebudou správně nastaveny, neudělá robot čtverec. Neujede o správný počet centimetrů a už vůbec se neotočí přesně o 90°. Vyzkoušejte, jak se bude v této úloze robot chovat, pokud změníte nastavení vzdálenosti kol od sebe. Dále upravte program tak, aby robot udělal čtverec při jízdě pozadu a otočkách doleva.

Všimněte si pravé části obrázku (viz Obrázek 21). Při každém programu je nastaveno, že pokud stisknete na řídící jednotce Cancel, program se ukončí. Pro lepší přehlednost tato část programu nebude již na dalších zobrazení vidět. Počítejte ale s tím, že byla v každém programu použita.


**Obrázek 21:Jízda do čtverce**

*Zdroj: vlastní*

**Název:** **Jízda – Drive forward steering**

**Zadání:** Zjistěte, jak se bude robot chovat, při nastavování různých hodnot u Drive forward steering.

**Popis:** Po spuštění programu robot pojede 2 vteřiny daným směrem, a poté se na 2 vteřiny zastaví. Použijte postupně všechny hodnoty, které lze nastavit u Drive forward steering a posuďte, při kterých hodnotách jakým směrem robot jede.

**Moduly:** Při této úloze jsou zapotřebí dva moduly a to servomotory. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B.

**Program:** Pokud začneme u hodnoty 200 a budeme ji postupně snižovat až na hodnotu -200 tak se po spuštění programu se robot rozjede, ujede 2 vteřiny doleva, poté se na 2 vteřiny zastaví a opět se rozjede, tentokrát pojede 2 vteřiny mírně doleva a opět se poté na dvě 2 vteřiny zastaví. Dalším směrem pojede 2 vteřiny rovně a na další 2 vteřiny zastaví. Při hodnotě -100 pojede mírně doprava a při hodnotě

-200 zatáčí doprava. Na Obrázek 22 je vidět, jak jsou postupně vyzkoušeny všechny možné hodnoty a podle nich se robot pohybuje určitým směrem.

**Shrnutí:** Je 5 různých směrů, kudy může robot jet. Pokud započítáme i to, že můžeme nastavit jízdu pozadu, těchto směrů je ve výsledku 10. Při tomto cvičení je hlavním úkolem posoudit, kterým směrem robot jede při různě nastavených hodnotách a při zatáčení určit, o kolik stupňů se robot otáčí.


**Obrázek 22: Jízda – Drive forward steering**

*Zdroj: vlastní*

**Název:** **Jízda – Drive - různá**

**Zadání:** Osvojte si další možné jízdy robota při konfiguraci Driving. Vytvořte program, kde robot pojede určitou vzdálenost rovně, poté sníží rychlost, bude zatáčet doleva po určitou vzdálenost a poté se prudce otočí pravým směrem. Nepoužívejte již Drive forward steering.

**Popis:** Nejprve by měla být nastavená určitá rychlost, kterou robot pojede, poté se rozjede a ujede 50 cm rovně. Po ujetí této vzdálenosti snižte rychlost robota a nastavte jízdu tak, aby robot zatáčel po určitou vzdálenost o daný počet stupňů doleva. Nakonec nastavte rychlou otočku doprava.

**Moduly:** Při této úloze jsou zapotřebí dva moduly a to servomotory. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B.

**Program:** Na začátku programu je nejprve nastavena rychlost na 60 cm/s a robot ujede 50 cm rovně. Poté se přenastaví rychlost na 20 cm/s a robot bude pokračovat stále dopředu se zatáčením do leva o 90°. Tímto směrem ujede 50 cm. Nakonec se otočí prudce doprava o 150°. Tím je program ukončen.

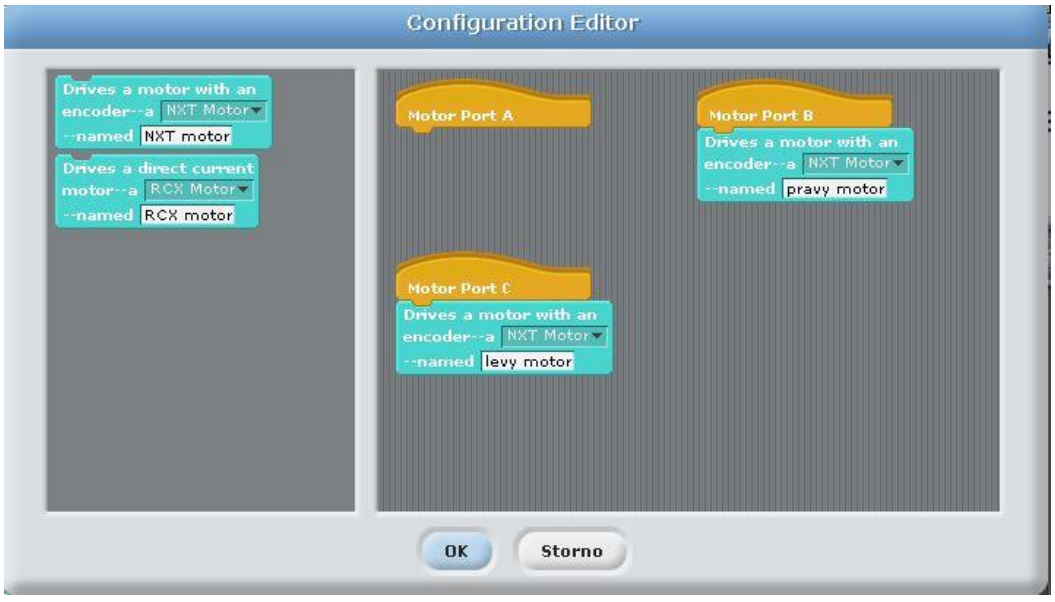
**Shrnutí:** Studenti by si měli vyzkoušet všechny různé druhy jízdy, který program nabízí a sami si zjistit, jak se robot chová podle různých instrukcí.


**Obrázek 23: Drive - jízda různá**

*Zdroj: vlastní*

### Pohyb Motors

Další možný pohyb robota umožňuje konfigurace jednotlivých motorů. Jak je zřejmé z Obrázek 24, nastavuje se každý servomotor zvlášť. Proto je třeba při programování pracovat s každým motorem jednotlivě. Editor slouží právě i kvůli tomu, pokud nemáme jen 2 servomotory ale 3. Třetí servomotor je použit při dalších úkonech. Pravý servomotor je připojen k potru B a levý servomotor k portu C, stejně jako v ostatních úlohách.




**Obrázek 24: Konfigurace Motors**

*Zdroj: vlastní*

**Název:** **Jízda – Motors - různá**

**Zadání:** Seznamte se s různými způsoby jízdy robota pomocí nastavení dvou servomotorů.

**Popis:** Vyzkoušejte 4 různé techniky jízdy a pozorujte, jak se robot zachová. Rozjeďte robota dopředu a následně zastavte pomocí Stop by braking. Dále spusťte jízdu pozadu a zastavte robota pomocí Stop by coasting. Následně vyzkoušejte, co se stane, pokud nastavíte stejnou rotaci obou motorů o kladný počet stupňů s parametrem Until done a rotaci obou motorů, z nichž jeden bude mít nastaveny kladně stupně a druhý záporně s parametrem Just start it.

**Moduly:** Při této úloze jsou zapotřebí dva moduly a to servomotory. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B.

**Program:** V průběhu programu se na displeji zobrazí číslovky od 1 do 4. Je to z toho důvodu, aby byly lépe rozpoznatelné jednotlivé kroky. Prvním příkazem je jízda vpřed trvající 3 vteřiny a zastavení Stop by braking. Pomocí Stop by braking se nejdříve zastaví pravý motor, až poté se zastaví levý motor. Při vyšší rychlosti je možné, že se robot mírně stočí doprava. Druhou úlohou je jízda vzad a následné zastavení Stop by coasting. Stop by coasting znamená, že začne zastavovat pravý motor a hned v jeho průběhu zastavování se začne zastavovat i levý motor. Oba motory by se ve výsledku měli zastavit přiměřeně ve stejnou dobu. Třetí úlohou je otočení obou servomotorů o 200° s parametrem Until done (dokud se nedokončí). Pokud bychom zadali zápornou hodnotu stupňů, motory by se otáčely vzad. V případě nastavení parametru Until done, otočí se nejprve pravý motor o 200° a teprve až bude příkaz vykonán, otočí se i druhý motor o 200°. V tom případě jízda není plynulá vpřed, ale nejprve se robot stočí na levou stranu a poté na pravou stranu. Naproti tomu, pokud by úloha byla s parametrem Just start it (prostě začni), oba motory by se spustili téměř souběžně a robot by jel kupředu. V úloze čtvrté je nastaven pravý motor na 200° otáčení a levý motor na -200° otáčení s parametrem Just start it. Oba motory se spustí téměř souběžně a robot se stočí doleva.

