Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Vzorové úlohy pro senzory NXT

Marek Přikryl

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Marek Přikryl**
Osobní číslo:	**I13204**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Vzorové úlohy pro senzory NXT**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je popsat vybrané senzory k řídící jednotce Lego NXT a navrhnout a realizovat vzorové úlohy.

Programovacím jazykem je vyšší programovací jazyk Java nebo C - dle volby studenta.

Text diplomové práce bude obsahovat princip funkčnosti vybraných senzorů, popis programových prostředků pro jejich obsluhu a také popis a řešení vzorových úloh.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. BAGNALL, Brian. Intelligence unleashed: creating LEGO NXT robots with Java.
Winnipeg, Manitoba: Variant Press. ISBN 978-098-6832-208

2. KISZKA B. 1001 tipů a triků pro jazyk Java. Computer Press, 2009. ISBN 9788025124673.

Vedoucí bakalářské práce: **Ing. Michael Bažant, Ph.D.**
Katedra softwarových technologií

Datum zadání bakalářské práce: **31. října 2016**
Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Mgr. Josef Horálek, ~~Ph.D.~~
vedoucí katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

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.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 1. 5. 2017

Marek Přikryl

PODĚKOVÁNÍ

Chtěl bych poděkovat Ing. Michaelu Bažantovi, Ph.D., vedoucímu bakalářské práce, za pomoc a rady při tvorbě této práce. Dále bych chtěl poděkovat svým přátelům, za podporu po celou dobu studia.

ANOTACE

Tato práce obsahuje dvě vzorové úlohy pro stavebnici LEGO Mindstorms NXT. Programy jsou napsány za použití vyššího programovacího jazyka Java, využívajícího zásady objektově orientovaného programování. Úlohy jsou vypracovány v programovém prostředí leJOS, které je implementováno do vývojového prostředí NetBeans.

KLÍČOVÁ SLOVA

LEGO, NXT, robot, Java, algoritmy, leJOS, senzory, Segway, inverzní kyvadlo, Rubikova kostka, NetBeans

TITLE

Sample tasks for NXT sensors

ANNOTATION

This bachelor thesis contains two sample tasks for LEGO Mindstorms NXT. Programs are written in the higher programming language Java, utilizing principles of the object-oriented programming. The tasks are developed in leJOS programming environment that is implemented in a developing environment called NetBeans.

KEYWORDS

LEGO, NXT, robot, Java, algorithms, leJOS, sensors, Segway, inverted pendulum, Rubik's cube, NetBeans

OBSAH

Seznam obrázků

Seznam tabulek

Seznam zkratek

LED	Light-Emitting Diode
OOP	Objektové orientované programování
PID	Proporcionálně-integračně-derivační regulátor
SD	Secure Digital
USB	Universal Serial Bus

0 ÚVOD

Cílem bakalářské práce je vytvoření několika programů pro stavebnici LEGO Mindstorms NXT. Programy jsou napsány ve vyšším programovacím jazyku Java tak, aby dodržovaly zásady OOP. Součástí práce je také porovnání stavebnice NXT k ostatním stavebnicím ze série LEGO Mindstorms. Dále je v práci probrán princip fungování základních senzorů, které jsou dodávány k základní sadě stavebnice a dalších rozšiřujících senzorů, které jsou v práci použity.

V další kapitole práce jsou probrány softwarové možnosti pro programování robota. V kapitole je zmíněn také program a způsob programování pro výchozí program, který je se stavebnicí dodáván. V kapitole se také nachází popis hlavního programu leJOS, který je použit pro programování aplikací v této práci. Je zde popsán návod, jak správně nastavit program pro spolupráci s vývojovým prostředím NetBeans.

Praktická část práce se věnuje návrhem a řešením dvou vzorových úloh. První praktickou úlohou je samo balanční vozítko tzv. Segway. Je zde probrána problematika inverzního kyvadla a výčet možností jeho řešení. Úloha je doplněna sekvenčním diagramem včetně jeho popisu a ukázkami zdrojového kódu. Druhou úlohou je problematika hlavolamu Rubikovy kostky s popisem nejznámějších metod pro jeho řešení a základními vlastnostmi Rubikovy kostky, které je nutné znát pro jeho složení. Součástí je popis Ortegovy metody skládání kostky, což je metoda, která je v programu využita a základní popis konstrukce robota. Praktická úloha je zde doplněna UML vývojovým diagramem a popsaným sekvenčním diagramem.

1 LEGO MINDSTORMS NXT

1.1 Stavebnice LEGO Mindstorms

LEGO Mindstorms je stavebnice, jak již název napovídá, od společnosti LEGO. Tato společnost vyrábí tyto robotické stavebnice již od roku 1998. Stavebnice kombinuje prvky klasické stavebnice LEGO, programovatelné řídící jednotky a mnoho senzorů, pro přijímání vnějších signálů a motorů. Tyto stavebnice jsou vhodné pro vzdělávání a zábavu v oblasti robotiky a programování [1].

1.2 Verze LEGO Mindstorms

Firma LEGO již vydala několik verzí těchto stavebnic. První generace, s názvem Robotic Command eXplorers, obsahovala pouze dva motory, dva dotekové senzory a světelný senzor. Kostka byla programovatelná v jazyku Brick Logo. Další generací byl Lego Mindstorms NXT a později také Lego Mindstorms NXT 2.0. Tato verze s sebou přinesla řadu dalších senzorů a také možnost programovat kostku ve více programovacích jazycích (např.: Java, C). Momentálně poslední generací je Lego Mindstorms EV3, která byla vydaná 1. srpna 2013 od svého předchůdce se liší hlavně v hardwarovém vybavení řídící jednotky, hostitelským portem USB, možnosti vložení Micro SD karty a ovladačem na dálkové ovládání robota [2].

Následující tabulka ukazuje rozdíly vlastností jednotlivých verzí stavebnic.

Tabulka 1 – Porovnání jednotlivých verzí stavebnic Lego Mindstorms

Vlastnosti	**EV3**	**NXT**	**RCX**
Rok vydání	2013	2006	1998
Displej	178x128 pixelů	100x64 pixelů	segmentovaný
Procesor	TI Sitara AM1808 (ARM926EJ-S core) @300 MHz	Atmel AT91SAM7S256 (ARM7TDMI core) @48 MHz	Hitachi H8/300 @16 MHz
Paměť RAM	64 MB	64 KB	32 KB
Hostitelský port USB	Ano	Ne	Ne
Bluetooth	Ano	Ano	Ne

Zdroj: [2][3]

1.3 Řídící jednotka NXT

Hlavním aktivním prvkem celé stavebnice je řídící jednotka NXT, někdy též nazývána jako kostka NXT, která řídí veškeré činnosti robota. Název NXT vznikl zkrácením slova next (další), protože se jedná o druhou generaci ze série LEGO Mindstorms. Přijímá signály ze senzorů a na základě toho vyhodnocuje své další chování. Na přední straně kostky se nachází LCD displej, který má rozlišení 100x64 pixelů. Dále zde nalezneme 4 tlačítka, která mohou sloužit pro navigaci uživatelského rozhraní jednotky, nebo mohou sloužit jako funkční tlačítka v rámci určitého programu. Na čelní straně na pravém boku je reproduktor. Na spodní straně se nacházejí 4 vstupní porty (označené čísly 1–4), do těchto portů se zapojují senzory, které přijímají signály z okolního prostředí [3]. Rozložení řídící jednotky včetně popisků lez vidět na obrázku 1.

Obrázek 1 – Řídící jednotka NXT

Na spodní straně se také může nacházet napájecí konektor, pokud je kostka napájena z nabíjecí baterie. Kromě nabíjecí baterie může být kostka také napájena 6x bateriemi typu AA. Na vrchní straně kostky jsou 3 výstupní porty (označené A, B, C), tyto porty slouží pro zapojení motorů. Dále je zde USB port 2.0 typu A-B, viz obrázek 2, který slouží pro propojení s počítačem [3].

Obrázek 2 – USB typu A-B

1.4 Senzory

Senzory jsou hlavním prostředkem, jakým řídící jednotka zaznamenává vnější podněty a na základě těchto podnětů zpracovává informace.

Základní sada NXT obsahuje tyto senzory a motory:

- 3x servomotor,
- 1x světelný senzor,
- 1x ultrazvukový senzor,
- 1x zvukový senzor,
- 2x dotykový senzor,
- 3x světelná kostka včetně redukčních kabelů.

Kromě těchto základních senzorů je možné dokoupit senzory další (gyroskopický senzor, senzor zrychlení, úhlový senzor atd.) [4].

1.4.1 Světelný senzor

Světelný senzor slouží robotovi k měření intenzity odraženého světla a tímto také rozlišovat jednotlivé barvy. Intenzita světla je měřena v procentech v rozmezí od 0 do 100 %. Kromě tohoto lze také tento senzor použít jako senzor vzdálenosti [5].

Senzor má také k dispozici světlo vyzařující diodu (LED), která osvětluje prostředí před senzorem. Senzor také dokáže detekovat světlo, které lidské oko vidět nedokáže, jedná se například o infračervené světlo, které vydává dálkový ovladač na televizor [6].

Jak vypadá světelný senzor je možné vidět na následujícím obrázku 3.

Obrázek 3 – Světelný senzor

1.4.2 Ultrazvukový senzor

Tento senzor je hlavním prostředkem pro vnímání okolí robota, díky tomu se dokáže vyhnout překážkám. Senzor slouží pro detekování objektů a měření vzdálenosti od objektu, dále také umožňuje detekovat pohyb. Senzor dokáže měřit do vzdálenosti 255 cm s přesností ± 3 cm. Senzor funguje na základě echolokace, senzor vyšle akustickou vlnu o vysoké frekvenci a vypočítá dobu, za kterou se odražená vlna vrátí zpět [5].

Obrázek 4 – Ultrazvukový senzor

1.4.3 Zvukový senzor

Zvukový senzor snímá intenzitu zvuku v okolí a tu měří v decibelech. Senzor dokáže měřit zvuky do úrovně 90 decibelů (to odpovídá hluku běžící sekačky na trávu). Senzor zvládne vyjádřit intenzitu také procentuálně v dBA, kdy 4–5 % je jako ticho v pokoji a 30–100 % je jako prostředí s hlasitou hudbou [5].

