Univerzita Pardubice

Fakulta ekonomicko-správní

Ústav systémového inženýrství a informatiky

Pohyb robota v prostoru

Tomáš Havrda

Bakalářská práce
2014

Univerzita Pardubice
Fakulta ekonomicko-správní
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš Havrda**
Osobní číslo:	**E11200**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Informatika ve veřejné správě**
Název tématu:	**Pohyb robota v prostoru**
Zadávající katedra:	**Ústav systémového inženýrství a informatiky**

Zásady pro vypracování:

Cílem práce je navrhnout metody určené pro pohyb robota v prostoru. Robot se bude rozhodovat na základě rozmístění objektů v daném prostoru. Práce bude probíhat na několika robotických stavebnicích firmy Lego.
Osnova práce:
1. Složení robotických stavebnic
2. Tvorba algoritmů pro pohyb robota
3. Implementace algoritmů

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

1. TAUFER I. Algoritmy a algoritmizace - vývojové diagramy. Pardubice: Univerzita Pardubice, 2009. ISBN 978-80-7395-182
2. PŠENČÍKOVÁ, Jana. Algoritmizace. 1. vydání. Kralice na Hané : Computer Media, 2007. 128 s. ISBN: 80-86686-80-9.
3. MILKOVÁ, Eva. Algoritmy: objasnění, procvičení a vizualizace základních algoritmických konstrukcí. Praha : Alfa, 2008. 114 s. ISBN: 978-80-87197-10-3.

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

Datum zadání bakalářské práce: **1. října 2013**
Termín odevzdání bakalářské práce: **30. dubna 2014**

L.S.

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

prof. Ing. Jan Čapek, CSc.
vedoucí ústavu

V Pardubicích dne 1. října 2013

PROHLÁŠENÍ

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

Byl jsem seznámen 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. 2014 Tomáš Havrda

PODĚKOVÁNÍ:

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

ANOTACE

Obsahem této bakalářská práce je vysvětlení základních principů algoritmizace, algoritmů a stavba robota ze stavebnice Lego. Dále je návrh a implementace vytvořeného kódu v programu Enchanting. Tištěná práce je doplněna o CD s kompletním navrhnutým kódem v PDF formátu a ve formátu programu Enchanting

KLÍČOVÁ SLOVA

Algoritmizace, Enchanting, Lego, Robot

TITLE

The movement of the robot in space

ANNOTATION

The content of this work is to explain the basic principles of algorithms, algorithms and building a robot from Lego. Furthermore, the design and implementation of this code in the Enchanting. The printed work is completed by nominated CD with the complete code in the PDF format and the format of the program Enchanting

KEYWORDS

Algorithmics, Enchanting, Lego, Robot

OBSAH

ÚVOD **10**

1 ALGORITMY A ALGORITMIZACE **11**

1.1 POŽADAVKY NA ALGORITMUS 11
1.2 MOŽNOSTI ZÁPISU ALGORITMU 11
1.3 VÝVOJOVÉ DIAGRAMY 12
1.4 ZNAČKY VÝVOJOVÝCH DIAGRAMŮ 12
1.4.1 Mezní značky *12*
1.4.2 Sekvenční bloky *13*
1.4.3 Větvení *15*
1.4.4 Další značky *16*
1.5 CYKLY 17
1.5.1 Cyklus s pevným počtem opakování *17*
1.5.2 Cyklus s podmínkou na počátku *18*
1.5.3 Cyklus s podmínkou na konci *19*

2 LEGO MINDSTORMS **21**

2.1 ZÁKLADNÍ POPIS LEGO MINDSTORMS 21
2.2 LEGO MINDSTORMS NXT 21
2.2.1 Řídící jednotka *21*
2.2.2 Interaktivní servomotory *24*
2.2.3 Senzory *25*

3 ENCHANTING **28**

3.1 POPIS PROGRAMU ENCHANTING 28
3.2 FIRMWARE 28
3.3 PODMÍNKOVÉ BLOKY V ENCHANTINGU 28
3.4 CYKLICKÉ BLOKY V ENCHANTING 29

4 POPIS ÚLOHY A KONSTRUKCE ROBOTA **32**

4.1 POPIS ÚLOHY 32
4.2 KONSTRUKCE ROBOTA 32
4.3 PROBLEMATIKA OTÁČENÍ 4 KOLOVÝCH VOZIDEL 33
4.4 PROBLEMATIKA POUŽITÍ KOMPASOVÉHO SENZORU 34
4.5 PROBLEMATIKA URČENÍ PŘEKÁŽEK 34
4.6 KOMPLEXNÍ POPIS ÚLOHY A PRINCIP PROGRAMU 35

5 NÁVRH A IMPLEMENTACE KÓDU **38**

5.1 SCÉNÁŘ A 38
5.2 SCÉNÁŘ B 44
5.3 SCÉNÁŘ C 48
5.4 SCÉNÁŘ D 49
5.5 SCÉNÁŘ E 51
5.6 SCÉNÁŘ F 51
5.7 SCÉNÁŘ G 52

ZÁVĚR **53**

POUŽITÁ LITERATURA **55**

SEZNAM PŘÍLOH **57**

SEZNAM ILUSTRACÍ

Obrázek 1: Mezní značka – Začátek 12
Obrázek 2: Mezní značka – Konec 13
Obrázek 3: Sekvenční blok – Vstup 14
Obrázek 4: Sekvenční blok – Výstup 14
Obrázek 5: Sekvenční blok – Zpracování 14
Obrázek 6: První možnost zobrazení rozhodovací bloku 15
Obrázek 7: Druhá možnost zobrazení rozhodovací bloku 15
Obrázek 8: Blok příprava 16
Obrázek 9: Spojka 16
Obrázek 10: Podprogram 17
Obrázek 11: Cyklus s pevným počtem opakování s blokem přípravy 18
Obrázek 12: Cyklus s pevným počtem opakování s podmínkou 18
Obrázek 13: Cyklus s podmínkou na počátku 19
Obrázek 14: Cyklus s podmínkou na konci 20
Obrázek 15: Řídící jednotka 22
Obrázek 16: Vnitřní zapojení bloků řídící jednotky 23
Obrázek 17: Interaktivní servomotor 25
Obrázek 18: Světelný senzor 25
Obrázek 19: Zvukový senzor 26
Obrázek 20: Dotykový senzor 26
Obrázek 21: Ultrazvukový senzor 27
Obrázek 22: Kompasový senzor 27
Obrázek 23: Podmínka se dvěma větvemi instrukcí 28
Obrázek 24: Podmínka s jednou větví instrukcí 29
Obrázek 25: Čekající podmínka 29
Obrázek 26: Nekonečný cyklus 29
Obrázek 27: Cyklus s pevným počtem opakování 30
Obrázek 28: Cyklus „opakuj, dokud nenastane“ s podmínkou na počátku 30
Obrázek 29: Cyklus „opakuj, dokud nenastane“ s podmínkou na konci 30
Obrázek 30: Cyklus „opakuj dokola pokud“ s podmínkou na počátku 31
Obrázek 31: Cyklus „opakuj dokola pokud“ s podmínkou na konci 31
Obrázek 32: Model robota 32
Obrázek 33: Model robota 33
Obrázek 34: Odraz ultrazvuku od kolmé překážky 34
Obrázek 35: Odraz ultrazvuku od překážky pod úhlem 35
Obrázek 36: Schéma úlohy 36
Obrázek 37: Scénář A část 1. 38
Obrázek 38: Scénář A část 2. 39
Obrázek 39: Scénář A část 3. 40
Obrázek 40: Scénář A část 4. 41
Obrázek 41: Scénář A část 5. 41
Obrázek 42: Scénář A část 6. 42
Obrázek 43: Scénář A část 7. 42
Obrázek 44: Scénář A část 8. 43
Obrázek 45: Scénář A část 9. 44
Obrázek 46: Scénář B část 1. 44
Obrázek 47: Scénář B část 2. 45
Obrázek 48 Scénář B část 3. 45
Obrázek 49: Scénář B část 4. 47

Obrázek 50: Scénář B část 5. 48
Obrázek 51: Scénář C 49
Obrázek 52: Scénář D 50
Obrázek 53: Scénář E 51
Obrázek 54: Scénář F 52
Obrázek 55: Scénář G 52

ÚVOD

Robotika a aromatizace obecně nás v dnešním světě obklopuje na každém kroku. Jde o nahrazování lidské činnosti nějakým robotickým systémem, jehož provoz bude levnější, rychlejší, spolehlivější nebo zkrátka pro nebezpečné činnosti, kde hrozí možnost zranění, či dokonce ohrožení lidského života. Obory použití jsou prakticky neomezené od průmyslu jako např. svařovací roboti, v domácnostech např. regulace vytápění, či osvětlení. Pro složky IZS (Integrovaný záchranný systém) např. roboti pro prohledávání sutin, zbrojní průmysl např. bezpilotní letouny. V budoucnu nastane ještě užší propojení automatických a robotických systému s lidským životem. Již dnes se např. testují samostatně řízené automobily v reálném provozu.

Cílem práce je navrhnou a realizovat vlastní algoritmy pro pohyb robota v prostoru a posléze je realizovat pomocí programu Enchanting. Závěrem navrhnuté algoritmy implementovat do programovatelné jednotky stavebnice Lego Mindstorms, která bude připojena k vlastnímu návrhu robota totožné značky výrobce.

V práci je první kapitola věnována algoritmizaci a to základním prvkům, co který blok znamená a vysvětlení principu cyklů, bez nichž se žádný smysluplný program neobejde. V další části je stručný vývoj, technické specifikace a popis stavebnice Lego Mindstorms NXT. Dále jsou popsány základní moduly stavebnice, jako je řídící jednotka, servomotory a senzory. Třetí část je věnována programu Enchanting. Je zde základní popis programu a následné vysvětlení typů podmínek, principu a konstrukce možných typů cyklů, které lze v Enchanting použít. Předposlední kapitola je se věnována samotné konstrukci robota, vysvětlení principu úlohy, jejímu řešení a problémům, které nastaly během práce jak na stavbě, tak i při testování kódu a musely být nějakým způsobem vyřešeny. Poslední nejobsáhlejší kapitola je zaměřena na vysvětlení samotného principu kódu robota. Pro lepší pochopení a vysvětlení byly jednotlivé scénáře označeny písemnými znaky. Scénáře jsou doplněny textovými popisem. V příloze na CD je v souboru PDF umístěný kompletní přehled scénářů a také samotný kód v programu Enchanting.

1 ALGORITMY A ALGORITMIZACE

I přes vývoj a pokroky v informačních technologiích a nových metodách zpracování informací je stále aktivní účast uživatele v procesu jejich zpracování nezastupitelná. Počítač nezbavuje uživatele nutnosti přesně formulovat postup řešení problému a určit posloupnost a varianty jednotlivých kroků a řešení. Uživatel musí tedy při využívání počítače vědět, jak skloubit instrukce pro jednotlivé operace, aby počítač poskytl požadovaný výsledek. Znamená to, že nejdříve musíme vědět jako uživatelé, jak danou úlohu řešit a jak postupovat při výpočtu, neboli znát algoritmus jejího řešení.[15]

Algoritmus je ústředním pojmem informatiky. Je to postup skládající se z jednotlivých jednoznačně určených kroků, tzv. příkazů.

1.1 Požadavky na algoritmus

a) **elementárnost**: algoritmus se skládá z konečného počtu jednoduše a snadno realizovatelných kroků;

b) **determinovanost** (určenost): algoritmus musí být přesný, srozumitelný a jednoznačný. V žádném kroku řešení nesmí být pochyby o tom, co je potřeba v dané kroku udělat a jaký bude navazující krok;

c) **konečnost**: algoritmus musí mít konečný počet kroků;

d) **rezultativnost**: algoritmus musí skončit po vykonání konečného počtu kroků a musí poskytnout nějaký výsledek;

e) **masovost** (hromadnost): alogismus by neměl být popisem řešení jediné úlohy, ale celé skupiny úloh.[1], [14], [15]

1.2 Možnosti zápisu algoritmu

a) grafický zápis (vývojový diagram, strukturogram);

b) pseudokód (přirozený jazyk doplněný klíčovými slovy);

c) zápis v libovolném programovacím jazyku.[10]

1.3 Vývojové diagramy

Vývojový diagram je symbolický algoritmický jazyk, který se používá pro názorné zobrazení algoritmu. Je to jedna z nejdokonalejších forem zápisu algoritmů. Používá se zejména při vývoji softwaru:

- jako komunikační prostředek při týmové spolupráci analytiků a programátorů na společném projektu;
- k dokumentačním účelům - vývojový diagram je přehlednější než výpis programu např. v programovém kódu.[11]

Některé softwary dokonce umožnují již programovat přímo pomocí bloků vývojových diagramů, ze kterých se skládá celý algoritmus místo použití programového kódu.