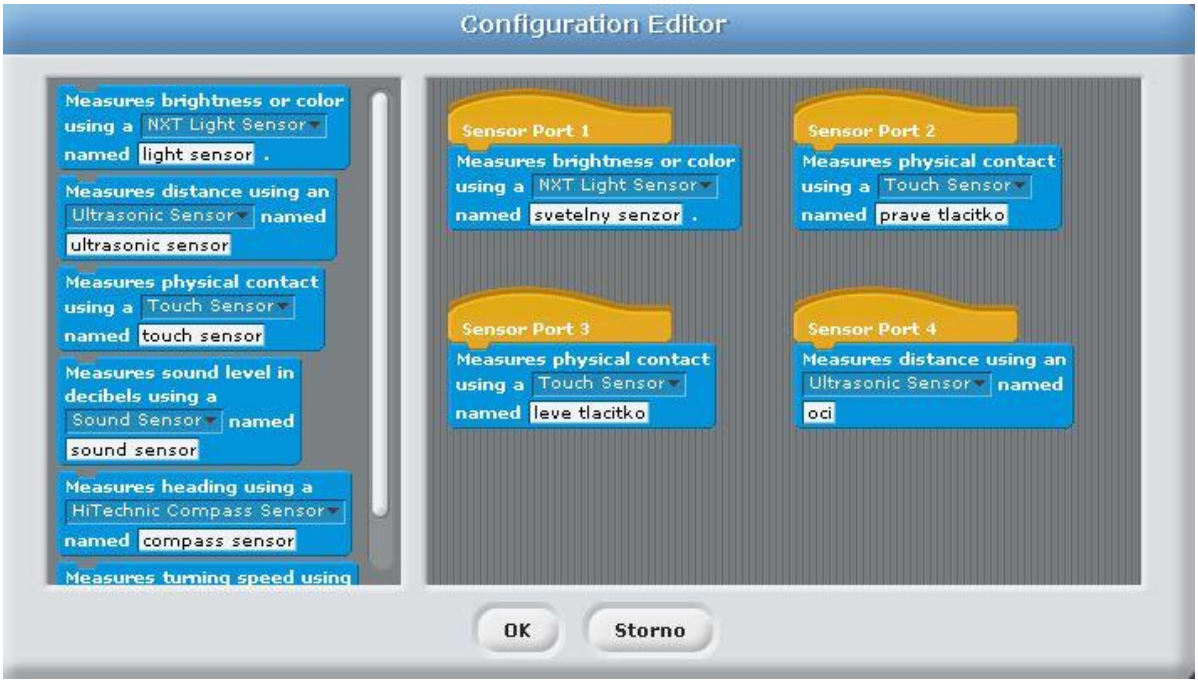
**Shrnutí:** V této úloze je jasně vidět, jak velké rozdíly jsou při malých změnách v nastavení. Studenti si mohou vyzkoušet i další možnosti nastavení motorů a jejich způsobů otáčení.


**Obrázek 25: Motors - jízda různá**

*Zdroj: vlastní*

### 4.1.2 Funkčnost senzorů

Pro práci s robotem budeme hodně využívat jeho přídavné moduly senzory. Na Obrázek 26 lze spatřit nastavení senzorů připojených k jednotlivým portům. Toto nastavení bylo využito ve většině úloh.




**Obrázek 26: Konfigurace senzorů**

*Zdroj: vlastní*

### Dotykový senzor

Dotykový senzor se dá využít dvojím způsobem, může být volně připnutý tak, abychom ho drželi v ruce a reagoval na stisk tlačítka prstem, anebo ho lze umístit při stavbě robota. V tom případě bude reagovat, pokud narazí na nějakou překážku.

**Název:** **Dotykový senzor – při stisku vydá tón**

**Zadání:** Vytvořte program, ve kterém, pokud zmáčknete tlačítko, řídící jednotka vydá tón. Program dále rozšiřte o zobrazení, zda je tlačítko stisknuto či ne.

**Popis:** Aby při každém stisku tlačítka vydal program tón, je třeba vytvořit nekonečný cyklus, v kterém bude podmínka, že pokud je tlačítko stisknuté, vydá tón. Pokud ne tak nic. Rozšíření o zobrazení hodnoty může být buď booleanovská 0 a 1, anebo nějaké sousloví jako například „Píp“ a „Píp píp“.

**Moduly:** Při této úloze je zapotřebí pouze modul dotykový senzor, který je zapojen do portu 3.

**Program:** Po spuštění programu, se na displeji zobrazí defaultní hodnota „Píp píp“, která znamená, že tlačítko není zmáčknuté. Pokud tlačítko zmáčknuté je, zobrazí se na displeji hodnota „Píp“ a řídící jednotka vydá určitý tón po 0,3 sekundy. Všechny úkony jsou zapouzdřené do nekonečného cyklu, aby se hodnoty měnily pokaždé, co tlačítko stiskneme.

**Shrnutí:** Všimněte si, že při stisku tlačítka se mění hodnoty přímo vteřinově. Pokud tlačítko držíte déle, není hodnota stále nastavena na stisknuté tlačítko, ale rychle se mění ze stisknuté na nestisknuté. Pokud máme tón nastavený na 0,3 vteřiny, tak bude vždy hrát přesně 0,3 vteřiny a ne míň, i přesto když tlačítko stiskneme dříve, než tón dozní.


**Obrázek 27: Dotykový senzor - při stisku tlačítka řídící jednotka vydá tón**

*Zdroj: vlastní*

**Název:** **Dotykový senzor - hudba**

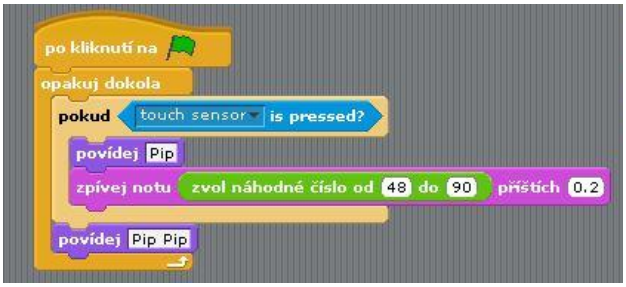
**Zadání:** Při stisku tlačítka vydá řídící jednotka náhodný tón.

**Popis:** Vytvořte program tak, aby při stisku tlačítka vydala řídící jednotka tón, který bude po každém stisku tlačítka jiný.

**Moduly:** Při této úloze je zapotřebí pouze modul dotykový senzor, který je zapojen do portu 3.

**Program:** Celý program je zapouzdřen do nekonečného cyklu. Pokud se stiskne tlačítko, zobrazí se na displeji „Píp“ a řídící jednotka vydá tón podle stupnice od 48 do 90 po dobu 0,2 vteřiny. Pokud tlačítko stisknuté nebude, řídící jednotka nevydá žádný tón a na displeji bude zobrazeno „Píp píp“.

**Shrnutí:** Rozšiřte tento program ještě dále o možnost, že náhodný tón bude znít náhodně dlouze.




**Obrázek 28: Dotykový senzor - při stisku tlačítka řídící jednotka vydá různé tóny**

*Zdroj: vlastní*

**Název:** **Dotykový senzor – hrací kostka**

**Zadání:** Vytvořte program, který při stisku tlačítka „hodí kostkou“ – vybere náhodné číslo od 1 do 6.

**Popis:** Program bude obsahovat nekonečný cyklus, ve kterém je vložena podmínka, zda je tlačítko stisknuté. Pokaždé, když se tlačítko stiskne, zobrazí se na displeji náhodné číslo od 1 do 6.

**Moduly:** Při této úloze je zapotřebí pouze modul dotykový senzor, který je zapojen do portu 3.

**Program:** Celý program je zapouzdřený do nekonečného cyklu, ve kterém je jedna podmínka. Pokud bude stisknuté tlačítko, na displeji se zobrazí náhodná hodnota od 1 do 6, což znázorňuje hod hrací kostkou. Po prvním stisknutí na

displeji již zůstane zobrazeno vybrané číslo a po každém dalším stisknutí se číslo změní. Pokud tlačítko stisknete opět dříve než za jednu vteřinu, číslo se nezmění.

**Shrnutí:** Jednoduchý a výstižný příklad na vyzkoušení si stisku tlačítka a také zobrazení pomocí Povídej. Uvědomte si, že nové číslo se vygeneruje teprve, pokud stisknete tlačítko déle jak po jedné vteřině od předchozího stisku tlačítka.


**Obrázek 29: Dotykový senzor - hrací kostka**

*Zdroj: vlastní*

### Světelný senzor

**Název:** **Světelný senzor – zobrazení intenzity světla**

**Zadání:** Vytvořte program, který na displeji řídící jednotky zobrazí intenzitu světla.

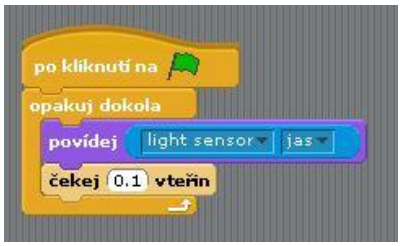
**Popis:** Pomocí příkazu Povídej zobrazte hodnoty intenzity světla pomocí světelného senzoru. Vyzkoušejte rozdíl hodnot při kalibraci světelného senzoru, a pokud kalibraci senzoru do programu nedáte.