Obrázek 5 – Zvukový senzor. Zdroj: [7]

1.4.4 Dotykový senzor

Senzor funguje jako tlačítko, které vrací hodnotu pravda / nepravda na základě zmáčknutí / uvolnění senzoru [5].

Obrázek 6 – Dotykový senzor

1.5 Motory

Robotův pohyb umožňují interaktivní servomotory, které se zapojují do výstupních portů. Motory lze využívat jako kola pro pohyb robota nebo jako pomocná ramena vykonávající určitou činnost. Podle toho lze také motory řídit buď nezávisle na sobě nebo synchronizovaně. Motory mají navíc také vestavěný rotační senzor, pomocí kterého lze určit hodnoty úhlu, nebo počet otáček o který se motory otočily. Motor může sloužit i jako vstupní zařízení, který informuje o úrovni natočení [5].

Obrázek 7 – Interaktivní motor

1.6 Rozšiřující senzory

Ke stavebnici je možné dokoupit celou řadu dalších senzorů, které rozšiřují základní sadu. Mezi tyto senzory patří například gyroskopický senzor, teplotní senzor, senzor zrychlení atd. Všechny rozšiřující senzory použity v této práci jsou od firmy HiTechnic Products. Pro použití rozšiřujících senzorů v programu NXT-G, je potřeba stáhnout příslušný blok a importovat ho do programu. Bloky lze stáhnout například z oficiálních stránek výrobce HiTechnic Products na adrese http://www.hitechnic.com/products [8][9].

1.6.1 Gyroskopický senzor

Tento senzor obsahuje jednoosý gyroskopický senzor, který detekuje rotaci a vrací hodnotu reprezentující rotaci vyjádřenou v počtu stupňů za sekundu. Hodnota rotace se může pohybovat v rozmezí ± 360 ° za sekundu. Rotace může být vyhodnocena přibližně 300krát za sekundu. Ke správnému fungování je potřeba senzor umístit černou částí nahoru, jak je ukázáno na obrázku 8 [8].

Obrázek 8 – Gyroskopický senzor

1.6.2 Barevný senzor

Barevný senzor umožňuje rozlišovat barvy a rozpoznat až 18 barev. Senzor se namíří na určitý povrch a ten vrátí číslo reprezentující danou barvu. Číslo 0 znázorňuje nejtmavší barvu,

černou, a číslo 17 barvu nejsvětlejší, bílou. Senzor disponuje LED žárovkou, která svítí bílou barvou a umožňuje lepší rozpoznání barev. Barevný senzor je potřeba nastavit, aby nemířil na povrch pod úhlem 90 °. Natočení zabraňuje, aby světlo odražené zpět od povrchu nezpůsobilo špatné načtení barvy. Senzor dokáže snímat barvy až 100krát za vteřinu [9].

Obrázek 9 – Barevný senzor. Zdroj: [9]

1.7 Spojovací kabely

Kabely jsou pro práci nezbytné, a proto jsou také součástí stavebnice. Kabely se používají k propojení senzorů a motorů k řídící jednotce. V základní sadě se nachází sada sedmi spojovacích kabelů o třech různých délkách. Krátké kabely mají délku 20 cm, střední kabely jsou délky 35 cm a nejdelší kabely mají délku 50 cm. K této základní sadě je možné dokoupit i další kabely o délkách 12, 16, 70 a 90 cm. Kabely je možné vyrobit z klasických telefonních kabelů RJ12, hlavním rozdílem mezi těmito kabely je konektor. Kabely ve stavebnici mají u konektoru posunutou horní část, která drží kabel v portu, dále od středu [10].

2 SOFTWAROVÉ MOŽNOSTI

Při pořízení stavebnice je dodáváno vývojové prostředí nazvané NXT-G, které je velmi vhodné pro začátečníky, protože má velice jednoduché a intuitivní ovládání. Kromě tohoto je možné robota programovat pomocí programovacích jazyků (Java, C/C++, Python…).

Řídící jednotka je vybavena speciálním vnitřním softwarem, tzv. firmwarem, který slouží pro správný chod aplikací. Jedním z úkolů firmwaru je spouštění programů a komunikace s konkrétním programovacím jazykem, proto je důležité mít v řídící jednotce vždy aktuální verzi firmwaru pro daný programovací jazyk. Oficiální verzi je možné stáhnout z oficiálních stránek LEGO[1], tato verze je vhodná pro vývojové prostředí NXT-G [3].

Firmware od společnosti LEGO je vyvíjen jako open source, tzn. lze si stáhnout zdrojové soubory a upravit je. Pokročilý uživatelé si mohou dále stáhnout tyto vývojové sady:

- vývojová sada pro Bluetooth,
- vývojová sada pro hardware,
- vývojová sada pro software [11].

2.1 Programová prostředí

Robota lze programovat v různých programovacích jazycích, viz tabulka 2, proto je nutné nejdříve určit, jak bude robot naprogramován. Pokud se bude jednat o jednodušší program, je možné použít některý z grafických prostředí pro vývoj. V případě složitějších programů je možné psát program ve vybraném programovacím jazyce.

Tabulka 2 – Přehled programovacích prostředí

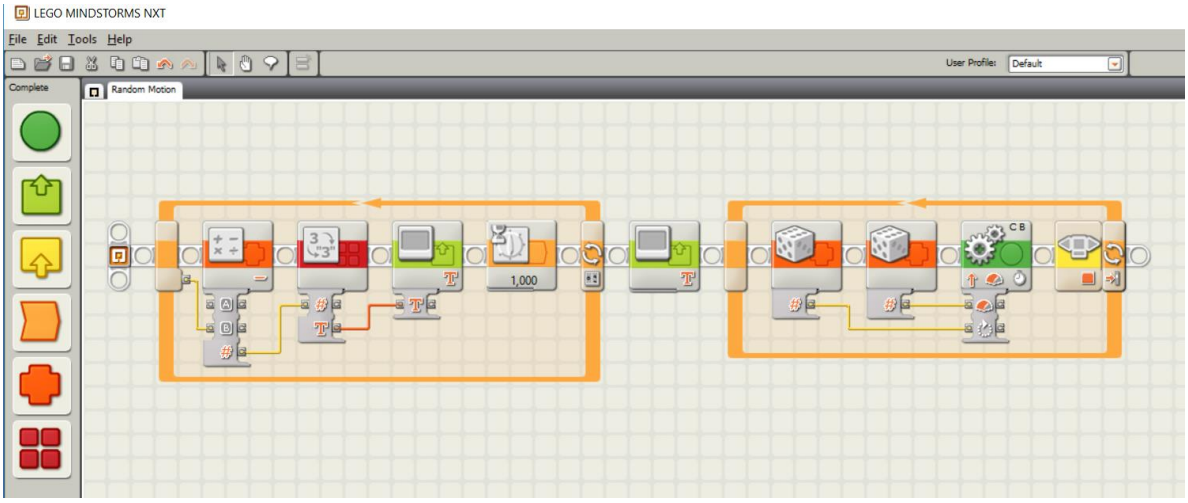
Vlastnosti	NXT-G	leJOS	ROBOTC	NXT-Python	Ruční ovládání
Programovací jazyk	Grafické rozhraní	Java	C	Python	Grafické rozhraní
Náročnost	Jednoduché	Pokročilý	Středně pokročilý	Pokročilý	Jednoduché
Zařízení	NXT, EV3	NXT, EV3	RCX, NXT, EV3	NXT	NXT, EV3
Platforma	Windows, Mac	Windows, Linux, Mac	Windows	Windows, Linux, Mac	Řídící jednotka

Zdroj: [2][12]

[1] https://www.lego.com/cs-cz/mindstorms/downloads

2.1.1 NXT-G

NXT-G je základní grafické uživatelské rozhraní pro programování řídící jednotky. Tento software je dodáván společně se stavebnicí LEGO a je vhodný pro úplné začátečníky, sestavování programu zde probíhá na základě spojování bloků akcí robota a není tedy zde potřeba znalostí programovacího jazyka [13].



Obrázek 10 – Program NXT-G

Na obrázku číslo 10 je ukázka programu vytvořeného programem NXT-G. Tento program v prvním cyklu provede odpočet od 3 do 1, přičemž každé číslo je převedeno na text a zobrazeno na obrazovku řídící jednotky. Po odpočtu se na obrazovce zobrazí nápis Start. V druhém cyklu, na pravé polovině obrazovky, program vygeneruje 2 náhodná čísla, která slouží jako vstupní parametry pro pohyb motorů. První hodnota udává, jak dlouho se budou motory pohybovat a druhá hodnota, jakou rychlostí se budou pohybovat.

Program NXT-G obsahuje také podrobné návody pro stavbu robotů, viz obrázek 11, je možné si tak krok za krokem pomocí přiloženého manuálu postavit robota, a to včetně vytvoření programu. Dále program umožňuje přehrání firmwaru kostky NXT.