Vývojové diagramy se skládají z jednotlivých symbolů, které jsou mezi sebou spojeny orientovanými čarami tzv. hranami. Jednotlivé hrany jsou označeny šipkou vyjadřující směr orientace neboli postupu algoritmu. Obecně používaný postup psaní značek je odshora dolů a zleva doprava. V některých případech nemůže být tento postup dodržen a proto je pro přehlednost a správnost, dodržování psaní šipek v těchto případech nezbytně nutné.[11]

1.4 Značky vývojových diagramů

1.4.1 Mezní značky

Mezní značka znázorňuje vstup z vnějšího prostředí do programu nebo výstup z programu do vnějšího prostředí, např. začátek nebo konec programu. Používá se také pro začátek či konec podprogramu, což je samostatně zpracované části programu. Mezní značka má vždy jen jednu orientovanou hranu, buď vstupní nebo výstupní.[11]

Začátek

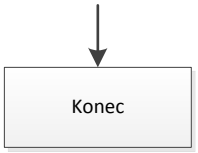
Pokud je mezní značka použita na začátku algoritmu pak, je v ní napsáno *Začátek* nebo *Start* (viz obrázek 1). Do značky nesmí vstupovat žádná hrana, jelikož je použita jako první a musí z ní vystupovat z dolního okraje jedna hrana směrem dolů.[11]

Obrázek 1: Mezní značka – Začátek

Zdroj:[11]

Konec

Pokud je mezní značka použita na konci algoritmu, pak je v ní napsán *Konec* (viz obrázek 2). Do značky musí vstupovat pouze jedna hrana a to do jejího horního okraje směrem ze shora dolů. Dále nesmí z ní vystupovat žádná hrana a za touto značkou už nesmí být žádná další značka.[11]



Obrázek 2: Mezní značka – Konec

Zdroj:[11]

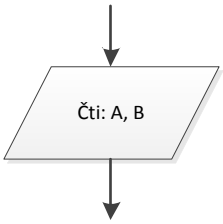
1.4.2 Sekvenční bloky

Sekvenčně bloky jsou použity uvnitř vývojového diagramu a nesmějí být použity jako první ani jako poslední. Označují sekvenční postup algoritmem, to znamená, že je možné se do něho dostat pouze z předchozího prvku a po jeho vykonání lze postoupit pouze na jeden prvek následující. V průběhu sekvenčních bloků nesmí dojít k rozvětvení algoritmu. To znamená, že musí mít pouze jeden vstup a jeden výstup.[11]

Při běhu programu je potřebná komunikace s počítačem. Je zapotřebí, aby se „dovnitř“ dostala data, která program potřebuje ke své činnosti. Může je zadat uživatel z klávesnice, nebo mohou být načtena z datového souboru. Dále aby se nakonec uživatel dozvěděl výsledky zpracování, může si je například nechat zobrazit na obrazovce počítače, vytisknout na tiskárně nebo si je uložit do souboru.[11]

Vstup

Značí načtení dat potřebných pro činnost programu. Uvnitř značky vstupu je napsáno *Čti:* a například nějaká proměnná, která má být načtena (viz obrázek 3).

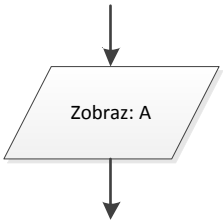


Obrázek 3: Sekvenční blok – Vstup

Zdroj:[11]

Výstup

Značí zobrazení výstupů programu na zobrazovacím zařízení jako je monitor počítače, nebo tisk tiskárnou či práce s diskovou jednotkou. Uvnitř značky je napsáno, buď *Zobraz:*, *Zapiš:*, *Ulož:* nebo *Vytiskni:* dle potřeby, co je potřeba s daty udělat a samozřejmě ještě specifikace dat například pomocí proměnné (viz obrázek 4).[11]

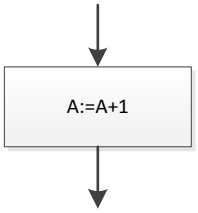


Obrázek 4: Sekvenční blok – Výstup

Zdroj:[11]

Zpracování

Blok zpracování (viz obrázek 5) znárodňuje nějakou činnost programu, během níž dochází k operaci s daty jako například sečtení dvou čísel či nějaké jiné matematické nebo logické operace. V bloku nemusí být zapsána pouze jedna instrukce, ale i více. Každá instrukce však musí být podrobná, že ji lze vykonat najednou a nesmí v sobě skrývat několik dalších operací.[11]



Obrázek 5: Sekvenční blok – Zpracování

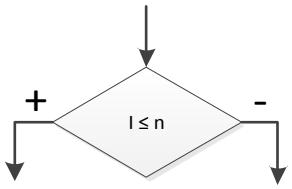
Zdroj:[11]

1.4.3 Větvení

Někdy nelze v algoritmu postupovat pouze sekvenčně, ale je potřeba algoritmus rozvětvit. K větvení dochází na základě podmínky, jejíž splnění či nesplnění určuje směr pokračování algoritmu. Pokud je podmínka splněna, tak program pokračuje jednou větví, a jestliže podmínka není splněna, pokračuje druhou větví.[11]

Rozhodovací blok

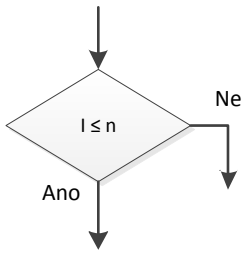
Rozhodovací blok (viz obrázek 6) je určen právě k rozvětvení programu na základě podmínky, která je uvedena uvnitř bloku. Pokud je podmínka splněna, tak program pokračuje větví označenou znaménkem plus + nebo nápisem *ano*. Pokud podmínka není splněna, tak program pokračuje větví označenou znaménkem minus – popřípadě nápisem *ne*. Umístění jestli větev označená znaménkem + bude vlevo a větev označená znaménkem nebo naopak není rozhodující. Pouze záleží na přehlednosti vývojového diagramu.[11]



Obrázek 6: První možnost zobrazení rozhodovací bloku

Zdroj:[11]

Jako další možností zobrazení rozhodovacího bloku je, že výstupní větvíce nemusejí vycházet pouze z bočních vrcholů kosočtverce. Jedna větev může vycházet z dolního vrcholu (viz obrázek 7).[11]



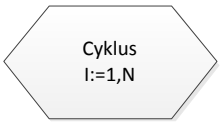
Obrázek 7: Druhá možnost zobrazení rozhodovací bloku

Zdroj:[11]

1.4.4 Další značky

Příprava

Blok příprava (viz obrázek 8) bývá někdy označován jako modifikační symbol. Označuje přípravnou fázi programu a užívá se například pro zahájení cyklu s pevným počtem opakování.[11]

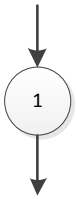


Obrázek 8: Blok příprava

Zdroj:[11]

Spojka

Účelem spojky (viz obrázek 9) je spojit dvě části vývojového diagramu, které nebylo možné nakreslit souvisle například z důvodu přerušení na konci stránky nebo kvůli křížícím se čarám. Spojky musí být označeny stejnými čísly jak konci přerušení, tak i na začátku pokračování daného přerušení.[11]

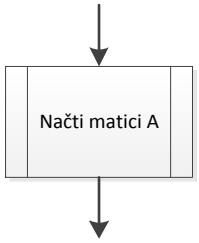


Obrázek 9: Spojka

Zdroj:[11]

Podprogram

Podprogram (viz obrázek 10) znázorňuje část programu, která může být součástí většího množství kroků. Pokud je nějaká část obsažena v algoritmu a tedy i v samotném programu na více místech, tak je výhodné danou část vypracovat samostatně a poté použít právě blok podprogramu, do kterého je vložena daná část algoritmu. Podprogram se pak vloží na daná místa. Podobné je to pokud se často v algoritmech objevuje stejný postup, je opět výhodné jej vypracovat zvlášť a označit ho blokem podprogramu a po té ho vložit do algoritmu na požadovaná místa výskytu. Použitím podprogramu se ušetří práce a algoritmus bude přehlednější. Další výhodou je ze snadnění odstraňovaní chyb, kterou stačí pouze opravit v podprogramu.[11]



Obrázek 10: Podprogram

Zdroj:[11]

1.5 Cykly

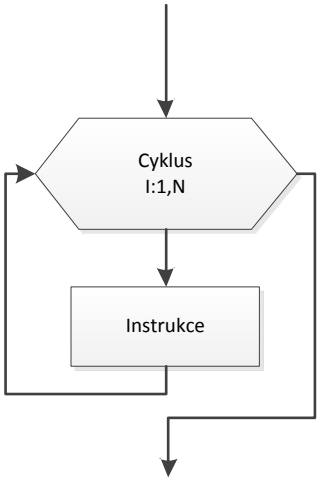
Cykly jsou určeny k opakujícím se operacím. Počet opakování může být předem určen, nebo je odvozen od splnění či nesplnění některé podmínky. Jsou tři základní struktury cyklu:

- cyklus s pevným počtem opakování (s předepsaným počtem opakovaní);
- cyklus s podmínkou na počátku;
- cyklus s podmínkou na konci.[15]

Cyklus kromě určení počtu opakování, aby byl kompletní a smyslu plný musí obsahovat instrukci s nějakou operací, která je obsažena v bloku zpracování.

1.5.1 Cyklus s pevným počtem opakování

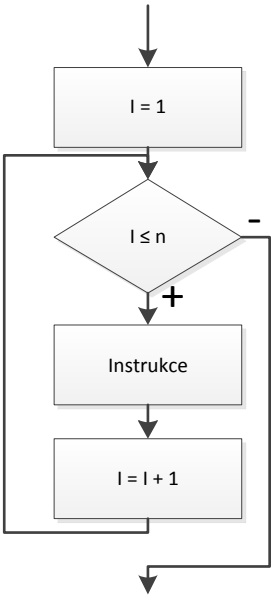
Tento druh cyklu (viz obrázek 11) můžeme sestavit s použitím bloku přípravy. V bloku přípravy je v tomto případě *I* tzv. řídící proměnnou cyklu. Zápis *I:1,N* znamená, že při prvním průchodu cyklem se *I* nastaví na hodnotu 1. Při každém dalším průchodu se *I* zvýší o hodnotu jedna. Což znamená, že při druhém průchodu bude hodnota *I* rovna 2 a při třetím 3 a tak déle až do N-té, kde *I* bude rovno N a cyklus proběhne naposledy a skončí.[15]



Obrázek 11: Cyklus s pevným počtem opakování s blokem přípravy

Zdroj: [15]

Další možností zápisu cyklu s pevným počtem opakování je bez použití bloku přípravy s použitím podmínky (viz obrázek 12).



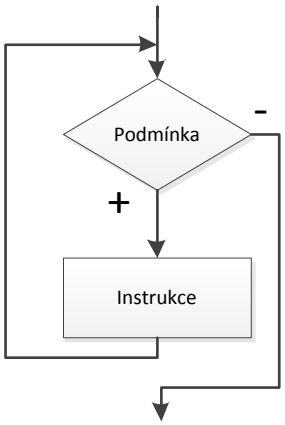
Obrázek 12: Cyklus s pevným počtem opakování s podmínkou

Zdroj: [15]

1.5.2 Cyklus s podmínkou na počátku

Princip cyklu s podmínkou na počátku (viz obrázek 13) je, že nejdříve dojde k vyhodnocení podmínky. Pokud je podmínka splněna, tak se po té provede vyplnění instrukce a pak následuje opět návrat na začátek cyklu a opět k vyhodnocení podmínky.

Činnost se opakuje stále dokola, dokud podmínka nepřestane platit, cyklus se ukončí a pokračuje druhou větví. V podmínce musí dojít pří každém průběhu cyklu ke změně hodnoty logického výrazu podmínky, například k inkrementaci proměnné, aby tak nedošlo k zacyklení. Pro tento druh cyklu je charakteristické, pokud podmínka neplatí hned na začátku, tak se cyklus ukončí, aniž by proběhl alespoň jednou a vůbec nedojde k uskutečnění dané instrukce. Podmínky ukončení cyklu *splnění* + nebo *nesplnění* – mohou být samozřejmě i opačně.[15]

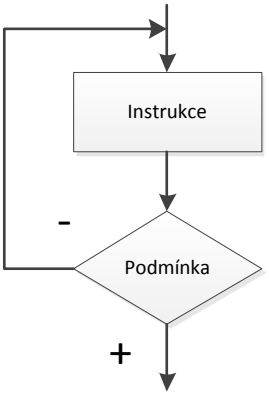


Obrázek 13: Cyklus s podmínkou na počátku

Zdroj: [15]

1.5.3 Cyklus s podmínkou na konci

Princip cyklu s podmínkou na konci (viz obrázek 14) je, že nejdříve dojde k provedení instrukce a až po té k vyhodnocení podmínky. Pokud je podmínka splněna, tak se cyklus ukončí a program pokračuje dál druhou větví. Pokud podmínka není splněna, opakuje se provedení instrukce a po té dojde k vyhodnocení podmínky. Opět v podmínce musí dojít při každém průběhu cyklu ke změně hodnoty logického výrazu podmínky, například k inkrementaci proměnné, aby tak nedošlo k zacyklení. Vzhledem k tomu, že první provedení instrukce není spjato se splněním podmínky, což znamená, že dojde provedení instrukce alespoň jednou. To vede k tomu, že cyklus proběhne alespoň jednou a po té se může na základě podmínky pokračovat anebo ukončit. Podmínky ukončení cyklu *splnění* + nebo *nesplnění* – mohou být samozřejmě i opačně.[15]



Obrázek 14: Cyklus s podmínkou na konci

Zdroj: [15]

2 LEGO MINDSTORMS

2.1 Základní popis LEGO Mindstorms

Lego Mindstorms je řada programovatelných robotických stavebnic vyráběných firmou Lego. První verze Lego Mindstorms byla vydána na trh v roce 1998 pod názvem Robotics Invention System (RIS). Další verze byla vydána v roce 2006 jako Lego Mindstorms NXT. Další verze známá jako Lego Mindstorms NXT 2.0. byla vydána 5. srpna 2009. Nejnovější verze Lego Mindstorms EV3 pochází z 1. srpna 2013. Stavebnice Mindstorms má své počátky vytvoření na MIT Media Laboratory. Stavebnice se používá jako vzdělávací nástroj díky partnerství mezi firmy Lego a MIT Media Laboratory.[7], [8]

Další podrobnější část práce bude zaměřena na verzi Lego Mindstorms NXT, do které spadá edice Mindstorms Education se kterou bylo pracováno, jejíž několik kusů vlastní i Univerzita Pardubice.