**Moduly:** Při této úloze je zapotřebí světlený senzor, který je zapojený do portu 1.

**Program:** Program obsahuje nekončený cyklus, ve kterém je příkaz Povídej a příkaz Čekej 0,1 vteřiny. Program s kalibrací senzoru obsahuje také nekonečný cyklus. Před ním je ovšem nejdříve automaticky kalibrován světelný senzor, aby se přizpůsobil okolnímu prostředí. Na začátku každého cyklu se senzor opět automaticky kalibruje podle aktuálního načítání.

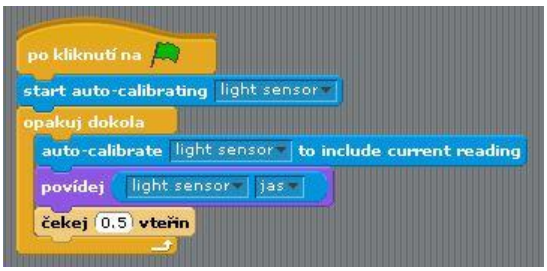
**Shrnutí:** A jaký je tedy rozdíl mezi programem s kalibrací (viz Obrázek 31) a bez auto kalibrace (viz Obrázek 30)? Při měření budou vycházet různé hodnoty podle prostředí, ve kterém se nacházíte. V mém případě vycházeli hodnoty při zobrazování různých barev bez auto kalibrace následovně: černá 36, světle modrá 41, červená 51, žlutá 55 a bílá 55. V programu s kalibrací světelného senzoru byly hodnoty následující: černá 0, světle modrá 50, červená 79, žlutá 90

a bílá 100. Různé hodnoty vyšly z toho důvodu, že při kalibraci světelný senzor ukazuje hodnoty v procentech, přičemž nejmenší možná hodnota je 0 (černá) a nejvyšší možná hodnota je 100 (bílá).




**Obrázek 30: Světelný senzor - zobrazení intenzity světla bez auto kalibrace**

*Zdroj: vlastní*




**Obrázek 31: Světelný senzor – zobrazení intenzity světla s auto kalibrací**

*Zdroj: vlastní*

**Název:** **Světelný senzor – rozpoznání barvy míčků**

**Zadání:** Vytvořte program, kde pomocí světelného senzoru rozpoznáte barvu přiložených míčků.

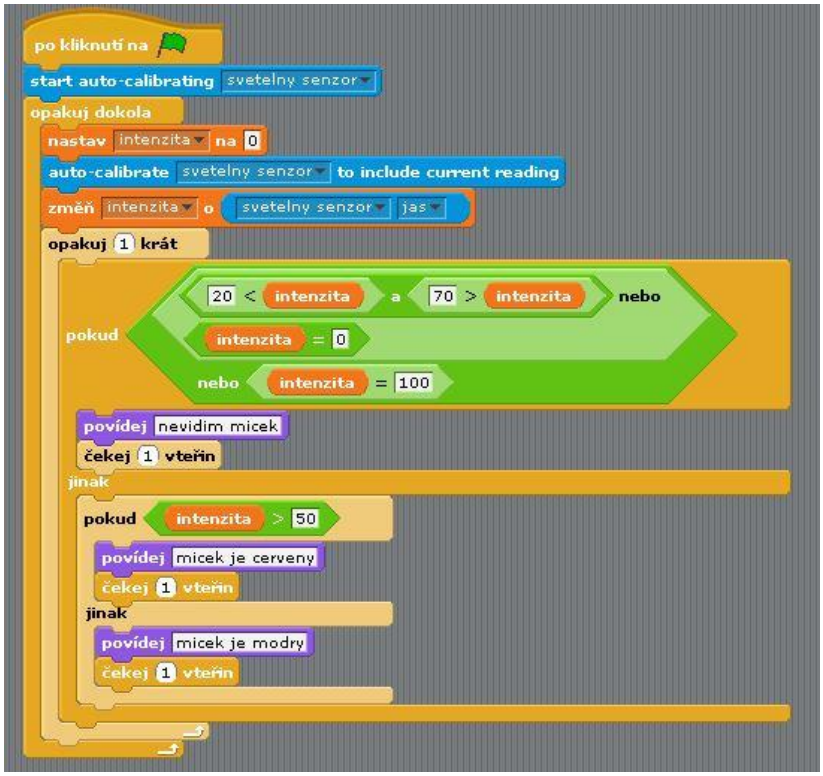
**Popis:** Vytvořte program, který určí barvu míčku podle intenzity odstínu. Na displeji poté zobrazte, zda je míček modrý, či červený, anebo pokud senzor nezachytí žádný míček, takž že senzor míček nevidí.

**Moduly:** Při této úloze je zapotřebí světlený senzor, který je zapojený do portu 1.

**Program:** Nejprve je třeba auto kalibrovat světelný senzor, podle aktuálního umístění robota. Dále již bude následovat nekončený cyklus. Na začátku každého cyklu se nastaví proměnná Intenzita na hodnotu 0 a překalibruje se světelný senzor. Proměnná Intenzita se změní, podle hodnoty senzoru. Následuje cyklus, který se bude opakovat vždy pouze jedenkrát, pro určení barvy míčku. Je zde podmínka, která určuje, zda je před senzorem míček či není. Pokud míček před senzorem není, vypíše se na displeji „nevidím míček“ a ukončí se cyklus. Pokud senzor míček rozpozná, je zde další podmínka, která určí intenzitu míčku. V případě, že

je Intenzita menší než 50 %, na displeji se zobrazí „míček je modrý“, v opačném případě se zobrazí „míček je červený“.

**Shrnutí:** Oba míčky jsou součástí stavebnice. Hodně záleží na tom, jaká je intenzita okolního světla, proto doporučuji nejprve vyzkoušet, jakou hodnotu mají míčky při přiložení, a teprve potom začít programovat a nastavovat hodnoty.




**Obrázek 32: Světelný senzor - rozpoznání barvy míčků**

*Zdroj: vlastní*

**Název:** **Světelný senzor – světelný graf**

**Zadání:** Vytvořte program, při kterém se na displeji bude zobrazovat graf podle intenzity světla.

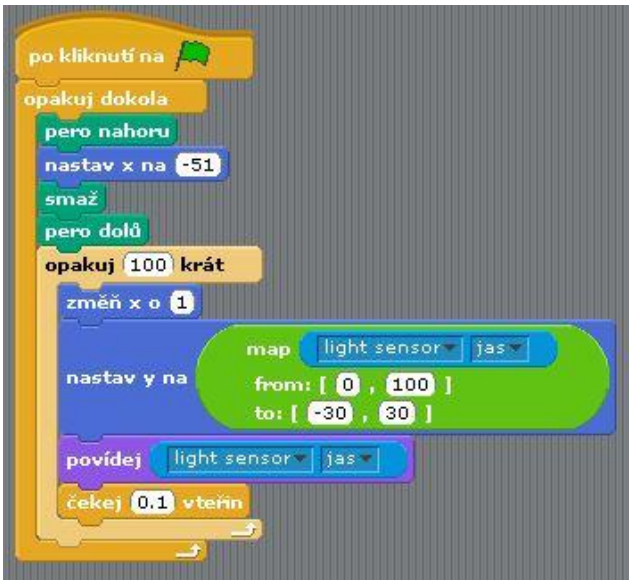
**Popis:** Program bude obsahovat nekonečný cyklus, díky kterému se budou na displeji řídící jednotky objevovat hodnoty světelného senzoru pomocí grafu. Pro zobrazení grafu použijte bloky Pohyb, Operátor a Pero.

**Moduly:** Při této úloze je zapotřebí světlený senzor, který je zapojený do portu 1.

**Program:** Program bez kalibrace světelného senzoru obsahuje nekonečný cyklus, ve kterém je na začátku nastavena hodnota X na -51. Postupně se tato hodnota bude přičítat o jedničku, a tím se bude graf vypisovat zleva doprava. Hodnota Y se bude měnit podle intenzity světla. Tím se bude Pero posouvat nahoru a dolů.

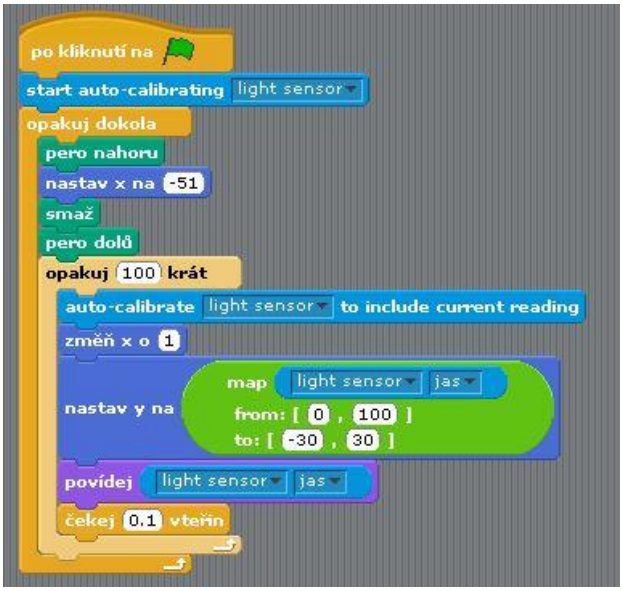
Na displeji se také zobrazí hodnota intenzity světla pomocí příkazu Povídej. V programu rozšířeném o auto kalibraci světelného senzoru jsou pouze přidané dva příkazy navíc (viz Obrázek 34), jinak je program stejný, jako bez auto kalibrace.