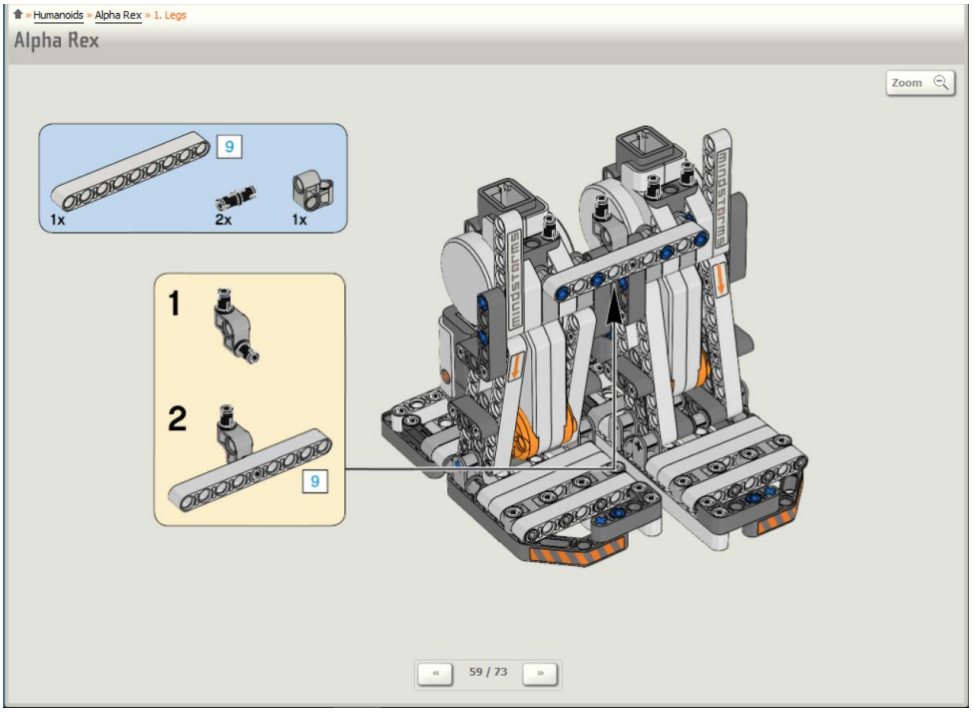
Výhody:

- jednoduché grafické ovládání,
- není potřeba znát programovací jazyky,
- přiložené manuály pro stavbu a programování robotů,
- dodávané společně se stavebnicí.

Nevýhody:

- není příliš vhodný při psaní rozsáhlejších programů,
- komplikované definování proměnných,

- programy zabírají více místa, rychleji dochází kapacita jednotky,
- absence polí jakožto datové struktury.



Obrázek 11 – Program NXT-G, manuál pro stavbu robota

2.1.2 LeJOS NXJ

LeJOS NXJ je programovací prostředí pro LEGO Mindstorms NXT, které umožňuje vytvářet programy v jazyce Java. Toto prostředí využívá vlastní firmware, který obsahuje Java Virtual Machine. Pomocí toho lze vytvářet programy stejným způsobem jako Java aplikace. LeJOS je projekt open source, tzn. program je možné zdarma stáhnout[2]. Program lze nainstalovat na většinu operačních systémů, Windows, Mac OS, Linux. Vedle zde popsané verze NXJ, existují také verze pro novější model stavebnice EV3, i pro starší model RCX [12].

Výhody:

- možnost programování v jazyce Java,
- podpora objektově orientovaného programování,
- zdarma ke stažení,
- multiplatformní,
- podpora více vláknového programování.

Nevýhody:

- znalost jazyka Java,

[2] https://sourceforge.net/projects/nxt.lejos.p/files/

- nemá vlastní IDE, pouze jako pluginy do vývojových prostředí (NetBeans, Eclipse),
- potřeba nahrát jiný firmware,
- komplikovanější příprava prostředí.

Protože veškerý program běží na řídící jednotce NXT, je při psaní rozsáhlejších programů potřeba řešit chyby, které při běhu programu mohou nastat. leJOS nabízí několik možností, jak provádět debug a jak odchytávat výjimky. Pro debugování programu je použita statická třída RConsole, která posílá výpis do programu nxjconsole na počítači. Po navázání spojení počítače a řídící jednotky lze kdekoliv v kódu zavolat metodu println ze třídy RConsole a na počítači sledovat výpis. Spojení může být navázáno prostřednictvím kabelu USB nebo technologií Bluetooth. Následující ukázka znázorňuje příklad výpisu do počítače [14].

```
public static void main(String[] args) {
        RConsole.openUSB(0);    //Otevře spojení s počítačem
        RConsole.println("Vypis do konzole");
        for (int i = 0; i < 10; i++) {
            RConsole.print(i + ", ");   //Vypíše na počítači hodnoty i
        }
        RConsole.close();   //Skončí spojení s počítačem
    }
```

V případě, kdy se program dostane do bodu, kde není zachycena výjimka, vypíše se na obrazovce řídící jednotky zmenšený popis chyby a také informace s popisem místa výskytu chyby. Následující ukázka znázorňuje, jak vypadá výpis chyby na řídící jednotce.

```
Exception: 28
  at:  2(11)
  at:  3(1)
  at:  1(1)
```

První řádek udává, číslo třídy výjimky, pro kterou došlo k chybě programu. V tomto případě došlo k chybě ve třídě číslo 28, které označuje přetečení indexu pole. Druhý a další řádky popisují výskyt chyby, každý řádek označuje číslo metody, zde došlo k chybě ve druhé metodě, číslo v závorce udává lokaci v metodě. Jednotlivá čísla výjimek a čísla metod lze nalézt ve výpisu v NetBeans po nahrání programu do řídící jednotky [14]. Na obrázku 12 je ukázka číslování metod a na obrázku 13 seznam očíslovaných tříd chyb, k jakým v programu může dojít.

```
Method 0: java.lang.Thread.run() PC 652 Signature id 1
Method 1: org.lejos.example.naZkousku.main(java.lang.String[]) PC 672 Signature id 0
Method 2: org.lejos.example.naZkousku.metoda1() PC 676 Signature id 175
Method 3: org.lejos.example.naZkousku.metoda2() PC 691 Signature id 176
```

Obrázek 12 – Ukázka výpisu pro číslování metod

Obrázek 13 – Ukázka výpisu se seznamem chyb

2.2 Příprava prostředí

Postup instalace vychází z návodu [15]. Všechny programy v této práci jsou vytvořeny za pomoci programu leJOS (popsáno výše) ve vývojovém prostředí NetBeans.

Požadavky:

- Java Development Kit, verze 1.6 a vyšší, leJOS pracuje pouze s 32 bitovou verzí. Nejnovější verzi lze stáhnout z oficiálních stránek Oracle[3].
- Fantom Driver, ovladač pro připojení přes USB rozhraní. Ovladač lze stáhnout ze stránek LEGO Mindstorms[4], nebo je dodáván společně se stavebnicí.
- NetBeans, open-source vývojové prostředí[5] pro tvorbu nejen Java aplikací.

2.2.1 Návod instalace leJOS

V případě, že jsou splněné všechny požadavky, které jsou popsány výše, je možné přistoupit k samotné instalaci programu leJOS. Nejnovější verze je 0.9.1, ale tato verze již nemá podporu pluginu pro NetBeans, z toho důvodu bude použita verze 0.9.0 [16].

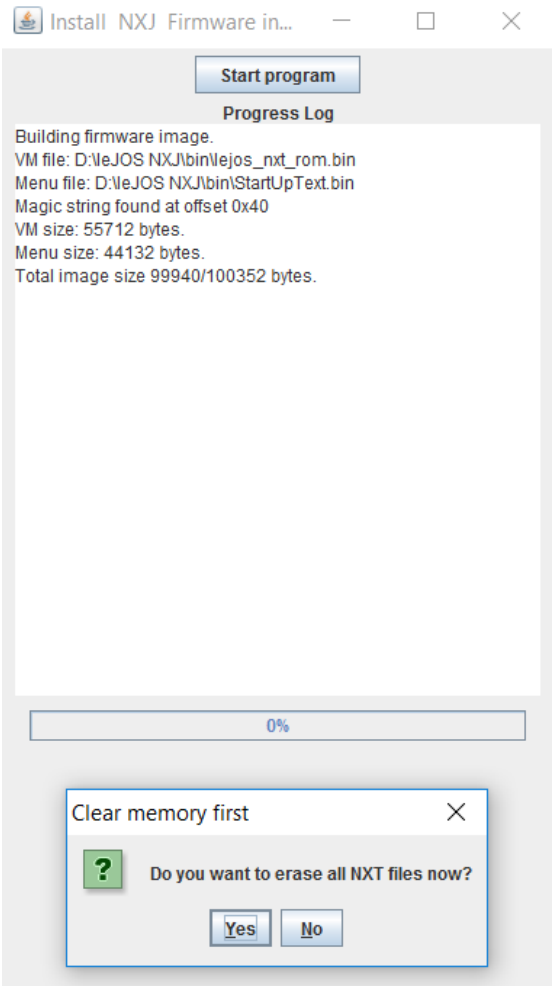
Samotná instalace leJOS není náročná, stačí se řídit instrukcemi v instalátoru. Po doběhnutí instalace se otevře okno pro přehrání firmwaru řídící jednotky NXT. Toto okno lze později kdykoliv vyvolat zadáním nxjflashg do příkazové řádky, nebo ve složce bin v adresáři, kde je nainstalovaný leJOS otevřít nxjflashg.bat.

Výchozí cesta k adresáři je *C:\Program Files (x86)\leJOS NXJ*

Přehrání firmwaru trvá přibližně 10 sekund, i přesto je nutné zkontrolovat, že je kostka dostatečně nabitá. V případě, že by došlo k narušení přehrávání, ať už z důvodu vybité baterie

[3] http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
[4] https://www.lego.com/cs-cz/mindstorms/downloads
[5] https://netbeans.org/downloads/

nebo vytržení USB kabelu z kostky, by toto přerušení mohlo nenávratně NXT kostku poškodit. Při přehrání firmwaru dojde ke smazání veškerých dat na řídící jednotce.