2.2 Lego Mindstorms NXT

Samotný set stavebnice se skládá krom standartních stavebních dílů známých z klasické stavebnice Lego, také z modulu:

- Řídící jednotky;
- Interaktivních servomotorů;
- Senzorů;
- Kabeláže.

2.2.1 Řídící jednotka

Hlavním aktivním prvkem stavebnice LEGO NXT je řídící jednotka (viz obrázek 15). Je to počítač, který řídí celé NXT. Řídící jednotka vlastně plní podobnou funkci jako mozek lidského těla na základě smyslových informací, které získá. Ovládá všechny komponenty robota. Umožňuje zobrazovat informace na obrazovce, získává vstupní informace od senzorů, dodává elektřinu do motorů, aby se mohli točit. Řídící jednotka je složena z několika univerzálních i specializovaných modulů, které spolu komunikují pomocí systému několika sběrnic. Vnitřní moduly se starají o jednotlivé funkce řídící jednotky. Důležitou součástí je hlavní procesor od firmy Atmel s architekturou ARM7. Do paměti procesoru jsou nahrána jak data, tak i instrukce s vytvořeným programem, jež bude zpracován a vykonán. Pro nahrání

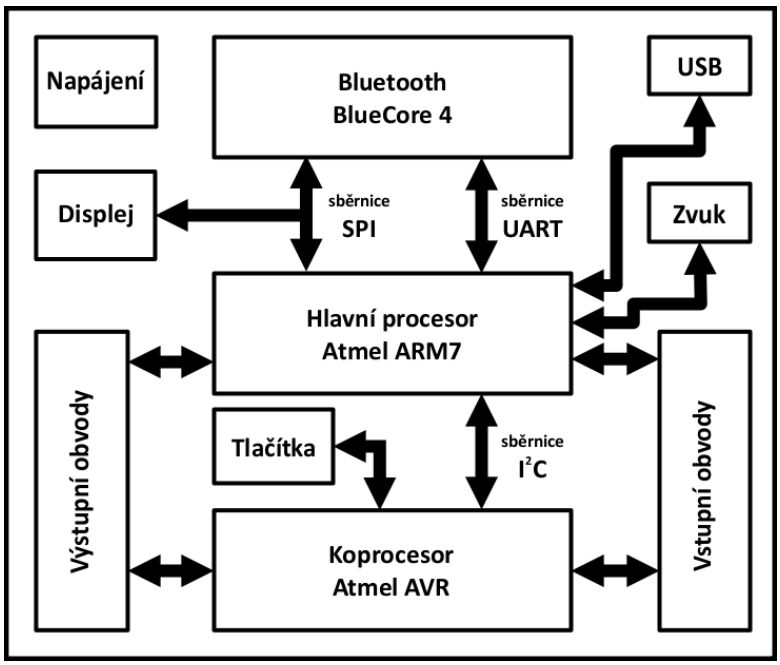
programu do paměti procesoru je k hlavnímu procesoru připojeno USB rozhraní pro komunikaci s počítačem. Jako další komunikační kanál je Bluetooth rozhraní s jehož pomocí lze také zasílat procesoru řídící program nebo dálkově ovládat v reálném čase či dokonce komunikovat s více řídícími jednotkami. Pro komunikaci řídící jednotky s vstupními a vstupními zařízeními slouží konektory, které jsou zabudované na přední a zadní stěně řídící jednotky. Pro vstupní zařízení, což jsou senzory, jsou určené 4 konektory na spodní stěně řídící jednotky. Pro výstupní zařízení jsou určeny 3 konektory umístěné na horní stěně řídící jednotky.[5], [9]

Obrázek 15: Řídící jednotka

Zdroj:[9]

Schéma propojení modulů

Schéma vnitřního vzájemného propojení modulů uvnitř řídící jednotky (viz obrázek 16).



Obrázek 16: Vnitřní zapojení bloků řídící jednotky

Zdroj:[5]

Paměť

Procesor řídící jednotky je vybaven FLASH pamětí s velikostí 256 KB. Je do ní ukládám jak firmware, tak i samotný vytvořený program včetně zvuků a obrázků. Jelikož paměť není moc velká používat obrázky a zvuky by se mělo s mírou, aby nedošlo k zaplnění. Technologie FLASH paměti umožňuje opakovatelné přepisování.[5]

Technické specifikace

Hlavní procesor

- 32bitový mikroprocesor ARM7 (Atmel AT91SAM7S256);
- 48 MHz, 256 KB FLASH, 64 KB RAM;

Přídavný procesor

- 8bitový mikroprocesor AVR (Atmel ATmega48);
- 8 MHz, 4KB FLASH, 512B RAM;

Porty vstupu a výstupu

- 4 vstupní porty (6vodičový LEGO RJ12 konektor, IIC rychlost 9600 bit/s);
- 3 výstupní porty s možností vstupu z kodéru (6vodičový LEGO RJ12 konektor);
- 4 tlačítka na modulu pro ovládání uživatelského rozhraní a programů;
- 8bitový zvukový výstup s reproduktorem (přehrávaná frekvence 2 – 16 kHz) ;
- Černobílý maticový displej 100 x 64 pixelů o velikosti 40,6 x 26 mm (programově použitelné 100 x 60 px);

Možnosti připojení

- USB 2.0 port (12 Mbit/sec);
- Bezdrátová Bluetooth® komunikace v2.0 třídy II, CSR BlueCore 4 v2.0 + EDR, 26MHz;

Operační systém

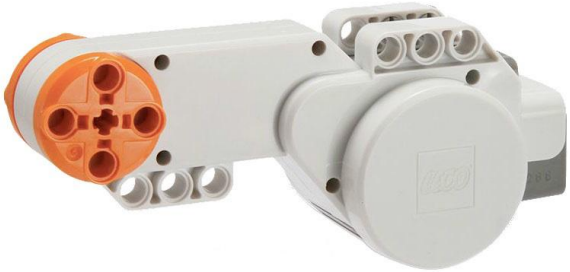
- Operační systém vlastní (proprietální);

Zdroj energie

- 6 AA článků nebo nabíjecí lithiová baterie s transformátorem dle národní normy (EU: 230 V, 50 Hz).[5]

2.2.2 Interaktivní servomotory

Servomotor (viz obrázek 17) je výstupní zařízení poskytující nejen schopnost řídit robota jako auto, ale v různých konfiguracích robota, také servomotory pohybují s různými komponenty či senzory. Servomotor dokonce poskytuje zpětnou vazbu pro řídící jednotku o jeho otáčení díky zabudovanému rotačnímu senzoru. Ten měří otáčení motoru ve stupních s přesností +/- jeden stupeň. Jedna otáčka servomotoru se rovná 360 stupňům.[6], [9]



Obrázek 17: Interaktivní servomotor

Zdroj:[9]

2.2.3 Senzory

Senzory umožňují robotovi nabýt vědomí, stejným způsobem, tak jako zvířata či lidé používají své smyslové schopnosti pro rozhodování. Na základě informací ze senzorů robota, lze naprogramovat řídící jednotku, aby prováděla různé činnosti na základě podnětů ze svého okolí. Po naprogramování robota, je schopen provedení činností v určitou dobu bez zásahu člověka a může jinými slovy fungovat samostatně.[9]

Světelný senzor

Světelný senzor (viz obrázek 18) slouží k rozlišování nejen světla a tmy, ale také k měření intenzity odraženého světla. Díky tomu lze měřit nejen intenzita světla v místnosti, tak i rozpoznávat skupiny barev jednotlivých povrchů. Jak intenzita barevných odstínů, tak i intenzita světla je načítána v procentech.[6]

Obrázek 18: Světelný senzor

Zdroj:[9]

Zvukový senzor

Zvukový senzor (viz obrázek 19) obsahuje mikrofon a slouží k měření intenzity zvuku. Měří intenzitu v decibelech (dB) a poskytuje je také procentuální vyjádření hlasitosti.[6]

Obrázek 19: Zvukový senzor

Zdroj:[9]

Dotykový senzor

Dotykový senzor (viz obrázek 20) funguje jako tlačítko. Jako výstup vrací buď hodnotu Pravda (True) nebo Nepravda (False). Senzor může reagovat různě:

- na zmáčknutí;
- na uvolnění;
- na zmáčknutí a uvolnění.[6]

Obrázek 20: Dotykový senzor

Zdroj:[9]

Ultrazvukový senzor

Ultrazvukový senzor (viz obrázek 21) slouží k měření vzdálenosti. Vzdálenost je měřena v rozsahu 0 až 255 cm s přesností +/- 3 cm. Lze ji měřit i v palcích. Princip senzoru je, že vyšle akustickou vysokofrekvenční vlnu pro lidské ucho neslyšitelnou a přepočítá návrat vlny od odraženého předmětu. Čímž určí vzdálenost předmětu, od kterého byla akustická vlna odražena. Díky této schopnosti může pak robot hledat předměty nebo se jim při jízdě vyhnout.[6]

Obrázek 21: Ultrazvukový senzor

Zdroj:[9]

Kompasový senzor

Kompasový senzor (viz obrázek 22) obsahuje digitální kompas pro měření magnetického pole Země a navrací hodnotu azimutu. Výstup senzoru je rozsah hodnot stupňů 0-359. Kde hodnota 0 stupňů znázorňuje sever. Hodnoty kompasu jsou zaokrouhleny na nejbližší celé číslo a obnova probíhá 100x za sekundu.[4]

Obrázek 22: Kompasový senzor

Zdroj:[3]

Světelný, zvukový, dotykový a ultrazvukový senzor patří do základních modulů stavebnice, ale kompasový senzor patří již mezi tzv. rozšiřující moduly.

3 ENCHANTING

3.1 Popis programu Enchanting

Enchanting je open-source grafické programovací prostředí pro programovaní robotů od společnosti LEGO Mindstorms NXT. Enchanting je založen na celosvětově rozšířeném programu Scratch vyvinutý na Massachusetts Institute of Technology MIT v roce 2007. Jedná se o poměrně snadný způsob, který umožnuje uživatelům prakticky s jakýmikoliv programovacími zkušenostmi a věkem experimentovat s koncepty plně univerzálního programování, použitím spojování grafických programovacích bloků k vládání obrázků, hudby a zvuků. Některé bloky byly z programu Scratch odstraněny jelikož nebyly potřeba nebo nešly použít. Dále je ještě založen na programu BYOB/Snap!. Enchanting je snadné a zábavné prostředí určené pro každého, kdo chce rychle a snadno vytvářet program bez znalosti programovacích jazyků, psaní kódů, sestavování a odstraňování chyb syntaxe dokud program nebude fungovat. Je to skvělá příležitost pro získání zkušeností do začátku pro toho, kdo se začíná zajímat o programování a do budoucna se mu chce věnovat.[2], [12]

3.2 Firmware

Enchanting umožnuje flash firmware, tedy nahrání firmwaru do řídící jednotky. Firmware je leJOS a polední verze pro řadu NXT je leJOS 0.9.1. Enchanting zkompiluje vytvořený grafický programový blok do Java souborů a nahraje do řídící jednotky. LeJOS pak překládá Java soubory pro ovládání řídící jednotky.[13]

3.3 Podmínkové bloky v Enchantingu

Podmínka se dvěma větvemi instrukcí

Při splnění podmínky (viz obrázek 23) se vykoná instrukce, a pokud podmínka není splněna, tak se vykoná druhá instrukce.