**Shrnutí:** Všimněte si rozdílů ve vytvořených grafech těchto dvou programů. V první verzi při přibližných hodnotách 36 až 55 je graf znázorněn spíše u středu displeje. V druhém programu, s auto kalibrací světelného senzoru, jsou hodnoty grafu znázorněny téměř od dolní části displeje až po horní část displeje.




**Obrázek 33: Světelný graf bez autokalibrace kalibrace**

*Zdroj:vlastní*




**Obrázek 34: Světelný graf s auto kalibrací**

*Zdroj: upraveno podle [19]*

### Zvukový senzor

**Název:** **Zvukový senzor – intenzita zvuku**

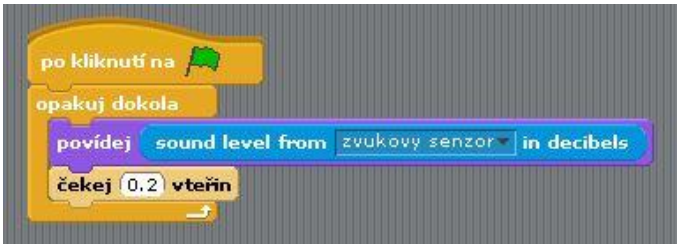
**Zadání:** Vytvořte program, který zobrazí intenzitu zvuku v decibelech.

**Popis:** Program bude mít nekonečný cyklus, ve kterém se na displeji zobrazí intenzita zvuku v decibelech.

**Moduly:** Při této úloze je zapotřebí zvukový senzor, který je zapojen do portu 2.

**Program:** V úloze bude nekonečný cyklus, aby se hodnoty neustále měnili podle aktuální situace. V cyklu je příkaz Povídej, který zobrazuje intenzitu zvuku. Dále je zde příkaz Čekej 0,2 vteřiny, který pomůže našemu vnímání. Pokud by zde příkaz Čekej nebyl, hodnoty by se měnili tak rychle, že by byli lidskému oku těžko rozeznatelné.

**Shrnutí:** Vyzkoušejte, jak se bude robot chovat, pokud bude v místnosti úplné ticho, pokud někdo promluví potichu a nahlas, či zatleská. Také vyzkoušejte, jakou robot naměří intenzitu hluku, pokud se bude pohybovat.




**Obrázek 35: Zvukový senzor - zobrazení hlasitosti**

*Zdroj: vlastní*

**Název:** **Zvukový senzor – zatleskej a popojeď**

**Zadání:** Při tlesknutí robot kousek popojede.

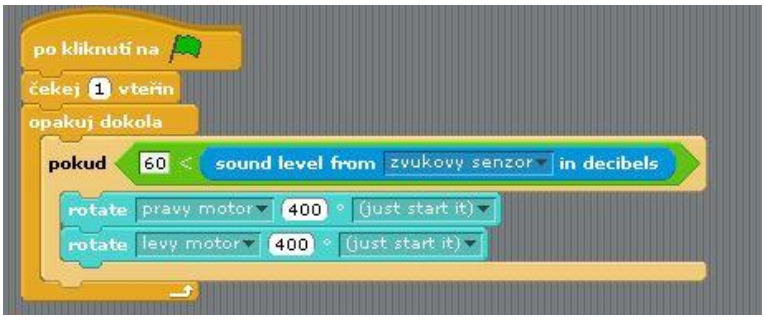
**Popis:** Vytvořte nekonečný cyklus, ve kterém pokud tlesknete, robot popojede dopředu. Vyzkoušejte jízdu pomocí konfigurace Motors.

**Moduly:** Při této úloze jsou zapotřebí 3 moduly a to 2 servomotory, pomocí nichž bude robot jezdit, a zvukový senzor. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B. Zvukový senzor je zapojen do portu 2.

**Program:** Na začátku programu robot vyčká 1 vteřinu. Poté je zde nekončený cyklus, ve kterém je podmínka, že pokud bude zvukový senzor přijímat hluk větší jak 60 decibelů, roztočí se dva servomotory o 400° a robot popojede dopředu. Poté

zastaví a čeká opět na hlasitý zvuk.

**Shrnutí:** Vyčkání na začátku programu 1 vteřinu je z důvodu, že robot při spuštění vždy vydá zvuk. Aby zvuk nebyl zařazen již do programu, robot vyčká, a program začne fungovat až právě po 1 vteřině. Můžete vyzkoušet i různé jiné jízdy. Upozorním ale, že pokud je robot v pohybu, špatně se s ním pracuje, proto je lepší, když vyčká na zvukový tón a teprve poté je udán do pohybu.




**Obrázek 36: Zvukový senzor - zatleskej a pojeď**

*Zdroj: vlastní*

**Název:** **Zvukový senzor – náhodná jízda**

**Zadání:** Vytvořte program, při kterém robot, pokud „uslyší“ tlesknutí, pojede náhodným směrem.

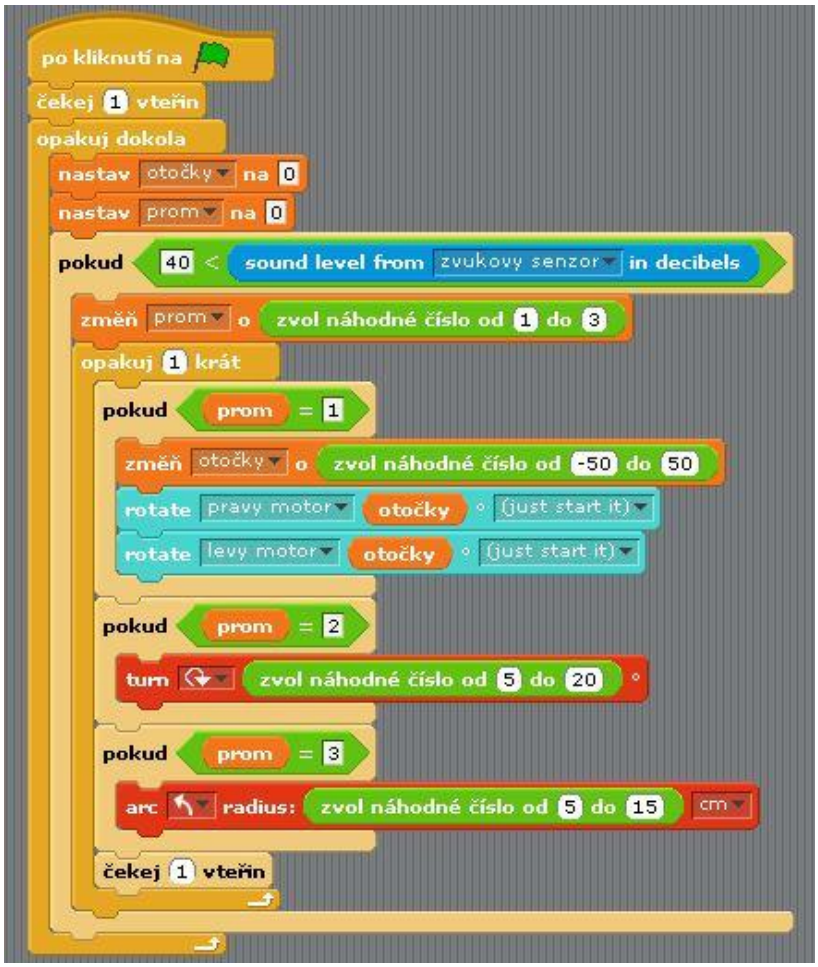
**Popis:** V programu použijte podmínku, že pokud robot zachytí hluk v určité intenzitě, rozjede se jedním ze třech směrů – rovně, vpravo, vlevo. Vypomožte si proměnnými.

**Moduly:** Při této úloze jsou zapotřebí 3 moduly a to 2 servomotory, pomocí nichž bude robot jezdit, a zvukový senzor. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B. Zvukový senzor je zapojen do portu 2.

**Program:** Na začátku programu robot vyčká 1 vteřinu, než začne něco dělat. Je to z toho důvodu, že při spuštění programu řídící jednotka vydá tón, který může narušit chod programu. Dále je zde vytvořen nekonečný cyklus, ve kterém jsou na začátku nadefinované dvě proměnné. Proměnná Otočky nastavená na hodnotu 0 a proměnná Prom nastavená na hodnotu 0. Dále je zde nastavena podmínka, pokud bude hodnota zvuku větší než 40 decibelů, vykonají se následující příkazy. Zde využijeme proměnnou Prom, kterou změníme náhodným výběrem čísla na čísla 1 až 3. Podle náhodně vygenerovaného čísla se uskuteční určitý příkaz. Pokud hodnota Prom bude o velikosti 1, změní se proměnná Otočky na

hodnotu od -50 do 50. To zařídí, aby pokaždé robot ujel jinou část dráhy a to buď dopředu, nebo zpětně. Pokud se proměnná Prom nastaví na hodnotu 2, robot se otočí doprava o 5° až o 20°, podle náhodně vygenerovaného stupně. Poslední možností je, že se proměnná Prom nastaví na hodnotu 3. V tomto případě by robot ujel část dráhy zatáčející doleva. Dráha by byla dlouhá od 5 cm do 15 cm, podle vygenerovaných centimetrů. Po vykonání jednoho ze tří příkazů robot vyčká 1 vteřinu a celý cyklus se opět opakuje.