Obrázek 14 – Aplikace na přehrání firmwaru

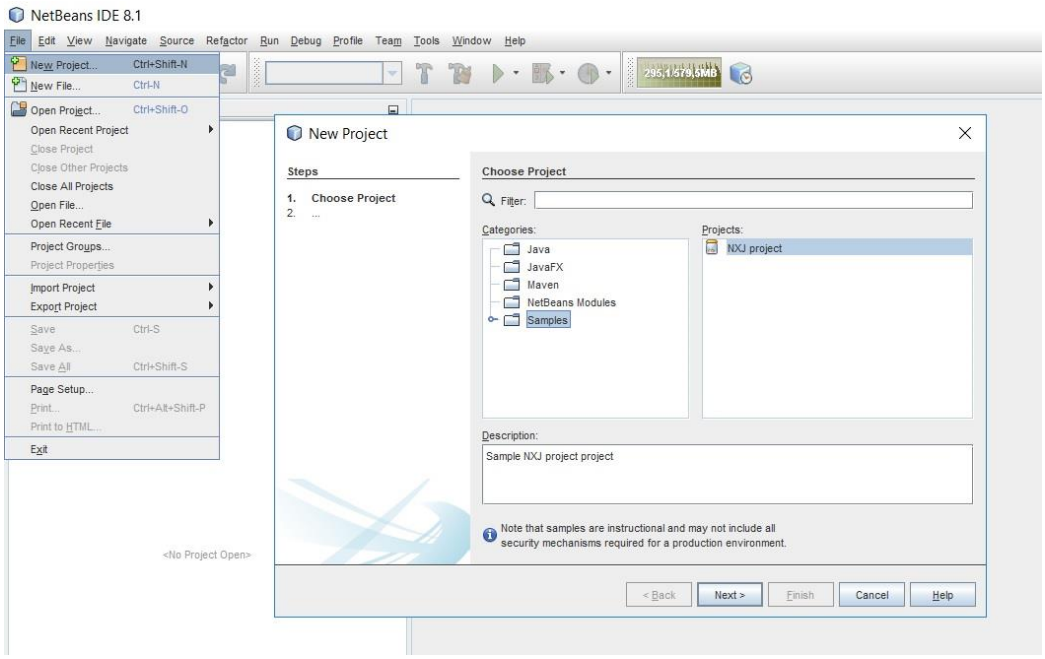
2.2.2 Příprava NetBeans

Abychom mohli NetBeans správně používat společně s programem leJOS, je potřeba nejdříve nahrát plugin. Tento plugin se nainstaloval společně s instalací leJOS a lze ho nelézt ve složce leJOSNXJProjects. Pokud se v instalátoru nezměnila jeho výchozí cesta, adresář je umístěn v *C:\Users\"jmeno_uzivatele"*. Úplná cesta k pluginu je tedy:

C:\Users\"jmeno_uzivatele"\leJOSNXJProjects\NXJPlugin\build\nxjplugin.nbm

Do NetBeans se plugin přidá v záložce Tools – Plugins – Downloaded – Add Plugins…, zde se vybere cesta k pluginu.

Po přidání pluginu je již možné vytvářet projekty NXJ následujícím způsobem, File – New Project – Categories – Samples – NXJ project, viz obr. 15 [17].

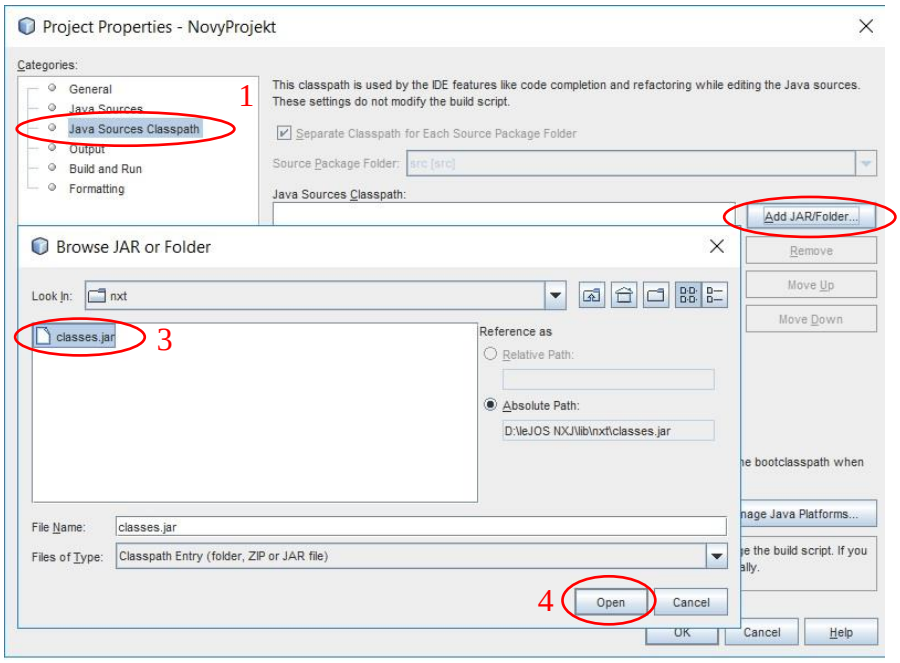


Obrázek 15 – Vytváření NXJ projektu

Po takto vytvořeném projektu je potřeba zadat cestu k leJOS balíčkům, abychom mohli pracovat s leJOS třídami a metodami. Pravým tlačítkem myši klikneme na nově vytvořený projekt a pak Properties – Java Sources Classpath – Add JAR/Folder. Zde je potřeba zadat cestu k souboru classes.jar. Soubor se nachází ve složce, kam jsme nainstalovali leJOS NXJ v adresáři *lib/nxt*. Úplná cesta k souboru je

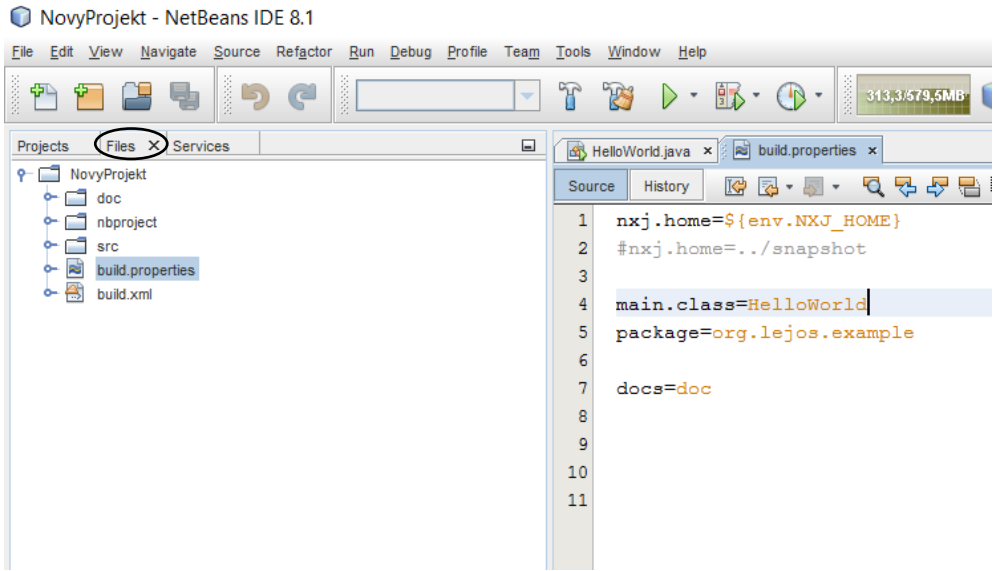
C:\Program Files (x86)\leJOS NXJ\lib\nxt\classes.jar

Ukázka přidání je znázorněna na obrázku 16, včetně znázornění jednotlivých kroků.



Obrázek 16 – Přidání leJOS balíčku

Dalším postupem je nastavení cesty k leJOS balíčku pro build.xml, pomocí kterého se nahraje program do NXT kostky. Je potřeba upravit soubor build.properties, který se nachází v adresáři, kde se vytvořil nový projekt. Cestu k adresáři lze zjistit kliknutím pravým tlačítkem na projekt pak Properties – General – Project Folder. Prvím možným způsobem je tento soubor najít a poté ho upravit například v programu PSPad nebo WordPad. Druhým způsobem, jak upravit soubor je pomocí NetBeans. V záložce Files se otevře složka s názvem projektu, ve které se soubor nachází, viz obr. 17 [17].



Obrázek 17 – Adresář se souborem build.properties

Na prvním řádku tohoto souboru, parametr nxj.home, je potřeba nastavit cestu k adresáři, kde je nainstalovaný leJOS program. Výchozí hodnota tohoto parametru je *../snapshot*, to je potřeba změnit. Je potřeba zkontrolovat, aby první řádek nekončil mezerou. Cestu k adresáři lze zapsat přímo do souboru za znak "=", cesta musí obsahovat normální lomítka (/) namísto zpětných lomítek (\). Pokud se změní název hlavní třídy programu, je potřeba ho změnit také zde, na řádku main.class. Cestu je možné také zadat pomocí proměnných prostředí systému. Proměnné prostředí lze nalézt v Ovládací panely – Systém – Upřesnit nastavení systému – Záložka Upřesnit – Proměnné prostředí. V horní části jsou definovány uživatelské proměnné do, kterých se přidá nová proměnná, viz tabulka 3 [17].

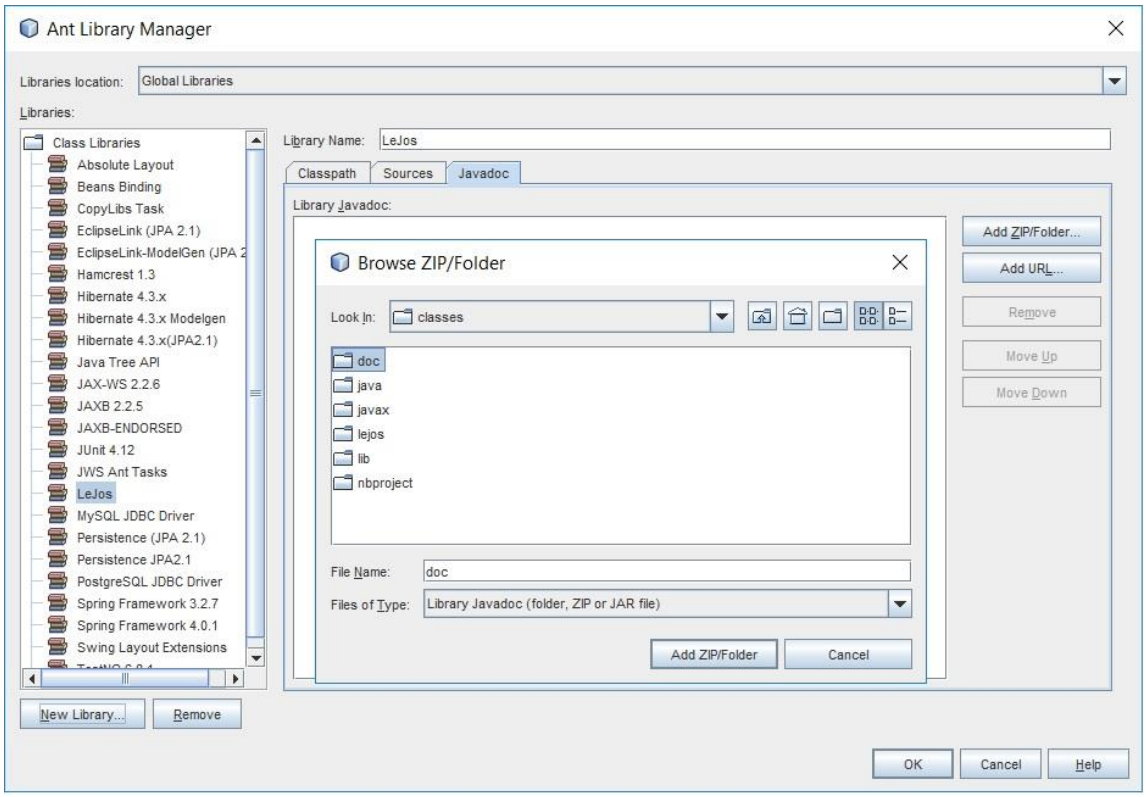
Tabulka 3 – Proměnné prostředí

Název	Hodnota
NXJ_HOME	Cesta, kde je nainstalován leJOS.