Obrázek 23: Podmínka se dvěma větvemi instrukcí

Zdroj: vlastní zpracování

Podmínka s jednou větví instrukcí

Pokud je podmínka splněna (viz obrázek 24), tak je instrukce vykonána a pokud ne, tak se přeskočí, žádná instrukce se nevykoná.

Obrázek 24: Podmínka s jednou větví instrukcí

Zdroj: vlastní zpracování

Čekající podmínka

Čekající podmínka (viz obrázek 25) zastaví program a čeká třeba na splnění podmínky, např. přiřazení hodnoty nějaké proměnné, od nějaké části programu či podprogramu.

Obrázek 25: Čekající podmínka

Zdroj: vlastní zpracování

3.4 Cyklické bloky v Enchanting

Základní cyklické bloky, nutné pro všechny programy.

Nekonečný cyklus

Prakticky se jedná o cyklus bez konce (viz obrázek 26), který není ukončen podmínkou. V cyklu je umístěna hlavní nebo dílčí část programu, která bude vykonávána stále dokola. Například příkaz k jízdě, který sám o sobě nic neudělá, dokud není umístěn v cyklu. I když to může být trochu zavádějící označení „nekonečný“ cyklus, jelikož cyklus musí mít správně konec, není tomu úplně tak. Cyklus bude ukončen ve chvíli, kdy bude ukončen celý program. Na „nekonečný“ cyklus se nedá napojit již žádný jiný blok.

Obrázek 26: Nekonečný cyklus

Zdroj: vlastní zpracování

Cyklus s pevným počtem opakování

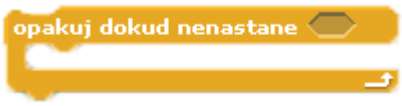
Zde se jedná o klasický cyklus s pevným počtem opakování (viz obrázek 27), kde se uvede počet požadovaných cyklů. Na následující cyklus se dají napojovat další bloky programu.

Obrázek 27: Cyklus s pevným počtem opakování

Zdroj: vlastní zpracování

Cyklus „opakuj, dokud nenastane“

Jde o standartní cyklus s podmínkou (viz obrázek 28), který probíhá stále dokola, dokud nedojde ke splnění podmínky. Po splnění podmínky je cyklus ukončen a pokračuje dalšími napojenými bloky. Cyklus nemusí proběhnout ani jednou a chová se tedy jako by měl podmínku na počátku.



Obrázek 28: Cyklus „opakuj, dokud nenastane“ s podmínkou na počátku

Zdroj: vlastní zpracování

Pro potřebu cyklu s podmínkou na konci (viz obrázek 29), stačí před cyklus umístit jednu totožnou instrukci, kterou obsahuje i samotný cyklus.

Obrázek 29: Cyklus „opakuj, dokud nenastane“ s podmínkou na konci

Zdroj: vlastní zpracování

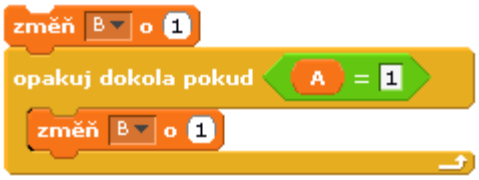
Cyklus „opakuj dokola pokud“

Cyklus stále probíhá dokola, dokud je splněna podmínka (viz obrázek 30). Jinak je ukončen. Cyklus nemusí proběhnout ani jednou a defaultně se chová jak s podmínkou na počátku.

Obrázek 30: Cyklus „opakuj dokola pokud“ s podmínkou na počátku

Zdroj: vlastní zpracování

Pro cyklus s podmínkou na konci je opět potřeba umístit jednu totožnou instrukci z cyklu před cyklus (viz obrázek 31).



Obrázek 31: Cyklus „opakuj dokola pokud“ s podmínkou na konci

Zdroj: vlastní zpracování

4 POPIS ÚLOHY A KONSTRUKCE ROBOTA

4.1 Popis úlohy

Myšlenka úlohy „pohyb robota v prostoru“ je dostat se z místa A do předem určeného místa B, jehož vzdálenost a směr bude znám, přitom se vyhnout neznáme překážce a dopočítat nový směr pomocí goniometrické funkce tangens a přepočítat vzdálenost pomocí Pythagorovi věty.

4.2 Konstrukce robota

Stavba robota byla zcela spontánní a nebyla zvolena žádná předloha, ani z návodu stavebnice či jiného zdroje a zabrala několik hodin. Robot (viz obrázek 32 a obrázek 33) je vybaven čtyřmi senzory a třemi motory přičemž dva se starají o samotný pohyb v prostoru a jeden se věnuje natáčení ultrazvukovému senzoru. Ultrazvukový senzor, jenž zjišťuje vzdálenost je cyklicky motorem natáčen v úhlech -90°,-60°,-30°,-0°,30°,60°,90° pro lepší orientaci v prostoru a detekci překážek. Dalším senzorem je kompasový senzor a nakonec dva dotykové.

Obrázek 32: Model robota

Zdroj: vlastní zpracování

Obrázek 33: Model robota

Zdroj: vlastní zpracování

4.3 Problematika otáčení 4 kolových vozidel

Při konstrukci bylo zvoleno čtyřkolové provedení robota, s čím vznikl problém s přenesením pohonu na další dvě kola, jelikož jsou k dispozici pouze dva motory a robot by se s pohonem dvou kol jen stěží otáčel, pokud vůbec. Klasický přístup je, že se jeden motor stará o pohon pravého kola a druhý o pohon levého. U této koncepce bylo setrváno s tou modifikací, že od pravého předního kola, které je přímo napojeno na motor, je pomocí hřídele poháněno i zadní pravé kolo a totéž je i na levé straně. Tím bylo docíleno schopnosti robota otočit se namístě o libovolný počet stupňů.

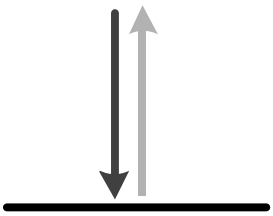
Po vyřešení problému s pohonem vyvstal problém s otáčením robota o potřebný úhel otočení. Jelikož při různých rychlostech otáčení se otáčel vždy o jiný úhel. S tím souvisí i problém jízdního povrchu, po kterém se robot pohybuje. Jelikož na kluzkém povrchu se otočí rychleji a snáze než na drsném nebo matném. Tento problém byl řešen použitím kompasu.

4.4 Problematika použití kompasového senzoru

S použitím kompasového senzoru, který měl sloužit k určení úhlu otočení robota, se objevil problém s jeho spolehlivostí a relevancí generovaných údajů. To je přisuzováno tomu, že samotné testování probíhalo v uzavřené místnosti budovy školy, kde je spousta elektrických vodičů generují magnetické pole a železobetonová konstrukce budovy. Tyto předpokládané faktory ovlivňovaly a vychylovaly hodnoty generující kompasem v různých částech místnosti. Další komplikací byly samotné otřesy při otáčení robota, které též snižují přesnost. Proto je praktické využití v úloze až na výjimku vyloučeno. Jediným řešením, i když také ne moc přesným byl nápad před samotným vykonáním jízdy robota učinit otočku na místě o 360° a „osahat“ si jízdní povrch. Otočka je zkontrolována kompasovým senzorem, kde jeho použít lze akceptovat, jelikož na jednom místě jsou stále stejné rušivé faktory a nedojde k takovému zkreslení generovaných údajů. Během otáčení je počítán čas, který robotovi trvá k otočení oněch 360°. Od naměřeného času se poté odvíjí výpočty k otočení o potřebný počet stupňů. Tím to způsobem byla zajištěna flexibilita různých jízdních povrchů.

4.5 Problematika určení překážek

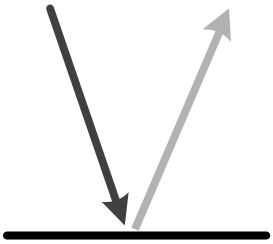
Určení vzdálenosti překážky není pro ultrazvukový senzor žádný problém, pokud je kolno (viz obrázek 34).



Obrázek 34: Odraz ultrazvuku od kolmé překážky

Zdroj: vlastní zpracování

Ten nastává ve chvíli kdy, je překážka o nějaký úhel natočená (viz obrázek 35) a ultrazvuk se odrazí a putuje úplně jinam, než je požadováno, jelikož úhel odrazu se rovná úhlu dopadu.



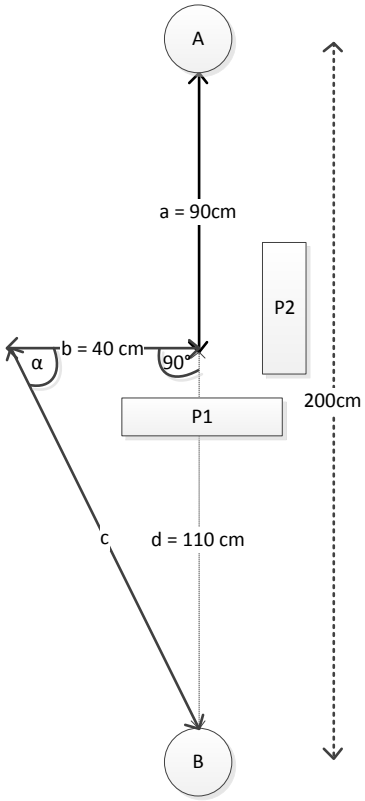
Obrázek 35: Odraz ultrazvuku od překážky pod úhlem

Zdroj: vlastní zpracování

Tomu se dá zabránit natočením ultrazvukového senzoru. Jenže stálá natáčení robota s pevně přidělaným senzorem by bylo značně neefektivní. Tomu bylo zabráněno umístěním ultrazvukového senzoru na motor a při měření vzdálenosti je cyklicky natáčen v úhlech -90°,-60°,-30°,-0°,30°,60°,90°. Tím je docíleno lepší možnosti sledování okolních překážek pro lepší orientaci v prostoru. Dále natáčením ultrazvukového senzoru je i vyřešen problém pro určený směru jak se vyhnout překážce, zda vpravo nebo vlevo.

4.6 Komplexní popis úlohy a princip programu

Po zpuštění programu je uživatel dotázán na řídící jednotce pro zadání vzdálenosti, kterou má robot urazit z bodu A do bodu B (viz obrázek 36).



Obrázek 36: Schéma úlohy

Zdroj: vlastní zpracování

Nejdříve uživatel zadá počet metrů a poté počet centimetrů. Po zadání vzdálenosti např. 2 m robot udělá testovací otočku o 360°, aby zjistil dobu pro otočení na daném povrchu. Kontrolu otočení zajišťuje kompas. Naměřený čas je dělen 360 a tím je získána doba pro otočení o 1° až bude potřeba robota počet o potřebný počet stupňů. Po celou dobu jízdy robota je rychlost programem nastavena na pevnou hodnotu 6 cm/s. Po té robot vyjede od A směrem k bodu B (úsečka a) a cestou stále zjišťuje ultrazvukový senzor okolní překážky v úhlu 180°, tomu je ještě pro jistotu vybaven dvěma dotykovými senzory v přední části. Pokud ultrazvukový senzor detekuje překážu v menší vzdálenosti nežli je 35 cm (překážka P1), pohyb je okamžitě ukončen. Do proměnné je uložena hodnota doby jízdy a ultrazvukový senzor udělá jeden cyklus, když robot stojí na místě pro zjištění aktuálních překážek. Z naměřených hodnot určí, na kterou stranu se překážce vyhne. Určení probíhá na jednoduché bázi, a to tak že sečtou naměřené vzdálenosti v úhlech -90°,-60°,-30°,-0° a v úhlech 0°,30°,60°,90° a proběhne porovnání, kde je víc prostoru a na tu stranu se robot zatočí o 90°. Jelikož na levé straně má překážku P2, tak otočení proběhne na pravou stranu. Po o točení proběhne kontrola překážky P1 ultrazvukovým senzorem, pokud překážka stále brání v pohybu správným směrem, robot se rozjede (úsečka b) a jede, dokud neskončí překážka P1, což stále zjišťuje ultrazvukovým senzorem. Jakmile robot zjistí, že mu již překážka P1

nebrání, ujede ještě 30 cm, aby došlo k větší pravděpodobnosti vyhnutí. Na to si robot vypočítá úhel α, o který se má otočit k zadanému cíli. Ten je vypočítá pomocí goniometrické funkce tangens (viz **Chyba! Nenalezen zdroj odkazů.**), ale vzorec aby mohl být použit, musí ýt modifikován tím, že bude vypočítán úhel β a odečten od 180° (viz **Chyba! Nenalezen zdroj odkazů.**).

$$\tan\ \beta\ = \frac{délka\ protilehlé\ odvěsny}{délka\ přilehlé\ odvěsny} \tag{1}$$

$$Úhel\ otočení\ \ \alpha\ = 180 - \tan\ \beta\ \frac{celková\ (zadaná)\ vzdánost - úsečka\ a}{úsečka\ b} \tag{2}$$

Pokud jsou do vzorce dosazeny hodnoty ze schématu (viz **Chyba! Nenalezen zdroj odkazů.**) ískáme přibližně výslednou hodnotu 114°.

$$Úhel\ otočení\ \ \alpha\ = 180 - \tan\ \beta\ \frac{200 - 90}{40} \tag{3}$$

Jelikož si robot na začátku celého programu zjistil, jaký čas potřebuje pro otočení o 1° nyní danou hodnotu pouze vynásobí 114. Dále je ještě potřeba vypočítat hodnotu úsečky c. Ta je zjištěna Pythagorovou větou (viz **Chyba! Nenalezen zdroj odkazů.**).

$$c^2 = a^2 + b^2 \tag{4}$$

Pro výpočet úsečky v úloze je vzorec trochu upraven (viz **Chyba! Nenalezen zdroj odkazů.**).

$$Úsečka\ c = \ \overline{(celková\ \ zadaná\ \ vzdánost\ - úsečka\ a\ )^2 + (úsečka\ b)^2} \tag{5}$$

Po dosazení hodnot do vzorce ze schématu (viz **Chyba! Nenalezen zdroj odkazů.**) získáme řibližně výslednou hodnotu 117 cm, kterou musí robot urazit k bodu B, kde bude program automaticky ukončen.

$$Úsečka\ c = \ \overline{(200 - 90)^2 + 40^2} \tag{6}$$

5 NÁVRH A IMPLEMENTACE KÓDU

Program ovládající robota se skládá ze sedmi scénářů. Pro lepší vysvětlení a následné porozumění principu je každému scénáři přidělen identifikátor A-G. Některé scénáře jako „A“ a „B“ jsou popsány po částech na několik stránek, jelikož jsou složitější.