**Shrnutí:** V úloze si můžete „pohrát“ s různými možnostmi pohybu robota. Zkuste rozšířit program o další možné jízdy, které může robot učinit. Uvědomte si, proč je použitá proměnná Otočky. Proč není jen jednoduše v pravém a levém motoru nastaveno vygenerování náhodného stupně otočení? Je to z toho důvodu, že by pravděpodobně tyto stupně nebyli stejné. V tom případě by robot nejel dopředu či dozadu, ale nějakým způsobem by se stočil.




**Obrázek 37: Zvukový senzor - jízda náhodnými směry**

*Zdroj: vlastní*

### Ultrazvukový senzor

Ultrazvukový senzor je velice vhodný pro měření vzdálenosti robota od různých překážek. V následující části jsou zobrazeny 4 úlohy počínaje základní jednoduchou na zobrazení hodnoty, dále rozšiřující úloha kde robot vydává tóny podle různé vzdálenosti překážek. Třetí a čtvrtá úloha jsou takové menší hry s robotem, který se chová stejně jako malé štěně. První štěně je stydlivé, naopak druhé štěně je zlobivé a chce si hrát. Mezi další programy, které se můžou studenti pokusit naprogramovat, by mohla být například úloha, kde se robot otočí dokolečka a posoudí, který objekt je k němu nejblíže, k němu poté dojede. Takovou úlohu jsem již nevytvářela.

**Název:** **Ultrazvukový senzor – zobrazení hodnoty**

**Zadání:** Napište program, který zobrazí na displeji hodnoty ultrazvukového senzoru.

**Popis:** Připojte k robotovi ultrazvukový senzor a měřte pomocí něj vzdálenost těles.

**Moduly:** Při této úloze je zapotřebí ultrazvukový senzor zapojený do portu 4.

**Program:** Program obsahuje nekonečný cyklus, ve kterém je jediný příkaz Povídej. Tento příkaz slouží, aby na displeji řídící jednotky byla zobrazena hodnota. Pomocí buňky Spoj, spojíme hodnotu z ultrazvukového senzoru a „ cm“, které se budou na displeji zobrazovat.

**Shrnutí:** Při zkoušení ultrazvukového modulu je dobré mít nějakou větší věc, která je dobře rozeznatelná. V nejlepším případě stěnu, od které se mohou zvukové vlny odrážet. Na displeji se budou zobrazovat hodnotu od 0 cm až po 255 cm.


**Obrázek 38: Ultrazvukový senzor - zobrazení vzdálenosti**

*Zdroj: upraveno podle [19]*

**Název:** **Ultrazvukový senzor – pípání podle vzdálenosti**

**Zadání:** Vytvořte program, díky kterému bude vydávat řídící jednotka různé zvuky podle naměřené vzdálenosti.

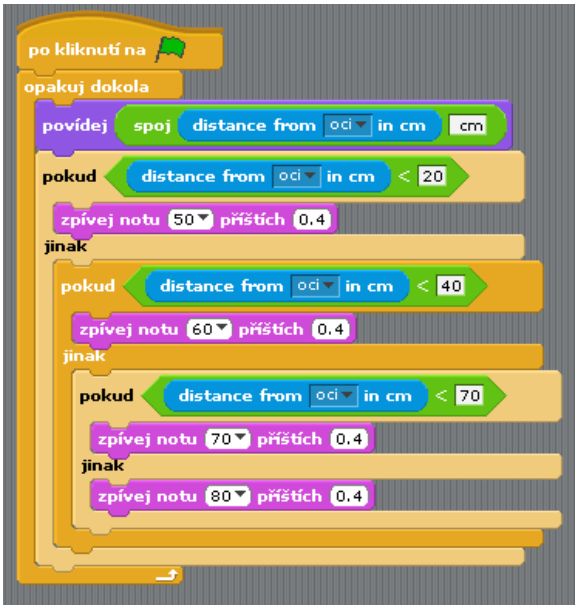
**Popis:** Vytvořte nekonečný cyklus, ve kterém bude řídící jednotka vydávat 4 různé tóny podle toho, v jaké vzdálenosti je objekt umístěn od ultrazvukového

senzoru.

**Moduly:** Při této úloze je zapotřebí ultrazvukový senzor zapojený do portu 4.

**Program:** Na začátku programu se na displeji zobrazí hodnota ultrazvukového senzoru. Poté je zde podmínka, že pokud je hodnota ultrazvukového senzoru menší než 20 cm, tak řídící jednotka vydá tón, který je v programu uváděn jako hodnota 50. Je zde další podmínka, která je splněna, pokud je hodnota ultrazvukového senzoru menší než 40 cm. V tom případě řídící jednotka vydá tón s hodnotou 60. Poté následuje další podmínka, pokud je ultrazvukový senzor udává vzdálenost menší než 70 cm, řídící jednotka vydá tón s hodnotou 70. V konečné řadě, pokud je vzdálenost vyšší než 70 cm, řídící jednotka vydá tón na stupnice 80. Tyto úkony se provádějí neustále dokola. Podle vzdálenosti se mění i stupnice tónu vydaného řídící jednotkou.

**Shrnutí:** V úloze řídící jednotka vydává různé tóny, podle hodnoty ultrazvukového senzoru. Rozmezí jsem ponechala na 4 stupních. Je však možné program rozšířit o více tónů podle vzdálenosti.




**Obrázek 39: Ultravukový senzor - pípání podle vzdálenosti**

*Zdroj: vlastní*

**Název:** **Ultrazvukový senzor – hra - Shy puppy (Stydlivé štěně)**

**Zadání:** Vytvořte program, při kterém si robot od vás bude držet určitou vzdálenost.

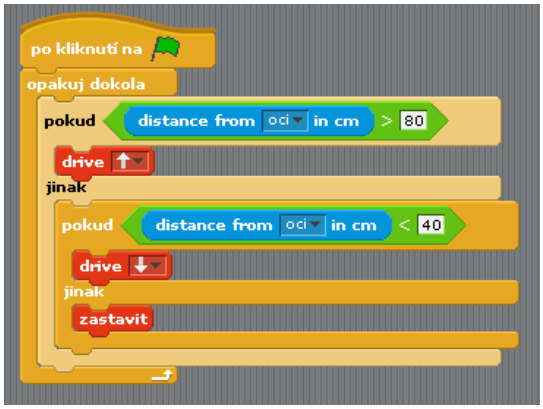
**Popis:** Pokud bude robot moct blízko, popojede kousek dozadu, naopak pokud bude robot moc daleko, přijede blíž. To udělá i v případě, že k němu přistoupíte či od

něho odstoupíte.

**Moduly:** Při této úloze jsou zapotřebí 3 moduly a to 2 servomotory a ultrazvukový senzor. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B. Ultrazvukový senzor je zapojen do portu 4.

**Program:** Program obsahuje nekonečný cyklus, ve kterém podle hodnoty ultrazvukového senzoru se robot hýbe dopředu, do zadu a nebo stojí. První podmínka je, že pokud je vzdálenost větší než 80 cm, robot přijde blíž. Ve druhé podmínce se ptáme, zdali je vzdálenost menší než 80 cm. Pokud ano, robot naopak popojede od nás dál. Pokud bude robot od nás ve vzdálenosti 40 až 80 cm, nebude se hýbat. Čeká, dokud se nepřiblížíme či neoddálíme my.

**Shrnutí:** Uznávám, že úloha Shy puppy je zábavná. Doporučuji pro tuto úlohu použít např. velký sešit, od kterého se ultrazvukové vlny dobře odrážejí.




**Obrázek 40: Ultrazvukový senzor - hra Stydlivé štěně**

*Zdroj: upraveno podle [19]*

**Název:** **Ultrazvukový senzor – hra – Zlobivé štěně**

**Zadání:** Vytvořte program, kde se robot bude chovat jako zlobivé štěňátko. Když od něj budete hodně daleko, přijede s velkou rychlostí blíž. Pokud od něj budete méně daleko, přijede také blíž, ale už ne s tak velkou rychlostí. Pokud u něj naopak budete moc blízko, odjede dál.

**Popis:** Program bude obsahovat nekonečný cyklus, ve kterém bude několik podmínek na určení vzdálenosti robota. Při velké vzdálenosti pojede kupředu hodně rychle, při menší vzdálenosti pojede středně rychle kupředu, při optimální vzdálenosti zůstane stát a při blízké vzdálenosti odjede naopak kus zpět.