Zdroj: [15]

Poslední věcí, kterou je potřeba udělat, je přidat dokumentaci (JavaDoc). Po přidání dokumentace se při psaní kódu budou zobrazovat nápovědy a možnosti k novým metodám a třídám. Nejdříve je potřeba vytvořit novou knihovnu, to se provede v Tools – Libraries – New Library. Do nově vytvořené knihovny se do sekce Classpath přidá classes.jar (stejný balíček jako se přidával do projektu viz obrázek 16). Do sekce Javadoc se přidá adresář doc, který se nachází v

C:\Users\"jmeno_uzivatele"\leJOSNXJProjects\classes\doc



Obrázek 18 – Vložení Javadoc do NetBeans

Po tomto nastavení jsou NetBeans připraveny, pro nahrání programu do NXT kostky stačí připojit USB kabel, nebo spárovat s PC přes Bluetooth, a kostku zapnout. V NetBeans vybrat položku Run – Run Main Program (nebo použít klávesovou zkratku F6), program se zkompiluje, nahraje do kostky a spustí. Tato operace je nastavena jako výchozí, toto chování se dá změnit stisknutím pravého tlačítka na název projektu a zvolit Properties – Build and Run. Zde se nachází vlastnost Run Project, která ovlivňuje, co se s programem stane po spuštění. Možnost upload program nahraje, ale ihned po nahrání ho nespustí, tuto možnost je lepší nastavit u pohyblivých robotů. Jednotlivé možnosti lze dále podrobněji nastavit v souboru build.xml [17].

3 PRAKTICKÁ ČÁST

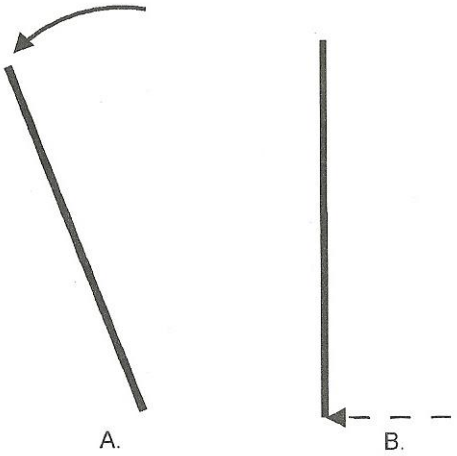
Součástí této práce je vypracování dvou úloh, pro stavebnici LEGO Mindstorms NXT a jejich naprogramování v programovacím jazyce Java. Programy pro praktickou část byly vytvořeny v programu NetBeans, do kterých byl implementován program leJOS.

3.1 1. Úloha – Vozítko Segway

První úlohou je program pro fungování samo balančního vozítka, tzv. Segway. Segway, přesněji Segway Personal Transporter, je dvoukolový elektrický dopravní prostředek pro jednu osobu. Segway ke svému pohybu využívá dynamické stabilizace. Stabilitu zajišťuje soustava pěti gyroskopů, která 100x za sekundu vyhodnocuje polohu základny a na základě těchto dat ovládá motory tak, aby byla plošina stále v rovině [18].

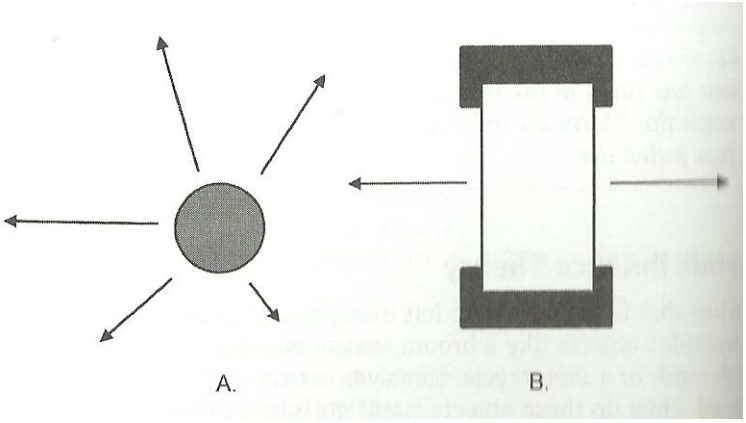
Obrázek 19 – Segway, osobní transportér. Zdroj: [19]

Dynamické balancování se zakládá na udržení pozice, do té doby, než objekt začne padat určitým směrem. Poté je potřeba přesunout centrum gravitace do směru, kterým objekt padá. Jak je znázorněno na obrázku číslo 20, pokud se tyč začne naklánět směrem doleva, je potřeba rychlý pohyb základny tyče také směrem doleva, díky čemuž dojde k obnovení rovnováhy [6].



Obrázek 20 – Ukázka vyrovnávání objektu. Zdroj: [6]

Balancování tyče je komplexní, protože se tyč může naklánět ve všech směrech, tedy v rozsahu 360 °. Přidáním nápravy a dvou kol lze docílit omezení náklonu pouze v jedné rovině, tzn. objekt má možnost náklonu pouze doleva nebo doprava. Takový postup je použit i u vozítka Segway [6].



Obrázek 21 – Ukázka možnosti směru pádu A. tyče, B. Segway. Zdroj: [6]

3.1.1 Vybavení robota

Pro tuto úlohu je potřeba kostka NXT, gyroskopický senzor, 2 motory, ultrasonický senzor a kabely pro propojení senzorů s kostkou NXT. U tohoto typu robota se doporučuje použít propojovací kabely menší velikosti (20 cm), delší kabely by mohli robotovi překážet v pohybu. Dále je nutné znát průměr použitých kol, pro správný výpočet algoritmu na udržení rovnováhy robota.

3.1.2 Cíl úlohy

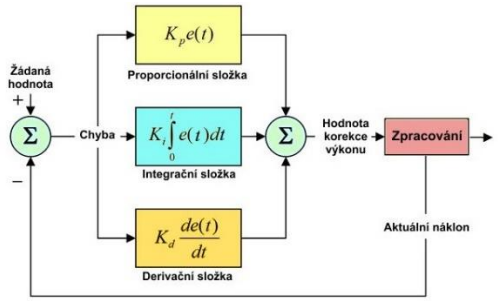
Cílem této úlohy je zajistit robotovu stabilitu, při použití pouze nápravy a dvou kol. Robot musí zvládat základní pohyby po rovné ploše. Za pomoci ultrasonickému senzoru, musí být schopen reagovat na okolní prostředí.

3.1.3 Řešení úlohy

K problému tohoto řešení je možné přistupovat několika způsoby.

1. PID regulátor

   Tento regulátor patří mezi základní regulátory pro řízení vozidel a zařízení. Písmena v názvu představují proporcionální, integrační a derivační složku chování regulátoru.
   P – aktuální chyba,
   I – akumulace přechozích chyb,
   D – predikce budoucích chyb [20].
   Následující obrázek znázorňuje výpočet regulační smyčky za pomoci jednotlivých složek regulátoru.



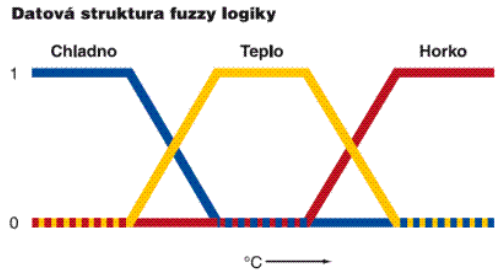
Obrázek 22 – Regulační smyčka PID regulátoru. Zdroj: [21]

2. Lineární kvadratický regulátor

   Lineární kvadratický regulátor je v literatuře též označován LQR. Tento regulátor se používá k nalezení optimální zpětné vazby ve stavovém prostoru. Máme nějaký aktuální stav regulovaného systému a chceme přejít do výchozího stavu [17].

3. Fuzzy regulátor

   Fuzzy řízení je postaveno na logice fuzzy. Tato logika si získala značnou popularitu koncem osmdesátých a na začátku devadesátých let. Nejúspěšnější aplikace jsou v řízení a regulaci, dále také v rozpoznávání obrazů, klasifikaci a rozhodování. Hlavním zdrojem je, že fuzzy logika umožňuje zahrnout nepřesnost, což je znázorněno na obrázku 23 [22].



Obrázek 23 – Příklad fuzzy logiky. Zdroj: [23]

Tento příklad je řešen pomocí upraveného regulátoru PID. Jak bylo popsáno výše, regulátor se skládá ze tří složek, které tvoří regulační smyčku. Čím více se bude zvětšovat proporcionální složka, tím více se bude projevovat rozdíl mezi požadovaným a aktuálním náklonem. Tomu se říká je tzv. chybový signál, a robot se bude srovnávat. Po dosažení určité hodnoty proporcionální složky začne robot oscilovat, bude se překlápět na jednu stranu a poté na druhou. Integrační složka oproti proporcionální nezačne působit okamžitě, integrace je zastoupena sumací všech předchozích chybových signálů. Po přidání této složky již robot nebude tolik oscilovat, ale stále bude náchylný při aktuální změně pohybu. Poslední částí je derivační složka regulátoru, ta představuje rozdíl aktuálního chybového signálu a předchozího chybového signálu. Zajišťuje působení proti jakékoliv změně náklonu robota. Po přidání této složky již robot bude schopen rychle zareagovat na změnu stavu a zabrání pádu. Aby složky správně kooperovaly, je potřeba najít a nastavit správné hodnoty konstant [21].