V kódu se nepracuje přímo s proměnnými, ale s jednorozměrnými poli a jejich souřadnicemi. V Enchantingu jsou jednorozměrná pole nazvány seznamy. Důvod používání seznamů je, že při použití proměnných program Enchanting správně nezkompiloval kód.

POUŽITÉ SEZNAMY

- A;
- A_uhly;
- B;
- B_prekazka;
- Uhel;
- Vzdalenost.

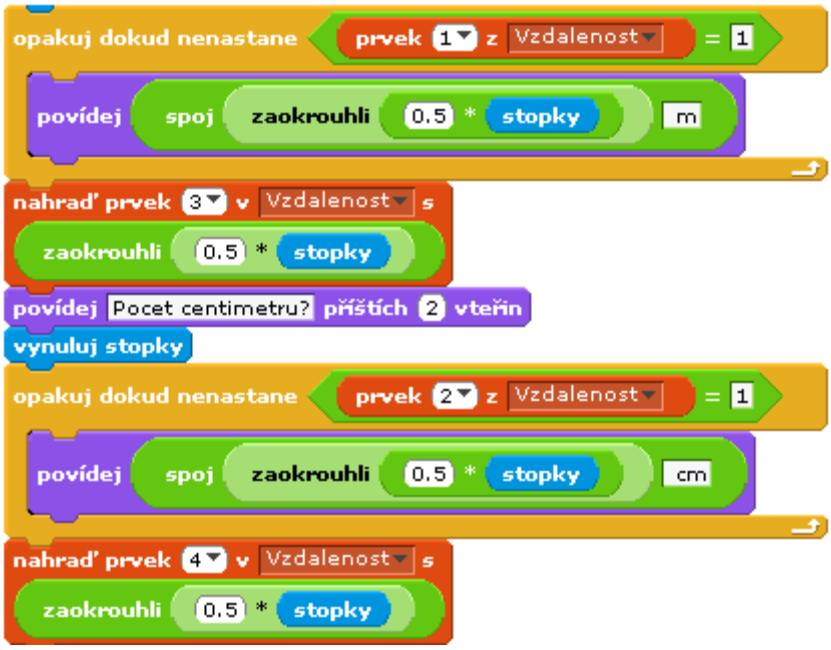
5.1 Scénář A

Program po spuštění (viz obrázek 37) začíná vypsáním upozorňovací hlášky *„Zadat vdalenot“*, která trvá 2 vteřiny a hned po té je vypsána hláška *„Počet metru?“*, která taktéž trvá 2 vteřiny. Jakmile skončí, proběhne okamžitě 6x cyklus s pevným počtem opakování, který vytvoří seznam *„Vzdalenost“* s 6 souřadnicemi, z nichž každá bude mít nulovou hodnotu. Na to dojde k vynulování vnitřního času běhu programu v řídící jednotce.

Obrázek 37: Scénář A část 1.

Zdroj: vlastní zpracování

Následuje cyklus s výstupní podmínkou (viz obrázek 38) *„opakuj, dokud nenastane, že 1. prvek v seznamu Vzdálenost je roven 1“* bude se na display řídící jednotky zobrazovat počet metrů v závislosti na čase (stopky*0,5), takže každé 2 vteřiny bude hodnota navýšena o jeden metr, dokud nedojde ke stisknutí tlačítka „enter“ na řídící jednotce. Jeho stisknutím, které je definováno ve scénáři „C“ (viz obrázek 51) dojde ke změně 1. prvku v seznamu vzdálenost na hodnotu 1, a tím k ukončení cyklu. Následuje přiřazení dané hodnoty k 3. prvku v seznamu „Vzdalenost“. Po té je na řídící jednotce vypsána hláška *„Počet centimetru“*, vynulování stopek a opět následuje cyklus s výstupní podmínkou *„opakuj, dokud nenastane, že 2. prvek v seznamu Vzdálenost je roven 1“*. Tentokrát pro vypisování centimetrů jako předtím metrů. Po stisknutí tlačítka „enter“, které je stále definováno ve scénáři „C“ (viz obrázek 51) dojde ke změně 2. prvku v seznamu vzdálenost na hodnotu 1 a tím ukončení cyklu. Následuje zase přiřazení zjištěné hodnoty tentokrát 4. prvku ze seznamu „Vzdalenost“.

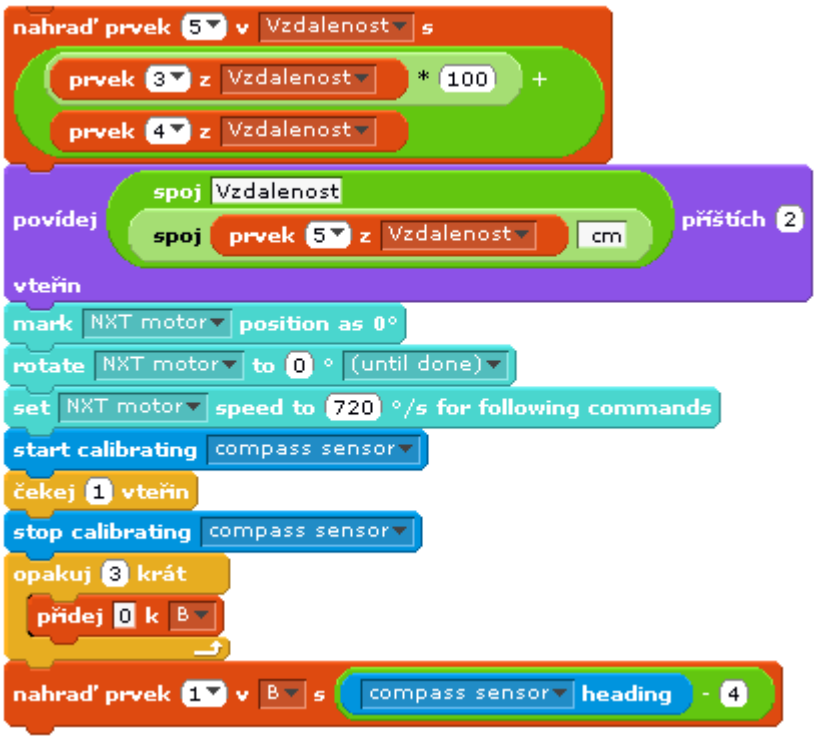


Obrázek 38: Scénář A část 2.

Zdroj: vlastní zpracování

Na řadě je nahrazení (viz obrázek 39) 5. prvku ze seznamu „Vzdalenost“ výslednou jednotnou hodnotou v centimetrech, kterou je hodnota zjištěných metrů vynásobená stem a k tomu přičtení hodnoty zjištěných centimetrů. Závěrem této části je vypsání celkové zadané vzdálenosti. Jako další je označení hodnoty 0° a rychlosti na motoru pro pohyb ultrazvukového senzoru. Po té je spuštěna kalibrace kompasového senzoru, počkání 1 vteřiny a ukončení kalibrace kompasového senzoru. Po kalibraci následuje vytvoření cyklem

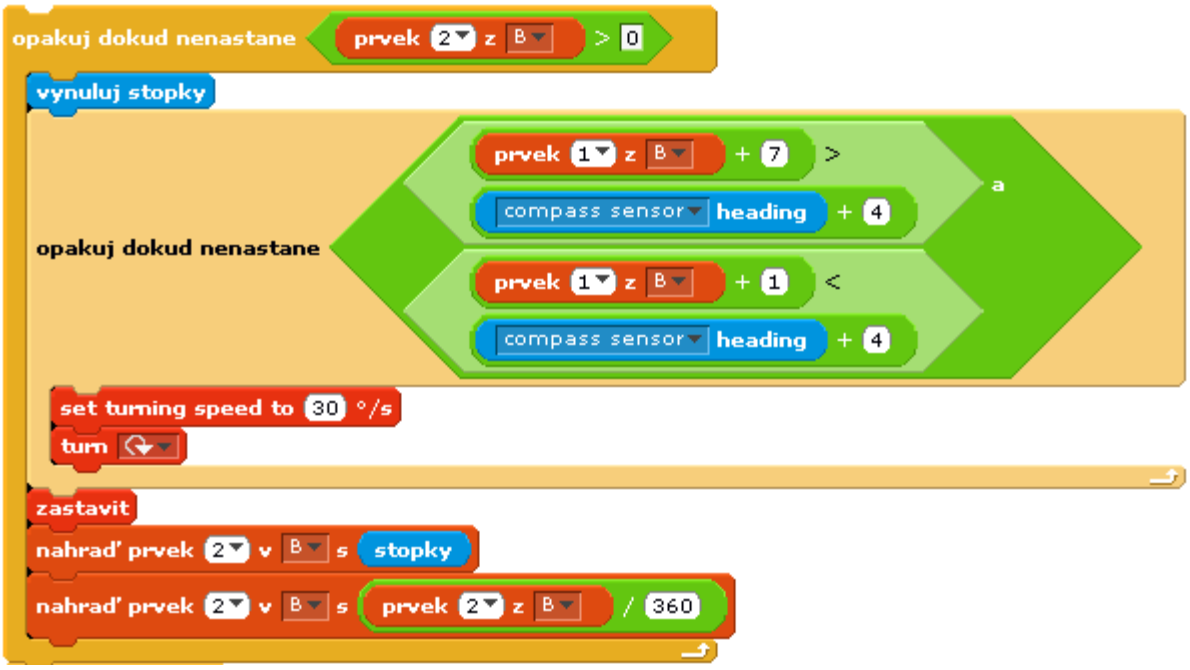
s pevným opakováním třech souřadnic k seznamu „B“. Pak přijde na řadu přiřazení k 1. prvku seznamu „B“ aktuální hodnotou kompasu a odečtení 4. Odečtení hodnoty 4 je z důvodu otáčení robota, aby v následujícím cyklu byla nesplněna podmínka cyklu.



Obrázek 39: Scénář A část 3.

Zdroj: vlastní zpracování

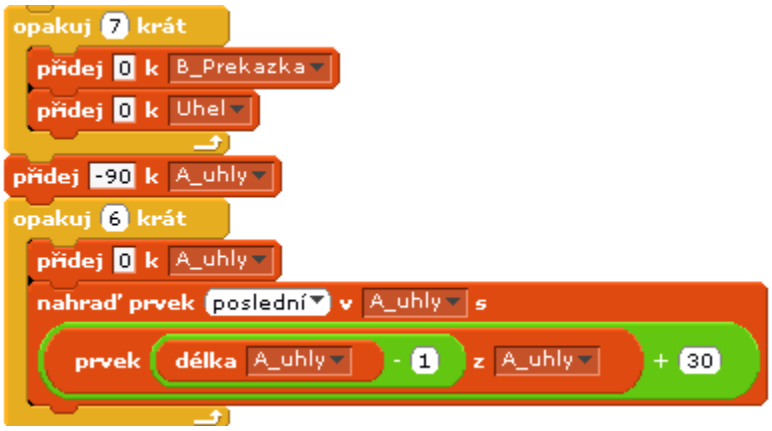
Dále je cyklus s podmínkou (viz obrázek 40) *„opakuj, dokud nenastane, že 2. prvek ze seznamu B bude větší než nula“*. Uvnitř cyklu proběhne vynulování stopek, aby bylo možné změřit dobu otáčení o 360° Uvnitř je umístěn další cyklus s podmínkou *„opakuj, dokud nenastane, že 1. prvek sezamu B plus 7 bude větší než aktuální hodnota kompasu plus 4 a zároveň 1. prvek seznamu B plus 1 bude menší nežli aktuální hodnota kompasu plus 4“*. Tato rozsáhlá podmínka je z důvodu nepřesnosti kompasu, jelikož při otáčení robota dochází k vibracím, které také na přesnosti nepřidávají. Jako instrukce cyklu je nastaveno otáčení robota na pravou stranu a rychlost otáčení. Jakmile je podmínka splněna dojde k vystoupení z cyklu a následnému zastavení robota. Na to dojde k uložení naměřené časové hodnoty do 2. prvku seznamu „B“ a hned na to ten samí prvek bude vydělen hodnotou 360. Tím byl získán čas na otočení o 1°.