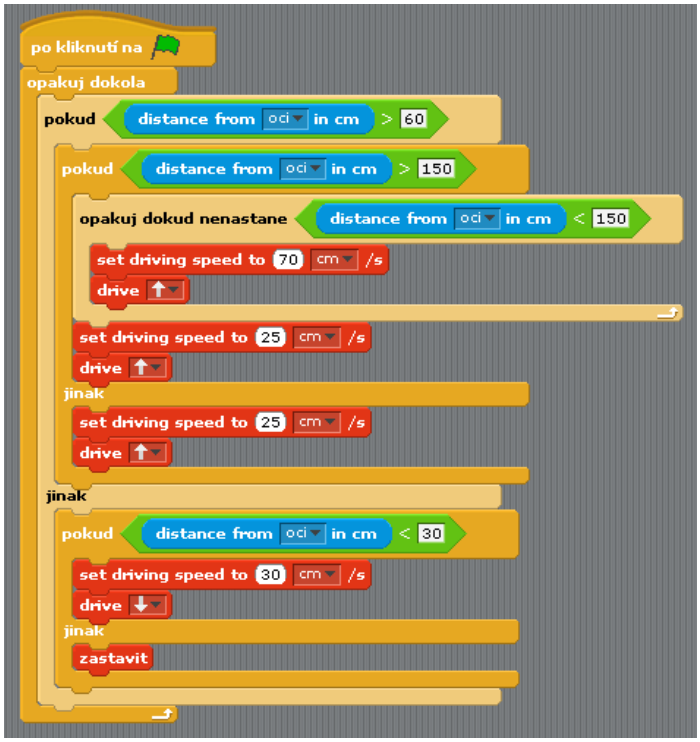
**Moduly:** Při této úloze jsou zapotřebí 3 moduly a to 2 servomotory a ultrazvukový

senzor. Levý servomotor je zapojen do portu C, a pravý servomotor je zapojený do portu B. Ultrazvukový senzor je zapojen do portu 4.

**Program:** Program obsahuje nekonečný cyklus, ve kterém jsou všechny úkony zapouzdřeny. Optimální vzdálenost robota je mezi 30 a 60 centimetry. První podmínkou v cyklu je, jestli je vzdálenost robota větší než 60 cm. Pokud ano, naskytne se další otázka. Je vzdálenost robota od překážky větší než 150 cm? Pokud je vzdálenost větší než 150 cm, robot se rozjede rychlostí 70 cm/s kupředu a pojede touto rychlostí tak dlouho, dokud nebude k překážce blíže než 150 cm. V případě že od překážky není blíže než 150 cm a případně, že tato podmínka vůbec nebyla splněna, pokračuje robot rychlostí 25 cm/s kupředu.

Na začátku programu jsme zjišťovali, zdali je robot vzdálen více jak 60 cm. Pokud tato podmínka splněná není, zajímá nás, zda je robot blíže jak 30 cm, v tom případě by se rozjel směrem dozadu rychlosti 30 cm/s, v opačném případě by byl optimálně daleko a zůstal by stát.

**Shrnutí:** Úloha Zlobivé štěně je rozšířením úlohy Shy puppy o podmínky, které určují rychlost jízdy robota. Věřím, že realizace úlohy Zlobivé štěně bude i pro ostatní tak zajímavá, jako byla pro mne. Učení formou zábavy dá člověku je opravdu velmi efektivní.




**Obrázek 41: Ultrazvukový senzor - hra Zlobivé štěně**

*Zdroj: vlastní*

## 4.2 Robotické úlohy

### 4.2.1 Jízda po černé čáře

Pro jízdu po černé čáře potřebujeme mít k robotovi připojený světelný senzor. Aby se robot pohyboval po vyznačené cestě, zjišťuje přechod mezi bílou barvou plátna a černou barvou čáry. Pro správnou funkčnost robota je lepší využít tyto dvě kontrastní barvy, které se dobře rozeznávají.

**Název:** **Jízda po černé čáře**

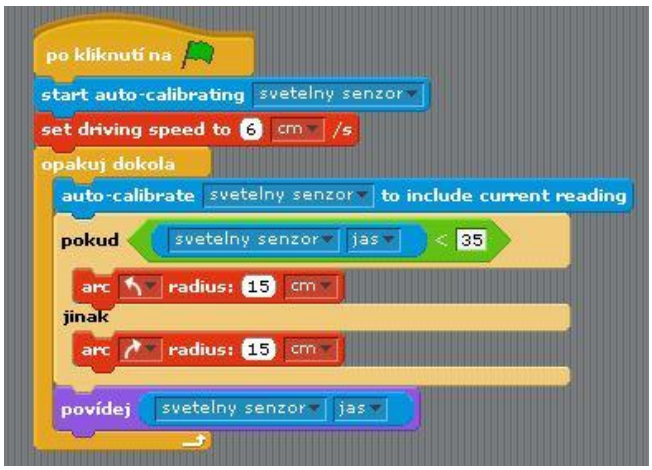
**Zadání:** Vytvořte program, díky kterému bude robot sledovat černou čáru.

**Popis:** Prostřednictvím světelného senzoru se bude robot držet černé čáry. Pojede při pravé straně čáry a podle přechodu zda je na bílé či černé bude postupně zatáčet ze strany na stranu. Tím se bude posouvat kupředu.

**Moduly:** Při této úloze jsou zapotřebí 3 moduly a to 2 servomotory a světelný senzor. Levý servomotor je zapojen do portu C, pravý servomotor je zapojený do portu B. Světelný senzor je zapojen do portu 1.

**Program:** Na začátku programu je třeba auto kalibrovat světelný senzor. Nastavíme rychlost motoru na 6 cm/s. Dále bude již nekonečný cyklus, ve kterém se vždy na začátku překalibruje světelný senzor. Je zde jednoduchá podmínka na to, zda pojede robot doprava či doleva. Směr bude určen podle světelného senzoru, jestli zachycuje černou či bílou. Pokud senzor zachytí černou barvu, začne zatáčet doprava. Pokud senzor zachytí bílou barvu, začne zatáčet zpět doleva. Tímto způsobem, že robot neustále zatáčí na obě strany, se pohybuje kupředu.

**Shrnutí:** Vyzkoušejte jízdu po rovné čáře, po „vlnité“ čáře i jízdu po kruhu. Dále vyzkoušejte změnit rychlost na 3 cm/s, 7 cm/s, 15 cm/s a 20 cm/s. Jak se robot při takových rychlostech chová? Je jízda moc pomalá či naopak se již nestíhá řídit podle černé čáry a ujede někam úplně pryč?




**Obrázek 42: Jízda po černé čáře**

*Zdroj: vlastní*

### 4.2.2 Orientace v prostoru (OVP)

**Název:** **OVP – jízda ke zdi a zastavit**

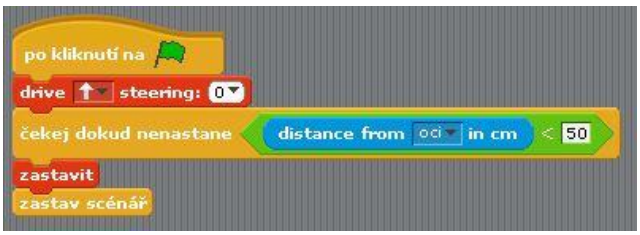
**Zadání:** Napište program, ve kterém robot pojede rovně a jakmile dojede ke zdi, zastaví. Tím program končí.

**Popis:** Po spuštění programu se robot rozjede a pojede rovně. Dokud se nedostane do určité vzdálenosti od překážky, bude stále pokračovat, poté zastaví. Použijte na určení vzdálenosti ultrazvukový senzor.

**Moduly:** Při této úloze jsou zapotřebí 3 moduly a to 2 servomotory a ultrazvukový senzor. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B. Ultrazvukový senzor je zapojen do portu 4.

**Program:** Program je zadán tak, že po jeho spuštění se robot rozjede vpřed a jede do té doby, než hodnota ultrasonického senzoru není menší než 50 cm. Poté robot zastaví a program končí.

**Shrnutí:** Na tomto jednoduchém příkladě si lze znázornit robota a jeho vnímání prostoru. Ať ho pošlete kterýmkoli směrem, vždy dojede k nějaké překážce a před ní se zastaví. Vyzkoušejte upravit hodnotu vzdálenosti, kterou musí robot splňovat pro zastavení. Co se stane, když bude hodnota vzdálenosti moc malá? Robot nestačí zastavit dostatečně rychle a narazí do překážky.