Základní a nejdůležitější částí je gyroskopický senzor, který měří aktuální hodnotu náklonu robota. Senzor je před použitím potřeba zkalibrovat. Kalibrací gyroskopického senzoru je získána hodnota offsetu, která se dále používá pro nastavení úhlové rychlosti. Při provádění kalibrace je nutné, aby se se senzorem nehýbalo. Pro vyřešení této úlohy lze alternativně použít světelný senzor nebo senzor akcelerace [8].

Příklad kalibrace:

```
private void kalibraceGyro() {
        int pocetOpakovani = 200;
        double soucetHodnot;
        int minHod, maxHod, gyroHodnota;

        LCD.clear();
        LCD.drawString("Kalibrace", 0, 0);

        do {
            soucetHodnot = 0.0;
            minHod = Integer.MAX_VALUE;
            maxHod = Integer.MIN_VALUE;
            for (int i = 0; i < pocetOpakovani; i++) {
                gyroHodnota = gyroskop.readValue();

                maxHod = gyroHodnota > maxHod ? gyroHodnota  : maxHod;
                minHod = gyroHodnota < minHod ? gyroHodnota : minHod;

                soucetHodnot += gyroHodnota;
            }
        } while ((maxHod - minHod) > 1);

        aktOffset = soucetHodnot / pocetOpakovani + 1;
        uhelGyroskopu = 0;
    }
```

Tento senzor ale neměří přímo hodnotu vychýlení ve stupních, ale úhlovou rychlost ve stupních za sekundu. Pro naměření úhlu je potřeba znát hodnotu intervalu mezi jednotlivými cykly ve kterém se měří úhlová rychlost.

Výpočet úhlu lze získat tímto způsobem:

```
private void vypocetIntervalu(long cLoop) {
        if (cLoop == 0) {
            intervalCyklu = 0.0055;
            casStartu = System.currentTimeMillis();
        } else {
            intervalCyklu = (System.currentTimeMillis() - casStartu) /
(cLoop * 1000.0);
        }
    }
```

```
private void updateGyroData() {
        float aktUhlovaRychlost = gyroskop.getAngularVelocity();
        aktOffset = OFFSET * aktUhlovaRychlost + (1 - OFFSET) * aktOffset;

        uhlovaRychlost = aktUhlovaRychlost - aktOffset;

        uhelGyroskopuGlobal += uhlovaRychlost * intervalCyklu;

        uhelGyroskopu = uhelGyroskopuGlobal;
    }
```

Samotná stabilizace a vyrovnávání robota poté probíhá ve smyčce, kde se vypočítávají všechny důležité parametry pro zachování rovnováhy. Pro získání síly motorů je použit hlavní výpočet:

```
silaMotoru = (int) ((KGYROSPEED * uhlovaRychlost
            +        KGYROANGLE * uhelGyroskopu) / pomerKola
            +        KPOS *       poziceMotoru
            +        KDRIVE *     cilovaRychlost
            +        KSPEED *     rychlostMotoru);
```

uhlovaRychlost – hodnota získána z gyroskopického senzoru udává úhlovou rychlost ve stupních za vteřinu. Pokud se robot bude naklánět dopředu tato hodnota bude nabývat kladných hodnot.

uhelGyroskopu – hodnota úhlu robota, udávaná ve stupních.

poziceMotoru – hodnota pozice motoru ve stupních. Pomocí této hodnoty se robot udržuje na konkrétním místě.

cilovaRychlost – tato hodnota je používaná pro rozpohybování robota dopředu nebo dozadu. Zajišťuje také stabilitu robota při rozjezdu a při zastavení, aby nedošlo k pádu.

rychlostMotoru – hodnota udávaná ve stupních za sekundu. Hodnota zajistí zklidnění robota, když se příliš rozkmitá.

pomerKola – ve stavebnici lze nalézt více velikostí kol a pro zachování správné hodnoty rovnice je potřeba tuto velikost zahrnout do rovnice. Hodnota se porovnává k velikosti kola 5,6 cm, velikost kola lze zvolit při startu programu na NXT kostce.

Výsledkem výpočtu je hodnota, která je pak přiřazena jako síla pro jednotlivé motory. Hodnota musí být menší jak 100, protože je tato hodnota předána jako procentuální vyjádření pro sílu motoru. Pro správný výsledek z rovnice jsou důležité hodnoty konstant, tyto hodnoty nejsou pevně dané. Je možné tyto hodnoty měnit a tím hledat správnou kombinaci pro co nejlepší vybalancování robota. Tyto hodnoty je možné nalézt na stránkách výrobce senzoru.

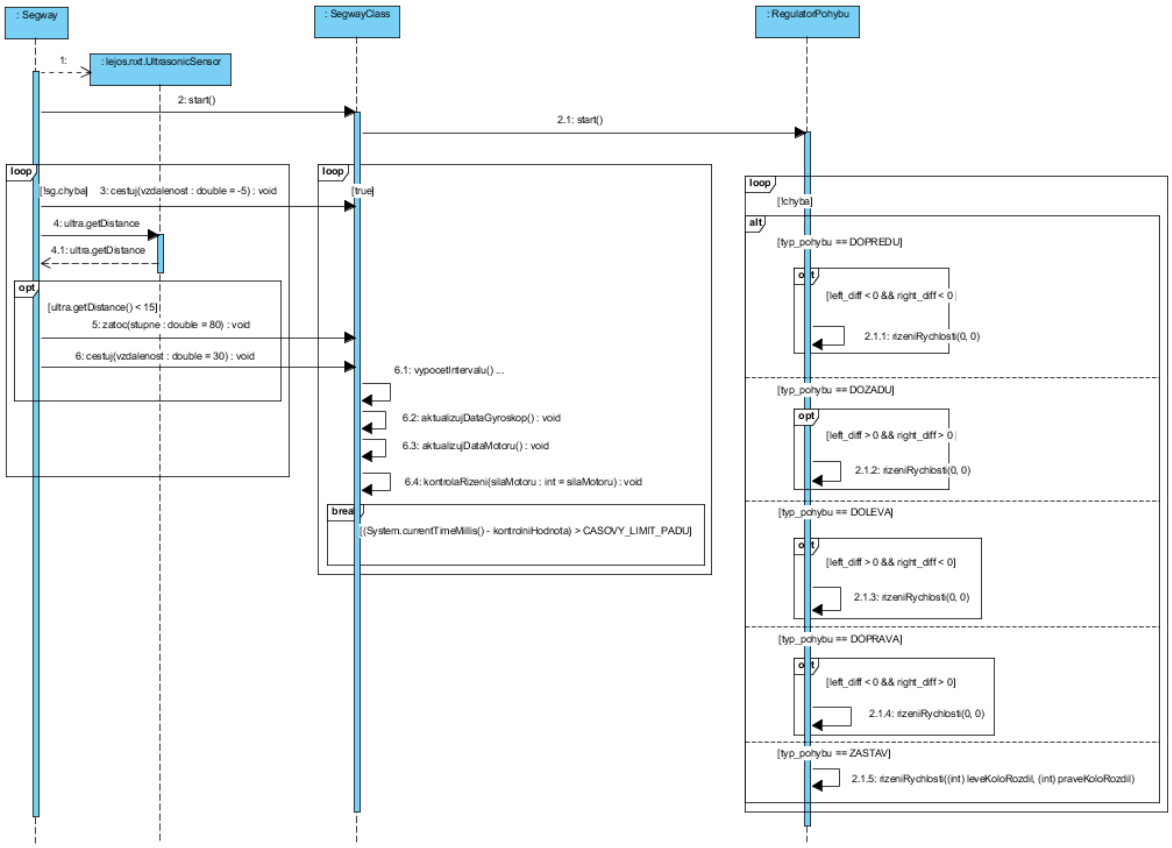
Konstantní hodnoty:

```
private final double KGYROANGLE = 7.5;
private final double KGYROSPEED = 1.15;
private final double KPOS       = 0.07;
private final double KSPEED     = 0.1;
private final double KDRIVE     = -0.02;
```

[24]

Sekvenční diagram na obrázku 24 znázorňuje životní cyklus programu a spolupráci více vláken. První, hlavní, vlákno ve smyčce kontroluje, jestli se před robotem neobjevila

překážka. Druhé vlákno řídí balancování robota a třetí vlákno udává směr jakým se robot pohybuje.



Obrázek 24 – Sekvenční diagram k 1. úloze

Na sekvenčním diagramu jsou znázorněny nejdůležitější body programu. Program se skládá ze tří tříd, každá třída si spustí vlastní vlákno, které běží až do skončení programu. Ve třídě Segway metodou číslo 2 se spustí vlákno v metodě SegwayClass a poté se spustí smyčka (loop). Smyčka zajišťuje že se robot pohybuje dopředu pomocí metody číslo 3 a poté zkontroluje vzdálenost od překážky před robotem. Pokud je překážka v menší vzdálenost jak 15 cm, smyčka vstoupí do podmínky (opt). V podmínce robot začne couvat a potě se otočí. Třída SegwayClass spustí metodou číslo 2.1 vlákno ve třídě RegultorPohybu a poté vstoupí do smyčky (loop). Tato smyčka udržuje robota ve stabilizované poloze a zároveň přijímá signály ze třídy Segway na změnu směru pohybu. Když je ve smyčce splněna podmínka (break), znamená to, že robot spadl a smyčka končí. Tímto také vyšle signál ostatním třídám, aby ukončili svoji činnost. Třída RegulatorPohybu reaguje ve smyčce na změnu pohybu a nastaví sílu motorům, dokud robot nedosáhne požadované vzdálenosti.

3.1.4 Závěr k úloze číslo 1

Nejdůležitějším bodem tohoto programu bylo pochopení a vyřešení rovnice pro vybalancování robota. Dále bylo důležité najít správnou kombinaci konstantních hodnot pro co nejlepší udržení rovnováhy a chování robota. Důkaz funkčnosti robota lze vidět na obrázku 25.