Obrázek 40: Scénář A část 4.

Zdroj: vlastní zpracování

Následuje cyklus s pevným počtem opakování (viz obrázek 41), který vytvoří seznamy „B_prekazka“ a „Uhel“, každý se 7 souřadnicemi. Pak přijde na řadu postupné vytvoření seznamu „A_uhly“. Jako první prvek bude -90 a dalších 6 souřadnic vytvoří cyklus s pevným počtem opakování. Celkem cyklus tedy proběhne 6x a vždy přidá souřadnici s hodnotou 0 a hned zároveň inkrementuje poslední hodnotu seznamu „A_uhly“ o 30. Tím vzniknou hodnoty seznamu -90°,-60°,-30°,-0°,30°,60°,90°.



Obrázek 41: Scénář A část 5.

Zdroj: vlastní zpracování

Na to navazuje (viz obrázek 42) vytvoření 11 souřadnic seznamu „A“. Poněvadž některé souřadnice mají hodnotu 0 a některé 1 pro pohodlnější další prací, tak nejsou umístěny v cyklu, jako tomu bylo doposud.

Obrázek 42: Scénář A část 6.

Zdroj: vlastní zpracování

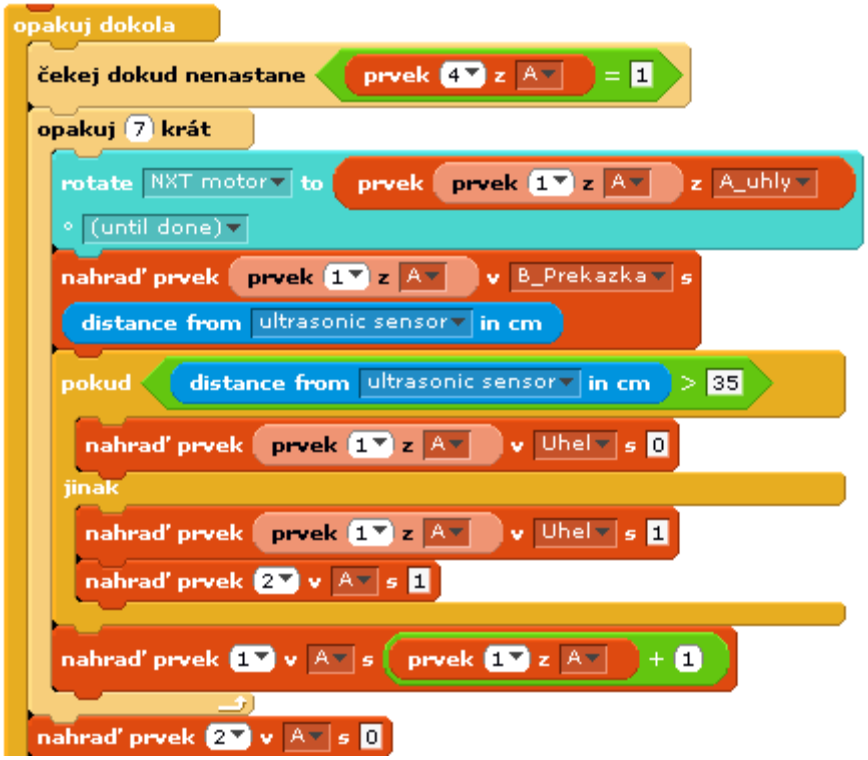
Jako další (viz obrázek 43) je ještě jednou vytvoření celkové vzdálenosti, jako tomu bylo u 5. prvku seznamu „Vzdalenost“ (viz obrázek 39), ale tentokrát jsou hodnoty přiřazeny 7. prvku seznamu „A“. Následně je získaná hodnota vydělena hodnotou 6 protože rychlost pohybu robota je 6 cm/s definovaná ve scénáři „B“ (viz obrázek 46), takže došlo k přepočtu vzdálenosti na čas. Vypočítaná hodnota byla zpětně přidělaná 7. prvku sezamu „A“.

Obrázek 43: Scénář A část 7.

Zdroj: vlastní zpracování

Nyní je na řadě samotný pohyb ultrazvukového senzoru a získávání z něho informací (viz obrázek 44). Vše je umístěno v jednom „nekonečném“ cyklu. Uvnitř se nachází čekající podmínka *„čekej, dokud nenastane, že 4. prvek seznamu A je roven jedné“*. Podmínka získává podnět k čekání a tedy hodnotu 0 ze scénáře „B“ (viz obrázek 47), ale pouze tehdy když ultrazvukový zaznamená překážku ve své definované blízkosti. Čekající podmínku následuje cyklus s pevným počtem opakování, který bude pohybovat s motorem na kterém je přidělaný ultrazvukový senzor. V celém cyklu se pracuje s 1. hodnotou seznamu „A“, která lze též zapsat A[1] a má hodnotu 1. Na počátku cyklu je, aby se motor nastavil do polohy, jejíž hodnota je uložena v prvku A[1] ze seznamu „A_uhly“. Hodnota již byla vytvořena a je -90. Následuje zapsání naměřené hodnoty ultrazvukovým senzorem do prvku A[1] seznamu

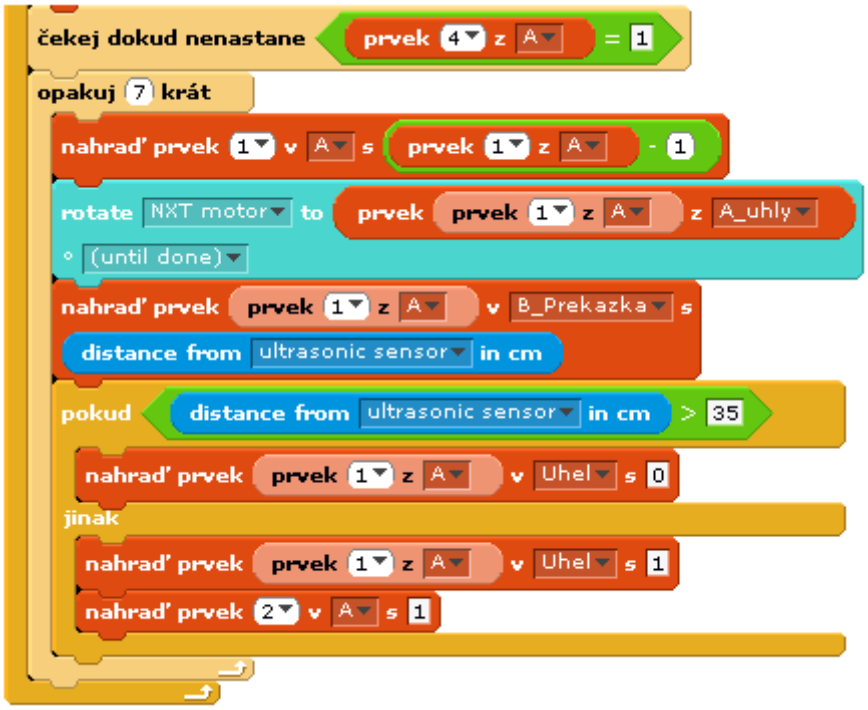
„B_prekazka“. Další částí v cyklu je podmínka, která zní, *„pokud naměřená hodnota ultrazvukovým senzorem je větší než 35 cm, tak prvek A[1] seznamu Uhel nahraď hodnotou 0“*. Pokud zmíněná podmínka není splněna, *„tak A[1] seznamu Uhel nahraď hodnotou 1“*. K této instrukci ještě dojde k přiřazení 2. prvku seznamu „A“ hodnotou 1, která bude mít za důsledek zastavení robota ve scénáři „B“ (viz obrázek 47). Za podmínkou je ještě umístěna inkrementace 1. prvku seznamu „A“, tedy v dalším taktu bude A[1] = 2 atd. jinak vše bude probíhat stejně. Jakmile cyklus 7x proběhne a bude ukončen, dojde k nahrazení 2. prvku seznamu „A“ hodnotou 0.



Obrázek 44: Scénář: A část 8.

Zdroj: vlastní zpracování

Dále následuje (viz obrázek 45) opět cyklus s pevným počtem opakování, prakticky stejný jako předchozí jen s tím rozdílem, že bude mít nestarosti zpětný pohyb ultrazvukového senzoru. Tudíž v něm bude tentokrát docházet k dekrementaci, což znamená, že v prvním taktu bude A[1] = 7.



Obrázek 45: Scénář A část 9.

Zdroj: vlastní zpracování

5.2 Scénář B

Tento scénář má na starosti pohyb robota. Hned na začátku scénáře (viz obrázek 46) je definována rychlost jízdy a rychlost otáčení. Zatím je umístěna čekající podmínka *„čekej, dokud nenastane A[1] = 7“*. To zajišťuje ze scénáře „A“ (viz obrázek 44) provedení alespoň jedno cyklu pro zjištění překážek v rozmezí 180° dříve než se robot rozjede.

Obrázek 46: Scénář B část 1.

Zdroj: vlastní zpracování

Jakmile byla předchozí podmínka splněna (viz obrázek 46), dojde k vynulování času (viz obrázek 47) a spuštění velkého „nekonečného“ cyklu, který obsahuje zbytek scénáře. V cyklu je podmínka, která je přes celý cyklus a zajišťuje buď jízdu, nebo řeší vyhnutí překážce. Podmínka je splněna pokud A[2] = 0, což znamená, že ultrazvukový senzor žádnou překážku v blízkosti 35 cm nezaznamenal a robot může jet. Dále je ještě instrukce, která přiřazuje stále aktuální hodnotu času do A[5] se kterou pracuje i scénář D (viz obrázek 52).

Jestliže podmínka splněna není a tedy ultrazvukový senzor zaznamenal překážku, tudíž A[2] = 1, dojde k zaznamenání ujetého času do A[11] a k vynulování A[5]. Hned na to dojde k zastavení robota a následně přiřazení hodnoty „0“ do A[4], to má za následek ve scénáři „A“ (viz obrázek 44 a obrázek 45) okamžité zastavení pohybu čidla a stopnutí získávání zmatečných informací, až se bude robot otáčet. Pro to je dále umístěn čekající blok, aby cyklus s ultrazvukovým čidlem mohl být bezpečně dokončen.

Obrázek 47: Scénář B část 2.