**Obrázek 43: OVP - jízda ke zdi a zastavit**

*Zdroj: vlastní*

**Název:** **OVP – jízda ke zdi se snižováním rychlosti**

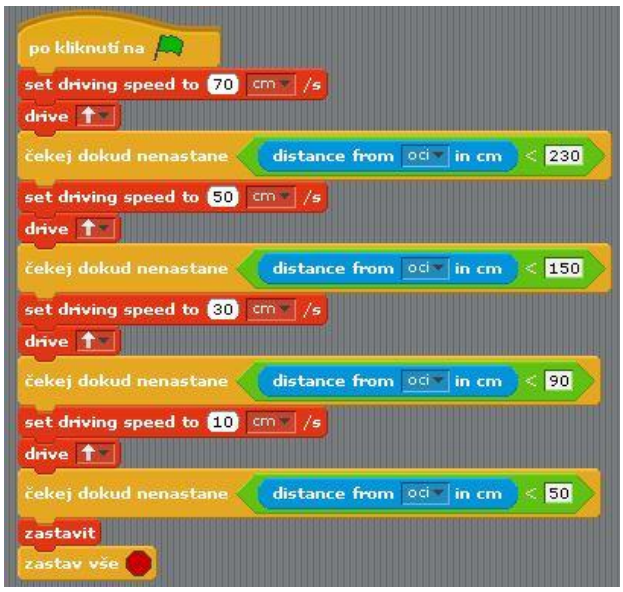
**Zadání:** Vytvořte program, kde robot při jízdě k překážce bude postupně snižovat rychlost.

**Popis:** Nejdříve nastavte robota na vysokou rychlost. Postupně dávejte podmínku, že když se dostane robot do určité vzdálenosti od překážky, tak se sníží jeho rychlost.

**Moduly:** Při této úloze jsou zapotřebí 3 moduly a to 2 servomotory a ultrazvukový senzor. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B. Ultrazvukový senzor je zapojen do portu 4.

**Program:** Nastavte na začátku programu rychlost na 70 cm/s. Robot se rozjede a ultrazvukový senzor je zapnutý. Až se robot dostane na vzdálenost od překážky menší než 230 cm, snižte rychlost na 50 cm/s. Dalším krokem je vzdálenost menší než 150 cm, potom snižte rychlost robota na 30 cm/s. Až se robot dostane k překážce blíže než 90 cm, snižte rychlost na 10 cm/s. Pokud bude rychlost menší než 50 cm, dejte příkaz Zastavit. Poté ukončete program.

**Shrnutí:** Jízda se snižováním rychlosti se dá využít i v jiných úlohách. Především pokud robot jezdí velké vzdálenosti, je lepší pro úsporu času nastavit vyšší rychlost jízdy. Naopak na krátké vzdálenosti se více hodí menší rychlost jízdy. Musíme také myslet na to, že při hodně velké rychlosti jízdy robot okamžitě nezastaví. Vzhledem k úměře jeho rychlosti je i úměrná jeho doba dojezdu. Stejně jako u auta, které když jede hodně rychle, má delší brzdnou dráhu.




**Obrázek 44: OVP - jízda ke zdi se snižováním rychlosti**

*Zdroj: vlastní*

**Název:** **OVP – vyhýbání se překážkám**

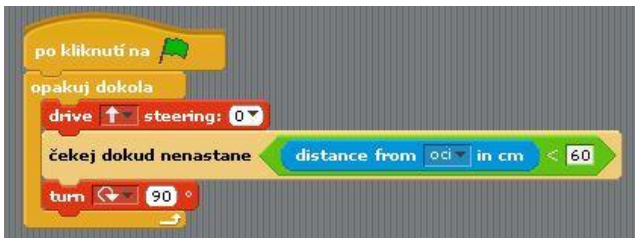
**Zadání:** Vytvořte program, při kterém se robot bude vyhýbat překážkám.

**Popis:** Vytvořte nekonečný cyklus, kde robot pojede rovně až do doby, kdy bude vzdálenost od překážky menší než 60 cm. Poté se robot otočí a bude pokračovat v jízdě.

**Moduly:** Při této úloze jsou zapotřebí 3 moduly a to 2 servomotory a ultrazvukový senzor. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B. Ultrazvukový senzor je zapojen do portu 4.

**Program:** Celý program je uzavřen do nekonečného cyklu. Robot je udán do pohybu rovně. Poté je zde podmínka, která dokud se nesplní, robot stále pojede rovně. Podmínka je, že vzdálenost překážky musí být menší jak 60 cm. Pokud je tato podmínka splněna robot se otočí doprava o 90° a pokračuje v jízdě.

**Shrnutí:** Díky této jednoduché úloze se robot vyhýbá všem překážkám a do žádné nenarazí. Ve výsledku se ale točí stále v kruhu doprava. I tak je tato úloha užitečná pro orientaci v prostoru. Upravte úlohu tak, aby se robot při velké vzdálenosti pohyboval rychleji a opět zpomaloval při přibližování se k překážce.




**Obrázek 45: OVP - vyhýbání se překážkám**

*Zdroj: vlastní*

**Název:** **OVP – vyhýbání se překážkám náhodnými směry**

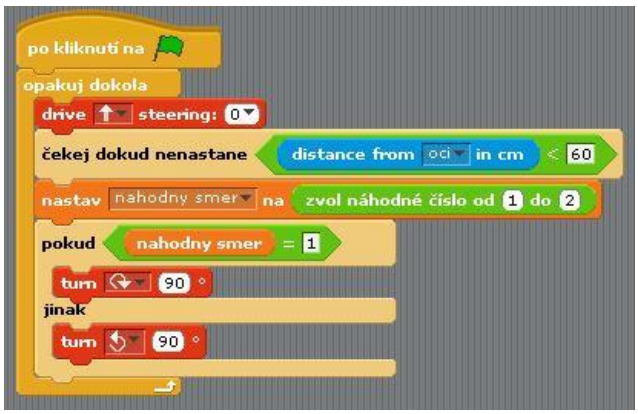
**Zadání:** Vytvořte program, kde se robot bude vyhýbat překážkám. Pokud dojede k nějaké překážce, otočí se náhodným směrem jinam a bude pokračovat dál.

**Popis:** Po spuštění programu se robot rozjede rovně. Když dojede k překážce, vygeneruje se náhodné číslo, podle kterého poté bude robot pokračovat buď doprava, nebo doleva.

**Moduly:** Při této úloze jsou zapotřebí 3 moduly a to 2 servomotory a ultrazvukový senzor. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B. Ultrazvukový senzor je zapojen do portu 4.

**Program:** Program obsahuje nekonečný cyklus, ve kterém jsou všechny příkazy. Je třeba si vytvořit proměnnou Náhodný směr. Na začátku programu je příkaz pro robota na jízdu vpřed. Dále zde je podmínka, že dokud vzdálenost od překážky nebude menší jak 60 cm, tak robot bude stále pokračovat v jízdě. Pokud vzdálenost bude menší jak 60 cm, do proměnné Náhodný směr se vloží vygenerované číslo od 1 do 2. Podle vygenerovaného čísla robot zatočí buď doprava o 90° a nebo doleva o 90°. Poté bude opět pokračovat rovně

**Shrnutí:** Podle tohoto programu se robot orientuje v prostoru a vyhýbá se překážkám. Pokud na nějakou narazí, vyhne se jí pomocí náhodného algoritmu. Rozšiřte tento program o další možné směry jízdy. Také rozšiřte program o různé rychlosti jízdy.




**Obrázek 46: OVP - vyhýbání se překážkám náhodnými směry**

*Zdroj: vlastní*

### 4.2.3 Jízda ovládaná tlačítky

**Název:** **Jízda ovládaná tlačítky**

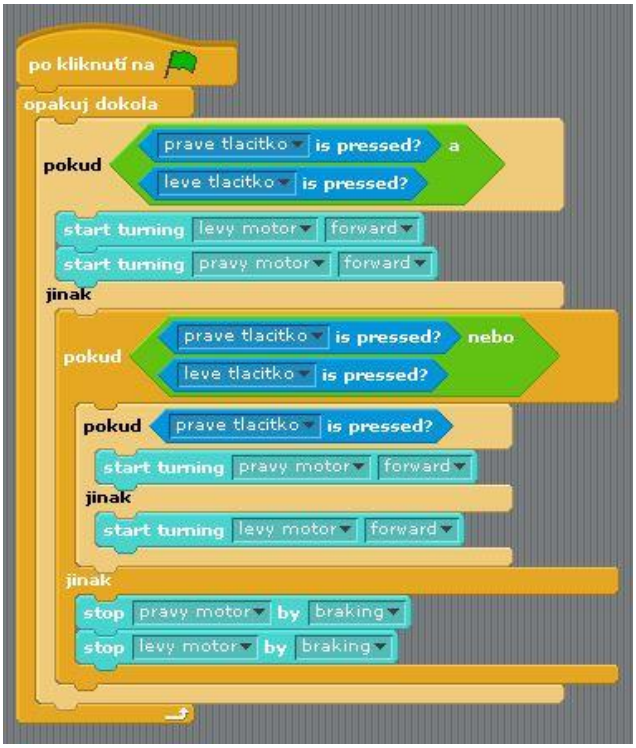
**Zadání:** Vytvořte program, kde budete robota ovládat dotykovými senzory tak, aby jel doleva, doprava a rovně. Použijte při tom dva dotykové senzory.

**Popis:** Vytvořte program, kde pokud stisknete obě tlačítka, robot pojede rovně, pokud stisknete tlačítko v levé ruce, robot bude zatáčet doleva a pokud stisknete tlačítko v pravé ruce, robot bude zatáčet doprava. Pokud žádné tlačítko nebude stisknuté, robot se nebude hýbat.