Možnosti vylepšení:

- Přidání dalšího sensoru, například dotykového, který by snímal zadní část robota, aby při couvání zjistil překážku a vyvaroval se poté pádu.
- Přidání dalšího motoru, který by robotovi pomohl vstát ze země, kdyby spadl nebo na začátku programu, aby se dokázal sám postavit.
- Možnost dálkového ovládání, které by umožňovalo řízení robota například prostřednictvím mobilního telefonu přes spojení Bluetooth.

Obrázek 25 – Robot ve stabilizované poloze

3.2 2. Úloha – Řešitel Rubikovy kostky

Další úlohou této práce je postavení robota, který obdrží hlavolam Rubikovy kostky a vyřeší jej v co nejkratším možném čase. Rubikova kostka je mechanický hlavolam, jehož autorem je Ernő Rubik, který hlavolam vynalezl v roce 1974 a o rok později získal na kostku patent. Hlavolam je krychle, 3x3, která se skládá z 26 malých krychliček, které jsou obarveny šesti různými barvami. Celkový počet kombinací, jak lze kostku poskládat je přes 43 triliónů ($43x10^{18}$). Tento hlavolam je známý po celém světě a pořádají se celosvětové závody v rychlostním skládání, nejrychlejší složená kostka člověkem byla za 4.73 vteřin [25].

Obrázek 26 – Nesložená Rubikova kostka

3.2.1 Vybavení robota

Pro tuto úlohu jsou zapotřebí 3 servomotory, světelný sensor, ultrasonický sensor. Kabely pro připojení sensorů a motorů ke kostce. Pro lepší detekování barev na kostce je lepší použít barevný senzor, například od firmy HiTechnic Products.

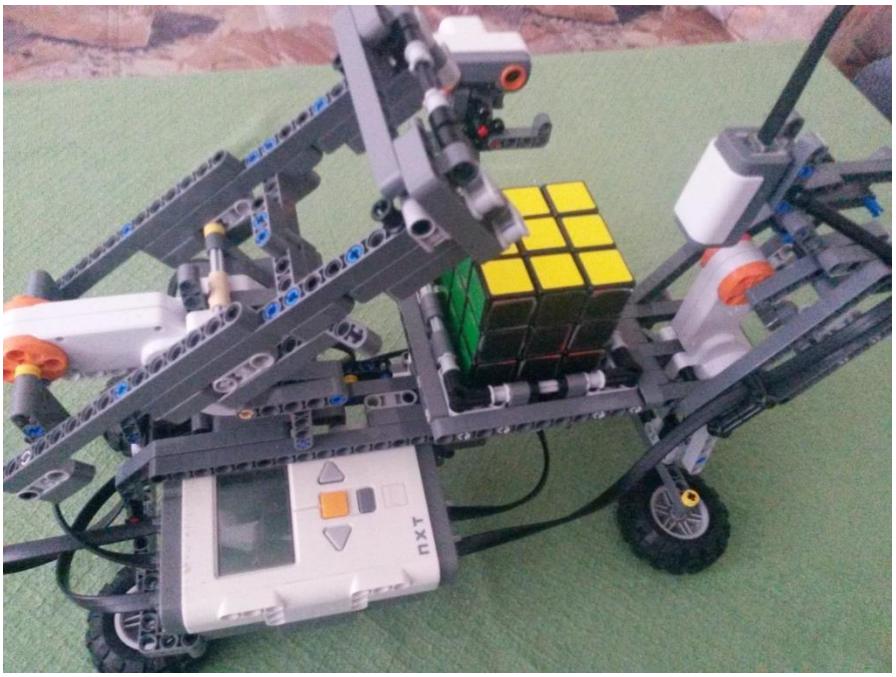
Pro tuto úlohu je také zapotřebí Rubikovy kostky, nelze však použít jakoukoliv kostku. Kostka musí mít správné rozměry tak, aby seděla do konstrukce robota a aby senzor naměřil hodnoty v určitých úhlech natočení.

3.2.2 Cíl úlohy

Cílem této úlohy je postavit a naprogramovat robota, který dokáže vyřešit hlavolam Rubikovy kostky, která bude v kterékoliv rozházené podobě.

3.2.3 Konstrukce robota

Konstrukci robota je možné rozdělit na tři základní části. První částí je nosič kostky, který drží kostku na místě a který otáčí kostkou ve vodorovném směru. Druhou částí je pohyblivé rameno, které má za účel kostku převrátit o 90 °. Poslední částí je rameno, které drží světelný senzor pro snímání kostky. Toto rameno je aktivní pouze na začátku programu, po naskenování kostky se stává nečinným. Pro poslední rameno je zapotřebí delší kabel pro propojení řídící jednotky s motorem, protože se od řídící jednotky nachází v největší vzdálenosti. Na obrázku 27 je ukázána konstrukce robota včetně vložené Rubikovy kostky.



Obrázek 27 – Konstrukce robota pro 2. úlohu

3.2.4 Řešení úlohy

Pro vyřešení hlavolamu Rubikovy kostky existují desítky algoritmů. Mezi nejjednodušší patří skládání kostky vrstvu po vrstvě, s touto metodou začíná většina lidí při skládání kostky. Mezi další nejpoužívanější metody patří:

CFOP (Fridrichova metoda) – tato metoda je pojmenována po Češce Jessice Fridrichové. Základem je vytvoření kříže jedné barvy a poté umisťováním rohu této barvy a jemu příslušné hrany. Tato metoda je vhodná pro mírně pokročilé řešitele, pro metodu je potřeba znát zhruba 78 algoritmů.

Zborowski-Bruchen metoda – obdoba CFOP. Metoda je poměrně těžká na naučení a na zvládnutí, pro tuto metodu je třeba si zapamatovat přes 800 algoritmů.

Ortegova metoda (Nejdřív rohy) – základem této metody je složení všech 8 rohů, poté hrany v jedné vrstvě a pak v protější [26].

V roce 2010 bylo dokázáno, že jakoukoliv kombinaci lze složit do 20 tahů. Tomuto číslu se říká Boží číslo, a je možné toho dosáhnout pomocí tzv. Božího algoritmu. Pouze několik kombinací skutečně potřebuje 20 tahů, většinu kombinací je možné složit s menším počtem tahů. Pro zjištění tohoto čísla byl napsán program, který řešil různé rozložení kostky. Počítač, na kterém program pracoval (čtyř-jádrový procesor Intel Nehalem, 2.8 GHz, více informací o specifikaci počítače není uvedeno), zvládnul vyřešit zhruba jeden milion různých rozložení za vteřinu, prověření všech kombinací pak trvalo 35 procesorových let [27].

Pro řešení této úlohy byl použít světelný senzor, který ale neměří přímo barvu, ale intenzitu odraženého světla. Proto je důležité jednotlivé barvy kostky rozlišit podle intenzity. Tento aspekt zhoršoval rozlišovací vlastnosti senzoru na barvy, jedná se například o rozdíly mezi barvami červená, oranžová a žlutá. Tyto barvy mají pro senzor téměř shodné hodnoty, z toho důvodu je dobré oranžovou barvu na kostce přelepit nebo překreslit jinou barvou. Nejlépe se pro to hodí barva černá, kvůli velice nízké intenzitě odraženého světla nebo naopak barvu hodně světlou až lesklou. Dalším problémem při skenování mohou být popisky na některých kostičkách a z toho důvodu mohou být naměřen rozdílné hodnoty i přesto, že se jedná o jednu barvu. Dalším aspektem při měření je intenzita okolního světla, která dopadá na kostku, tzn. jiné hodnoty budou naměřeny za denního světla a jiní ve tmavém prostředí.

Pro správné přiřazení hodnot intenzity světla k barvám, bylo potřeba provést měření, aby robot věděl, jakým způsobem rozlišovat barvy. Měření probíhalo za různých podmínek, nejdříve bylo provedeno měření za denního světla. Toto měření bylo prováděno světelným senzorem se zapnutým LED světlem a následně s vypnutým. Dále bylo měřeno za horších světelných podmínek, kdy již v místnosti nebylo denní světlo a místnost nebyla osvícena ani jiným zdrojem světla. Hodnoty ze senzoru byly snímány jako procentuální vyjádření intenzity světla tak i v normalizované formě. Normalizovaná forma hodnot je v rozmezí od 0 do 1023, tato hodnota se zde uplatní více, protože jednotlivé barevné hodnoty jsou od sebe více vzdálené a lze tak tyto barvy od sebe lépe odlišit. Výsledky měření demonstruje tabulka číslo 4, první hodnota udává procentuální vyjádření intenzity světla a druhá hodnota je normalizovaná forma hodnot.

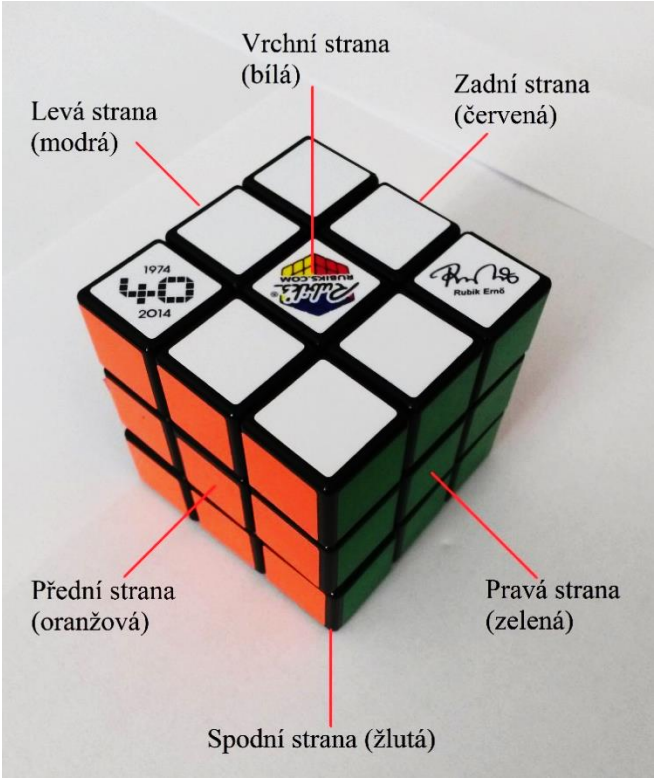
Tabulka 4 – Měřené hodnoty barev s použitím světelného senzoru

Barva	Denní světlo bez LED	Denní světlo s LED	Tmavá místnost bez LED	Tmavá místnost s LED
Bílá	49 %, 517	62 %, 643	15 %, 160	59 %, 615
Zelená	43 %, 446	48 %, 502	14 %, 148	45 %, 477
Modrá	40 %, 420	45 %, 470	14 %, 146	42 %, 440
Žlutá	48 %, 500	60 %, 625	15 %, 150	60 %, 621
Oranžová	45 %, 473	59 %, 617	14 %, 151	59 %, 615
Červená	46 %, 481	58 %, 599	14 %, 152	57 %, 593

Z důvodu horší rozpoznávací funkce barev je do programu přidána možnost manuálního zadávání barev, kdy je uživatel vyzván, aby ručně zadal barvy pro správný výpočet algoritmu, viz příloha B.