Zdroj: vlastní zpracování

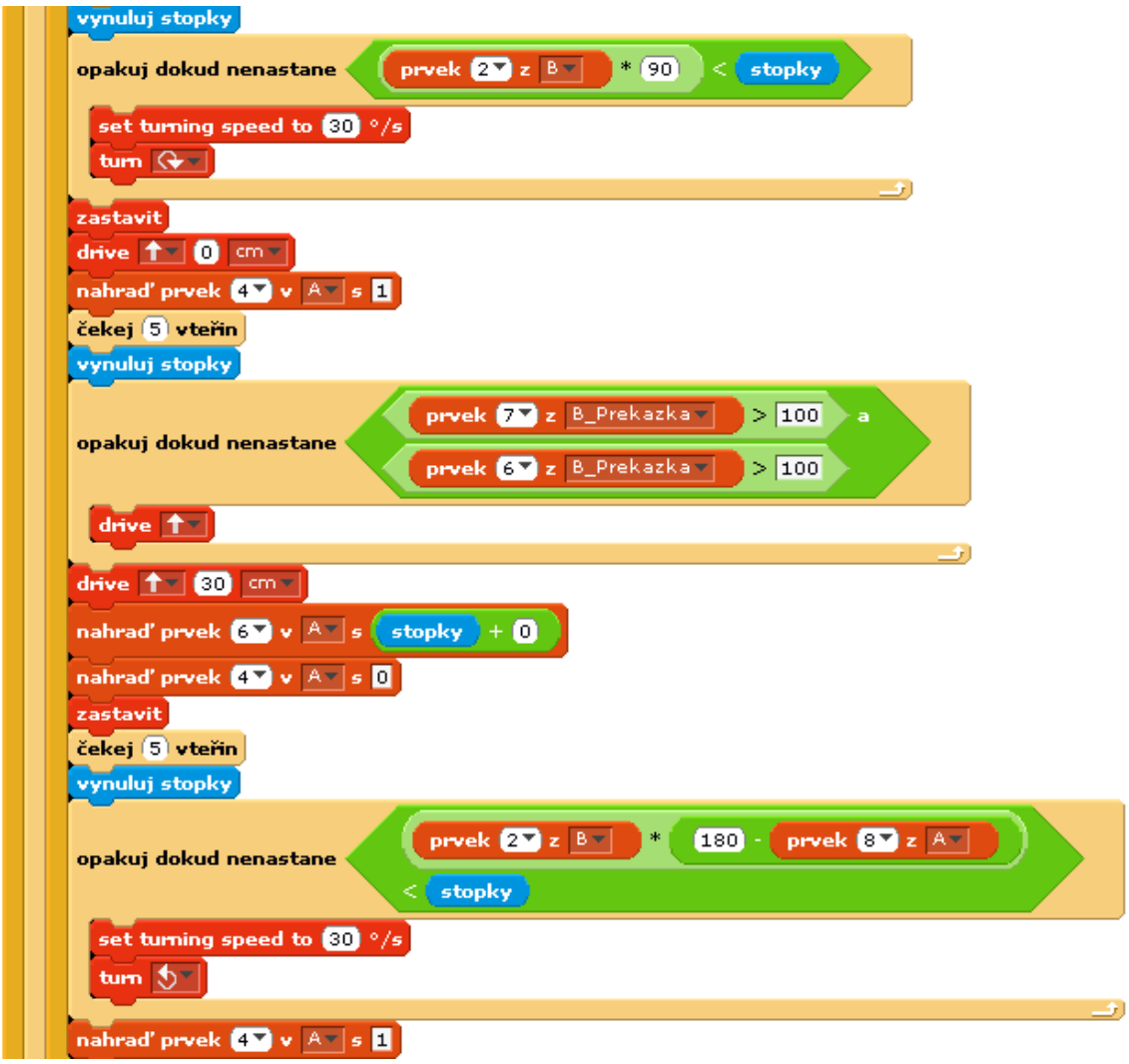
Nyní je na řadě podmínka (viz obrázek 48), která rozhodne, na kterou stranu se robot bude překážce vyhýbat. *„Pokud (B_prekazka[7] + B_prekazka[6] + B_prekazka[5] + B_prekazka[4]) < (B_prekazka[4] + B_prekazka[3] + B_prekazka[2] + B_prekazka[1])“* dojde k otočení robota doprava, ale to bude rozebráno nadále.

Obrázek 48: Scénář B část 3.

Zdroj: vlastní zpracování

Instrukce začíná (viz obrázek 49) vynulováním času, hned poté je cyklus s podmínkou *„opakuj, dokud nenastane (B[2]*90) < hodnota stopek (času)“*. B[2] je hodnota času pro

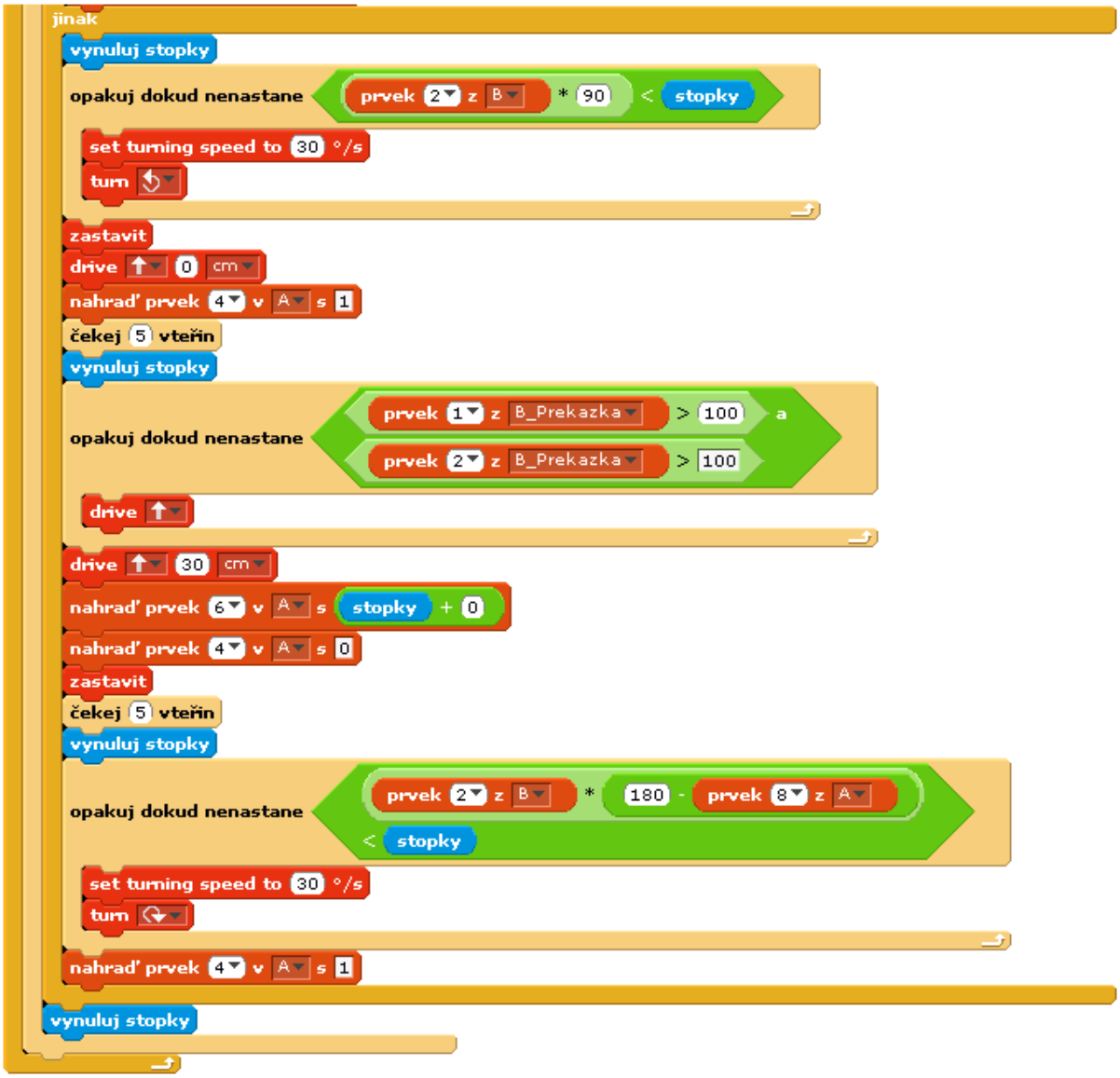
otočení robota o 1° a jelikož chceme otočení o 90° je B[2] vynásobeno 90. Instrukcí tohoto cyklu je otáčení doprava. Takže ve výsledku cyklus zajistí otáčení robotem, dokud čas otáčení nebude větší než B[2]*90. Hned na to se cyklus ukončí a dojde k zastavení pohybu otáčení. Za tím je přiřazení hodnoty 1 pro A[4]. To má za následek opět spuštění ultrazvukového senzoru ze scénáře A (viz obrázek 44 nebo obrázek 45). Následuje 5 vteřinové setrvání, než bude kód pokračovat a to z důvodu, aby ultrazvukový senzor udělal jeden cyklus pro zjištění okolních překážek. Dále je zase vynulování času, za kterým následuje podmínka *„opakuj, dokud (B_Prekazka[7] > 100) a (B_Prekazka[6] > 100)“* a instrukce pro jízdu robota. To znamená, že pokud vzdálenost od překážky je na úhlu -90° a -60° menší než 100 cm tak platí podmínka, cyklus se opakuje a robot jede vedle překážky. Jakmile bude vzdálenost větší, cyklus se ukončí a robot ujede ještě 30 cm, aby byla jistota vyhnutí překážce. Hned dojde k zápisu času trvání jízdy do seznamu A[6], přiřazení hodnoty 0 pro A[4], k zastavení ultrazvukového senzoru (viz obrázek 44 a obrázek 45) a zastavení pohybu robota. Počká se 5 vteřin na ukončení cyklu ultrazvukového senzoru a opět vynulování času. Dále je podmínka *„opakuj, dokud nenastane B[2]*180- A[8]“*. Jak již bylo řečeno B[2] osahuje čas na otočení o 1° a vynásobením 180 získáme čas na otočku o 180°. Od výsledného času bude odečtena hodnota tangentu A[8], který byl vypočítán ve scénáři „D“ (viz obrázek 52). Tím bude získán výsledný čas otáčení k cílovému bodu B. V instrukci cyklu je tedy definováno otáčení robota na levou stranu. Za podmínkou následuje přiřazení hodnoty 1 seznamu A[4] pro zpuštění cyklu s ultrazvukovým senzorem (viz obrázek 44 a obrázek 45). Tím byla vyřešena kladná větev podmínky *„pokud (B_prekazka[7] + B_prekazka[6] + B_prekazka[5] + B_prekazka[4]) < (B_prekazka[4] + B_prekazka[3] + B_prekazka[2] + B_prekazka[1])“* (viz obrázek 48).



Obrázek 49: Scénář B část 4.

Zdroj: vlastní zpracování

Pokud podmínka (viz obrázek 48) nebude splněna, tak následuje záporná instrukce (viz obrázek 50), která je prakticky stejná (viz obrázek 49), akorát že se robot otáčí inverzně a při vyhýbání překážce sleduje uhly úhlu 60° a 90°. Po ukončení cyklu dojde k vynulování času.



Obrázek 50: Scénář B část 5.

Zdroj: vlastní zpracování

5.3 Scénář C

Scénář „C“ (viz obrázek 51) pouze zajišťuje správný chod scénáře „A“. Na začátku je čekající blok než ve scénáři „A“ proběhne vypsání hlášek *„Zadat vzdalenost“* a *„Pocet metru?“* (viz obrázek 37). Po uplynutí 4 vteřin je umístěn čekající blok na stisk klávesy „enter“ na řídící jednotce. Během té doby scénář „A“ generuje počet metrů (viz obrázek 38). Až si uživatel vybere vhodnou hodnotu, stiskne tlačítko „enter“ a dojde k přidání hodnoty 1 do seznamu Vzdalenot[1], na kterou čeká scénář „A", který vygenerovanou hodnotu vloží do seznamu Vzdalenost[3]. Po zapsání hodnoty je znovu čekající blok nastaven na 2 vteřiny, během níž opět scénář „A“ vypisuje hlášku *„Pocet centimetru?“* (viz obrázek 38) a hned po té čeká na stisknutí tlačítka „enter“ během níž scénář „A“ generuje centimetry. Po stisknutí

klávesy dojde k přiřazení hodnoty 1 seznamu Vzdalenot[2], na niž zase čeká scénář „A“ jako u metrů. Na konci scénáře „C“ je blok pro „zastavení scénáře“ a to znamená, že se nadále v programu již se scénářem „C“ nepočítá a může být ukončen.

Obrázek 51: Scénář C

Zdroj: vlastní zpracování

5.4 Scénář D

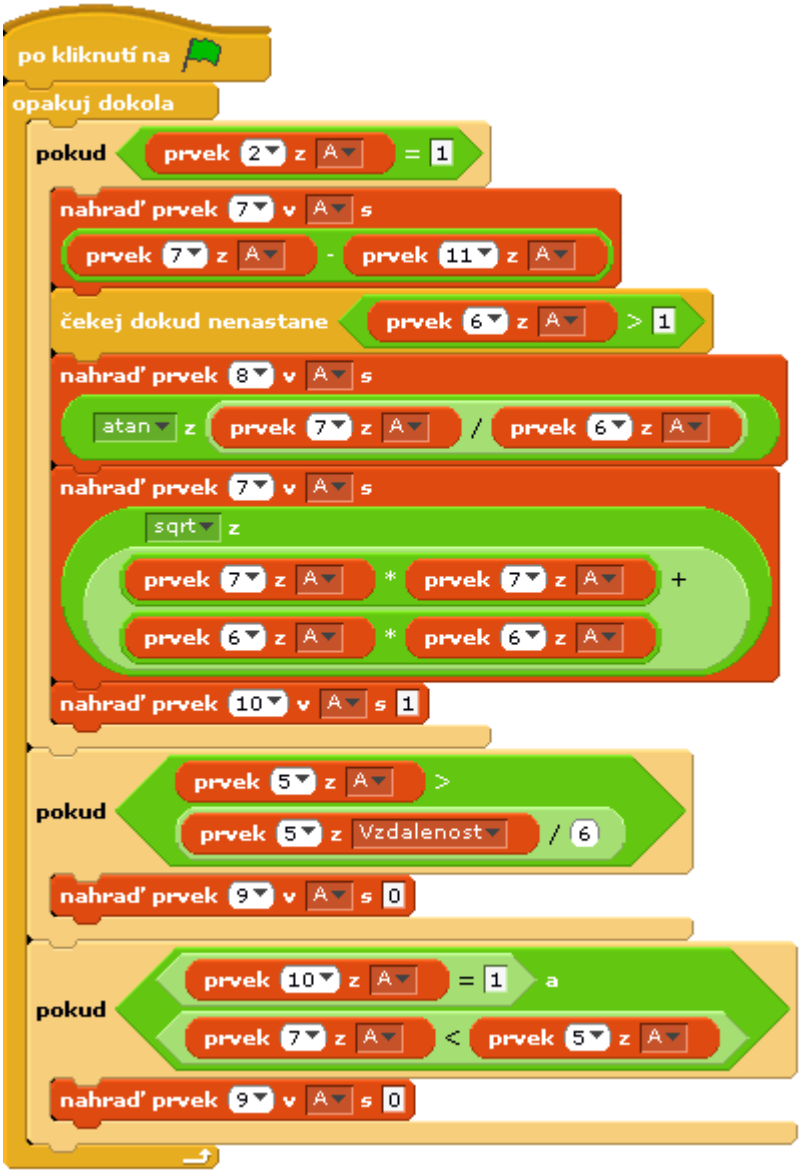
Scénář „D“ (viz obrázek 52) zajišťuje ukončení programu po úspěšném splnění úkolu. Dále ještě vypočítává hodnotu tangens a Pythagorovu větu. Přes celý scénář je „nekonečný“ cyklus, v němž jsou umístěny tři podmínky, každá s jednou větví instrukcí.