**Moduly:** Při této úloze je zapotřebí 4 modulů. Dva servomotory a dva dotykové senzory. Levý servomotor je zapojen do portu C a pravý servomotor je zapojen do portu B. Levý dotykový senzor je zapojen do potru 3 a pravý dotykový senzor je zapojen do portu 2.

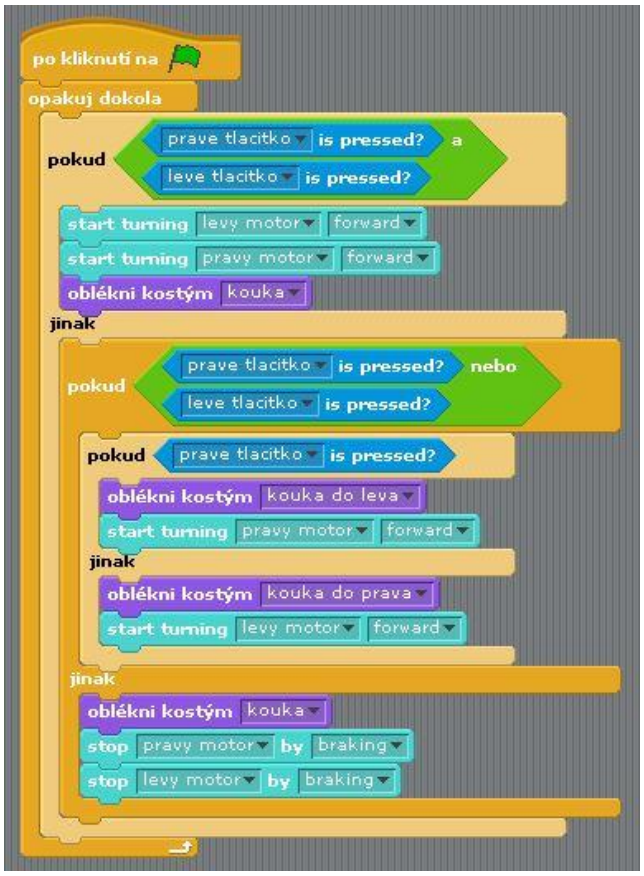
**Program:** Program bude obsahovat nekončený cyklus, ve kterém bude základní podmínka, zda jsou obě tlačítka stlačená. Pokud ano, robot se vydá kupředu, pokud ne, je zde další otázka, zda je alespoň jedno tlačítko stlačené. Pokud ano, a pokud je to pravé tlačítko, robot se vydá doprava, pokud ano a není to pravé tlačítko, robot se vydá doleva. V případě že není stlačeno žádné tlačítko, robot se zastaví a stojí, dokud opět některé tlačítko nezmáčknete.

**Shrnutí:** Tímto způsobem můžete ovládat robota pomocí dvou připojených dotykových senzorů. Rozšiřte program o možnost zobrazení „očí“ na displeji, které se budou dívat směrem jízdy (viz Obrázek 48).




**Obrázek 47: Jízda ovládaná tlačítky**

*Zdroj: vlastní*




**Obrázek 48: Jízda ovládaná tlačítky + koukání různými směry**

*Zdroj: vlastní*

# ZÁVĚR

Cílem bakalářské práce byla tvorba úloh pro roboty od společnosti Lego. Studenti docházející na předmět Základy algoritmizace se mohou pomocí těchto úloh naučit pracovat s roboty a následně vytvářet své vlastní úlohy.

Nejprve bylo pojednáno o základech algoritmizace a o jejich hlavních prvcích, které jsou důležité při tvorbě úloh.

Následuje seznámení se stavebnicí Lego Mindtorms, která byla použita při tvorbě této bakalářské práce. Jsou zde charakterizovány její jednotlivé prvky, především ty, z kterých byli sestaveni roboti. Také je v této kapitole bližší seznámení se stavebnicí Zelené město a jejími úlohami, které vytvořila firma Lego. Stavebnici jsem sestavila a vyzkoušela, jakým způsobem se udávají do chodu její části. Tvorba všech úloh by bohužel zabrala příliš mnoho času, proto jsem zvolila pouze jednu úlohu, a to Jízdu po černé čáře.

Třetí kapitola je o programu Enchanting, který jsem používala na tvorbu úloh. Zaměřila jsem se především na charakteristiku editoru malování a editoru pro vytvoření nového bloku, které jsou později použity v konkrétních úlohách.

Bližší seznámení s programem Enchanting je až v poslední kapitole prostřednictvím vytvořených úloh. Každá úloha obsahuje její popis a program, který je následně zobrazen na obrázku. První úlohy slouží k seznámení s robotem a jeho základními funkcemi. Studenti se musí nejprve naučit pohybovat se s roboty a teprve až poté zkoumat chování pomocí senzorů. V druhé části kapitoly jsou blíže popsány jednotlivé robotické úlohy. Všechny úlohy jsem samostatně vytvořila a následně implementovala do řídící jednotky NXT.

Závěrem bych chtěla říci, že použití vývojových diagramů při programování je opravdu výhodné, ne-li dokonce nezbytné. Po vlastní zkušenosti, kdy jsem se jeden program snažila zprovoznit metodou pokus omyl a i po 2 hodinách jsem stále nenašla logickou chybu. Poté jsem se rozhodla nakreslit si vývojový diagram, a po 3 minutách bylo jasné, v „čem byl háček“. Pokud bych si nakreslila vývojový diagram nejdříve, neztratila bych zbytečně čas. To je i účelem předmětu Základy algoritmizace. Pokud si studenti nakreslí vývojové diagramy, programování pro ně poté bude „hračkou“.

# POUŽITÁ LITERATURA


[1] TAUFER I. *Algoritmy a algoritmizace - vývojové diagramy.* Pardubice: Univerzita Pardubice, 2009. ISBN 978-80-7395-182-5.

[2] ČSN ISO 5807. *Zpracování informací. Dokumentační symboly a konvence pro vývojové diagramy toku dat, programu a systému, síťové diagramy programu a diagramy zdrojů systému.* Praha : Český normalizační institut, 1995.

[3] PŠENČÍKOVÁ, Jana. *Algoritmizace.* 1. vydání. Kralice na Hané : Computer Media, 2007. 128 s. ISBN: 80-86686-80-9.

[4] MILKOVÁ, Eva. *Algoritmy: objasnění, procvičení a vizualizace základních algoritmických konstrukcí.* Praha : Alfa, 2008. 114 s. ISBN: 978-80-87197-10-3.

[5] LEGO education [online]. 2013 [cit. 2013-04-03]. Dostupné z: <http://education.lego.com/en-us/products/>.

[6] Lego Mindstorms NXT [online]. 2012 [cit. 2013-04-13]. Dostupné z: <http://mindstorms.lego.com/en-us/default.aspx>.

[7] Scorpion – 9695 Scorpion [pdf]. 2010. Dostupné z: <http://education.lego.com/en-us/products/>.

[8] LEGO MINDSTORMS NXT – Robotické vzdělávání [online]. 2012 [cit. 2013-04-03]. Dostupné z: <https://lego.zcu.cz/web>.

[9] *Jádro ARM7, které se může stát legendou jako 8051* [online]. 2006 [cit. 2013-04-13]. Dostupné z: <http://noel.feld.cvut.cz/vyu/scs/prezentace2006/ARM7>.

[10] GRIFFIN, Terry. *The art of LEGO Mindstorms NXT-G programming*. San Francisco, CA: No Starch Press, 2010, xvii, 200 p. ISBN 15-932-7218-9. Dostupné z. <http://www.kramirez.net/Robotica/Material/Libros/The%20Art%20of%20LEGO%20MINDSTORMS%20NXT-G%20Programming.pdf>.

[11] Hitechnic [online]. 2001-2012 [cit. 2013-04-03]. Dostupné z: <http://www.hitechnic.com/>.

[12] Sensors. Vernier [online]. 2013 [cit. 2013-04-03]. Dostupné z: <http://www.vernier.com/products/sensors/>.

[13] Green City – 9594 Green city [pdf]. 2011. Dostupné z: <http://education.lego.com/en-us/products/>.

[14] Lego Green City – *2009594 Activity Pack for Green City for Mindstorms* [cd]. 2011. Dostupné z: <http://education.lego.com/en-us/products/>.

[15] BLACKMORE, Clinton. Enchanting [online]. 2011-2013 [cit. 2013-04-03]. Dostupné z: <http://enchanting.robotclub.ab.ca/tiki-index.php>.

[16] Scratch [online]. 2012 [cit. 2013-04-03]. Dostupné z: http://scratch.mit.edu/

[17] SNAP! (BYOB) [online]. 2011 [cit. 2013-04-03]. Dostupné z: http://snap.berkeley.edu/

[18] LeJos, Java for Lego Mindstorms [online]. 2009 [cit. 2013-04-03]. Dostupné z: <http://lejos.sourceforge.net/>.

[19] Enchanting Cards v1 [pdf]. 2013. Dostupné z: <http://enchanting.robotclub.ab.ca/tiki-index.php>.