První podmínka znamená, že pokud dojde ke splnění podmínky *„A[2] je rovno 1“* bude instrukce vykonána. Tato hodnota přichází ze scénáře „A“ (viz obrázek 44 a obrázek 45), a znamená, že ultrazvukový senzor narazil na překážku. Jako instrukce je nahrazení A[7] s A[7] - A[11]. A[7] je hodnota vzdálenost k ujetí ze scénáře „A“ (viz obrázek 43), kterou zadal uživatel na začátku. A[11] obsahuje hodnotu ze scénáře „B“ (viz obrázek 47), kterou robot urazil od prvního rozjetí po zastavení u překážky (délka úsečky a). Další je čekající blok s podmínkou *„čekej, dokud nenastane A[6]>1“*. A[6] je získáno ze seznamu „B“ (viz obrázek 49 a obrázek 50) a jedná se o vzdálenost „úsečky b“. Jakmile je získána hodnota větší než 1 pro A[6], následuje nastavení přiřazení hodnoty pro A[8]. A[8] je získána podílem prvku A[7] / A[6] a z nich vypočítaný tangens. Dále dojde k přepočtu hodnoty A[7], která bude reprezentovat „úsečku c“. Výpočet je (A[7] * A[7]) + (A[6] * A[6]) a následné odmocnění. Poté dojde k přiřazení hodnoty 1 k A[10].

Další podmínka zní *„pokud (A[5]) > (Vzdalenost[5] / 6)“*. Vzdalenost[5] je celková zadaná vzdálenost ze scénáře „A“ (viz obrázek 39). Tato hodnota je ještě podělena 6, aby byl získán převod na čas. A[5] (viz obrázek 47) je časová vzdálenost, kterou robot ujel od svého rozjetí (úsečka a). Instrukce má za následek přiřazení hodnoty 1 k A[9], což ukončí celý

program. Tato podmínka bude vykonána pouze, pokud robot nenarazí na žádnou překážku a pojede z bodu A do bodu B přímo.

Poslední podmínka zní *„pokud A[10] = 1 a zároveň A[7] < A[5]“*. A[10] = 1 je získáno ze závěru první podmínky tohoto scénáře „D“. Takže pokud nenastane první podmínka, která přepočítává hodnotu A[7], nenastane ani tato. Dále tato podmínka pracuje s A[5] ze scénáře „B“ (viz obrázek 47), což je časová vzdálenost, kterou robot ujel od svého rozjetí (úsečka a). Jakmile dojde k uskutečnění podmínky, nastane plnění instrukce a to přiřazení hodnoty 0 k A[9]. Tím dojde ve scénáři „G“ (viz obrázek 55) k ukončení celého programu.



Obrázek 52: Scénář D

Zdroj: vlastní zpracování

5.5 Scénář E

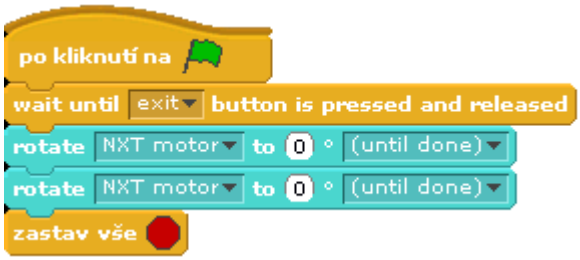
Scénář „E“ (viz obrázek 53) je určen k ovládání dotykových senzorů a v programu je pro jistotu, že by ultrazvukový senzor nějakou překážku nezaznamenal. Ve scénáři je jeden „nekonečný“ cyklus a uvnitř něho jsou dvě podmínky a každá s jednou instrukcí. První podmínka čeká na stisknutí levého dotykového senzoru. Pokud dojde k sepnutí dotykového senzoru, je vykonána instrukce což znamená, že je hodnota 0 přiřazena k A[4] a dojde ve scénáři „A“ (viz obrázek 44 a obrázek 45) k ukončení pohybu ultrazvukového senzoru. Další částí instrukce je přiřazení hodnoty 1 k A[2]. To je spouštěcí událost pro scénář „B“ (viz obrázek 47), který zastaví robota a začne řešit otáčení a vyhnutí překážce. Druhá podmínka ve scénáři „E“ čeká na sepnutí dotykového senzoru, jinak instrukce a průběh je totožný.

Obrázek 53: Scénář E

Zdroj: vlastní zpracování

5.6 Scénář F

Scénář „F“ (viz obrázek 54) je v programu umístěn pro případ nečekané potřeby ukončení programu uživatelem. Po stisknutí tlačítka „exit“ na řídící jednotce dojde k navrácení motoru s ultrazvukovým senzorem do polohy 0° a následně se úplně celý program zastaví.

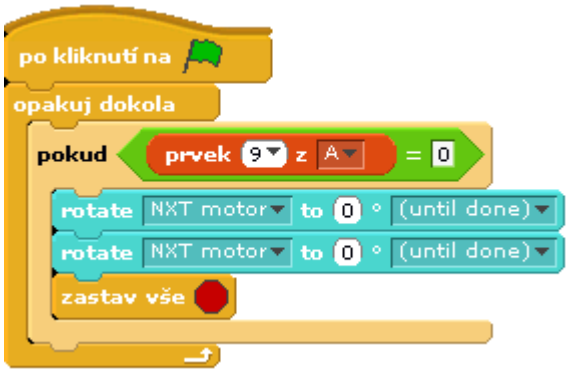


Obrázek 54: Scénář F

Zdroj: vlastní zpracování

5.7 Scénář G

Scénář „G“ (viz obrázek 55) pouze zajišťuje ukončení celého programu. Uvnitř je „nekonečný“ cyklus a vně je podmínka *„pokud A[9] = 0“*. Hodnota prvku A[9] je získána ze scénáře „D“ (viz obrázek 52). Pokud je tedy podmínka splněna dojde k nastavení servomotoru pro ultrazvukový senzor do základní polohy 0° a k ukončení celého programu.



Obrázek 55: Scénář G

Zdroj: vlastní zpracování

ZÁVĚR

V závěru práce se pokusím zrekapitulovat a shrnout získané poznatky. Cílem práce bylo navrhnutí a realizace algoritmů pro pohyb robota v prostoru a jejich následná implementace do programovatelné jednotky stavebnice Lego Mindstorms, která bude připojena k vlastnímu návrhu robota.

V první části jsem se věnoval vysvětlení respektive oživení problematiky algoritmizace, jejíž pochopení a znalost značně usnadňuje následný návrh a pochopení programu. Popsal jsem nejpoužívanější značky vývojových diagramů a jejich princip jako jsou rozhodovací bloky pro podmínky. Následně jsem se zaobíral vysvětlením jednotlivých druhů cyklů včetně krátkého popisu jejich principu.

Ve druhé části jsem se věnoval popisu stavebnice Lego Mindstorms, o původu stavebnice, vývoj, technické specifikace jednotlivých modulů jako je řídící jednotka, servomotory či čidla. Je zde uvedeno pro lepší představu i blokové schéma řídící jednotky, že se nejedná jen tak o nějakou černou skříňku, ale opravdu o počítač i když s parametry dnešní doby značně úsměvnými.

Třetí častí jsem popisoval program Enchantig. I přes to že je původně určen pro děti, dle mého názoru lze při troše snahy a obejití některých těžkostí v něm vypracovat docela zajímavé úlohy s poměrně rychlou stavbou kódu. Obrovskou výhodou je způsob sestavování kódu pomocí jednotlivých bloků, a tak odpadají problémy se syntaxí. V dané části jsem popsal podmínkové bloky, které lze Enchantigu použít. Dále jsem popsal cyklické bloky a opět jich způsoby použití a zápisu.

Ve čtvrté kapitole jsem se věnoval samotné konstrukci robota, způsobu pohonu, problémům které při stavbě a testování vznikly a způsoby, jak je co nejelegantněji vyřešit s minimální ztrátou přesnosti. Což zabralo poměrné množství času. Také je zde vysvětlen na schématu princip úlohy a její řešení.

V poslední části se věnuji samotnému vysvětlení kódu, který jsem vytvořil pro danou úlohu. Kód se skládá ze sedmy souběžných scénářů, které spolu navzájem spolupracují a předávají si navzájem hodnoty. Samotné navrhování kódu a jeho praktické testování zabralo nejvíce času. Každá přidaná část kódu musela být souběžně testována, jelikož ke konci kód vyrostl na tolik, že následné hledání chyby by bylo značně obtížné.

Během práce jsem se učinil několik zjištění, mezi něž patří, že stavebnice Lego Mindstorms je opravdu jen „hračka“ jelikož jsem se setkával se značnou nepřesností

některých čidel. Ale stavění a tvorba kódu byla zábava. Dalším zjištěním je, že navrhnutý kód pro úlohu je uplatnitelný pouze v laboratorních podmínkách a ještě s poměrně „ideálně“ rozmístěnými překážkami. Problematikou robotiky a automatických systému zabývají rozsáhlé vědecké týmy, jelikož existuje obrovské množství proměnných, které mohou průběh programu narušit či ovlivnit.

Potenciál robotiky a automatizace je obrovský, praktický omezený jen technologiemi, které se v současné době vyvíjí neuvěřitelným tempem.

POUŽITÁ LITERATURA

[1] DIVIŠ, Jozef. SPŠE MOHELNICE. Algoritmus [online]. [cit. 2014-04-28]. Dostupné z: http://www.spsemoh.cz/vyuka/algor/

[2] Enchanting: About Enchanting. [online]. [cit. 2014-04-21]. Dostupné z: http://enchanting.robotclub.ab.ca/About

[3] HiTechnic: About the new HiTechnic Barometric Sensor. [online]. [cit. 2014-04-22]. Dostupné z:http://www.hitechnic.com/blog/barometric-sensor/altimeter-gauge-using-the-new-barometric-sensor/

[4] JAKEŠ, Tomáš. Robotické vzdělání: Rozšiřující moduly systému Lego Mindstorms. [online]. [cit. 2014-04-22]. Dostupné z: https://lego.zcu.cz/web/rozsirujici-moduly-nxt

[5] JAKEŠ, Tomáš. Robotické vzdělání: Řídící jednotka NXT. [online]. [cit. 2014-04-21]. Dostupné z: https://lego.zcu.cz/web/ridici-jednotka

[6] JAKEŠ, Tomáš. Robotické vzdělání: Základní moduly systému Lego Mindstorms. [online]. [cit. 2014-04-22]. Dostupné z: https://lego.zcu.cz/web/zakladni-moduly

[7] Lego Mindstorms. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-04-28]. Dostupné z: https://en.wikipedia.org/wiki/Lego_Mindstorms

[8] Lego Mindstorms NXT. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-04-28]. Dostupné z: https://en.wikipedia.org/wiki/Lego_Mindstorms_NXT

[9] LANE, Aidan, Bernd MEYER a Jonathan MULLINS. Robotics with Enchanting and LEGO® NXT: A Project Based Introduction to Programming. [online]. [cit. 2014-04-22]. Dostupné z: http://www.loria.fr/~quinson/Mediation/Coding4Kids/Robotics-with-Enchanting-1.1.pdf

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

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

[12] Scratch. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2014-04-28]. Dostupné z: https://cs.wikipedia.org/wiki/Scratch

[13] Scratch Wiki: Enchanting (Scratch Modification). [online]. [cit. 2014-04-22]. Dostupné z:http://wiki.scratch.mit.edu/wiki/Enchanting_(Scratch_Modification)

[14] ŠTEFAN, Radim. Úvod do algoritmizace a programování [online]. [cit. 2014-04-28]. Dostupné z: http://www.oa-poruba.cz/vp/prg/soubory/uvod_do_programovani.pdf

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

SEZNAM PŘÍLOH

Příloha A kompletními scénáře v PDF a v programu Enchanting na CD