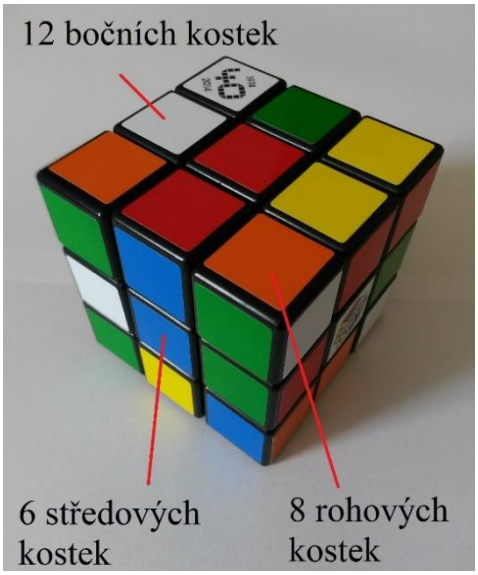
Jak již bylo popsáno výše možností, jak složit Rubikovu kostku je nespočet, robota ale pro skládání omezují jeho možnosti. Pro výpočet některých složitějších algoritmů, není řídící jednotka dostatečně vybavená, co se hardwaru týká. Některé výpočty by trvaly dlouhou dobu (mnohdy i více jak několik hodin) a některé by řídící jednotka nedokázala vypočítat kvůli nedostatku paměti.

U samotného skládání je potřeba uplatnit několik pravidel, které je potřeba dodržovat a na základě těchto pravidel poté kostku skládat. Dodržování pořadí barev je jedním z pravidel, je nutné vědět jaké barvy jsou na protější straně kostky. Bílá barva naproti žluté, oranžová barva naproti červené a modrá barva naproti zelené. Z tohoto pravidla vyplývá že barvy ve vedlejších stranách musí také souhlasit. Pokud je bílá strana nahoře a žlutá dole pak je zelená vpravo od oranžové, červená vpravo od zelené a modrá vpravo od červené. Složená Rubikova kostka s popisem stran je znázorněna na následujícím obrázku.



Obrázek 28 – Složená Rubikova kostka s popiskami jednotlivých stran

Aby mohl robot kostku složit, tedy vypočítat algoritmus pro jeho složení je potřeba kostku rozdělit na menší díly a ty poté postupně skládat v jeden celek. Dodržením tohoto principu je robot schopen vypočítat algoritmy v krátkém čase a nedochází robotovi paměť. Kostky se rozdělí podle jejich umístění na 6 středových kostek, každá taková kostka má pouze jednu barvu. 8 rohových kostek, kde každá kostka má 3 barvy a 12 bočních kostek, každá kostka má 2 barvy, viz obrázek 29.



Obrázek 29 – Rozložení Rubikovy kostky pro výpočet algoritmu složení

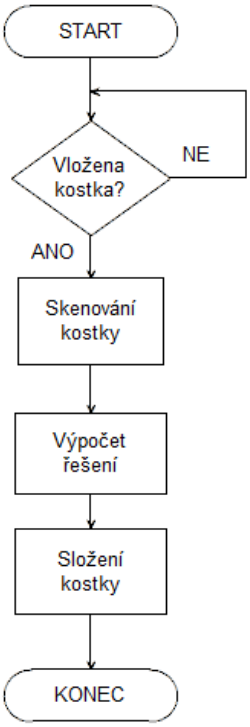
Pro výpočet řešení Rubikovy kostky je použita Ortegova metoda, někdy také známá pod názvem Metoda řešení nejdříve rohy. Metoda rozdělí skládání Rubikovy kostky na několik menších problémů a ty postupně vyřeší. Jak již název napovídá jako první se budou orientovat rohy. Jak již bylo popsáno výše Rubikova kostka obsahuje 8 rohových kostek, a proto se orientace rohů pak dále dělí na vrchní a spodní stranu. Průměrný počet pohybů pro orientaci všech rohů je 12. Dalším krokem je umístit všechny rohy do správné pozice, tzn. aby všechny sousedící barvy byly na správném místě. Průměrný počet pohybů pro tento krok by měl být 8. Po správném umístění všech rohů se začnou umisťovat boční kostky. Nejdříve se vyřeší 3 vrchní kostky a poté 3 spodní kostky, průměrný počet pohybů je 25. Jako poslední, co je třeba vyřešit, jsou hrany uprostřed kostky a poté se vyřeší kostky středové, zde je průměrný počet pohybů 13. Celkový počet pohybů pro vyřešení celé kostky je průměrně 58. V této metodě existuje pár případů, kdy kostku z původní pozice touto metodou nelze složit, v takovém případě je potřeba kostku ještě proházet [28].

Grafické znázornění postupu touto metodou je vidět na obrázku 30.

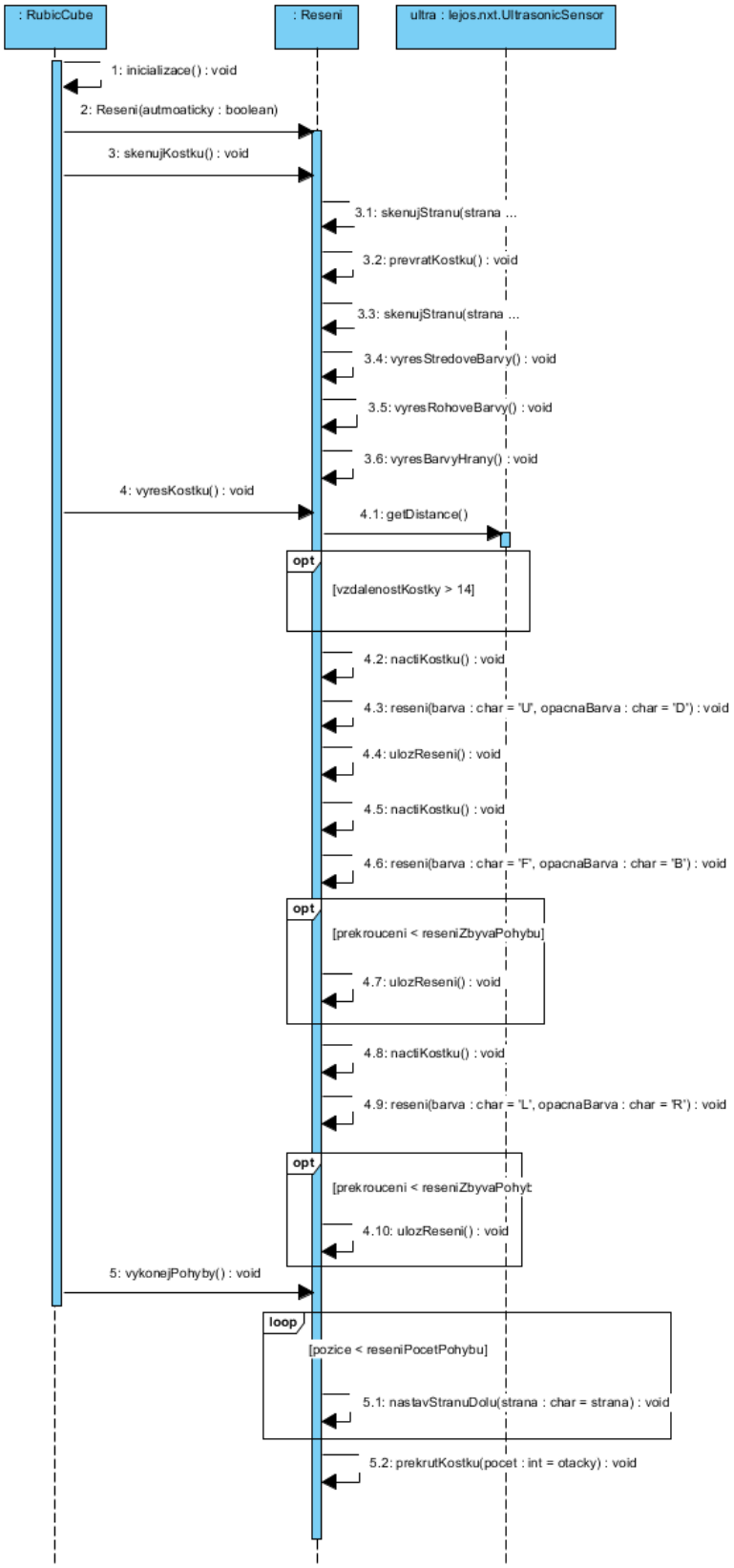
Obrázek 30 – Postup řešení pomocí Ortegovy metody. Zdroj: [28]

Po vložení kostky do robota robot nejdříve oskenuje všechny strany kostky a zapíše si do tabulky její rozložení. Poté robot vypočítá optimální řešení pro složení Rubikovy kostky, podle Ortegovy metody, která je popsaná výše. Výsledkem výpočtu je textový řetězec, který reprezentuje, jakou stranou a jakým směrem se má kostka otočit. Poté robot provede postupně jednotlivé kroky ke složení Rubikovy kostky. Celková doba složení kostky trvá cca 8 minut, doba skládání se liší v závislosti na složitosti rozložené kostky.

- Skenování kostky – cca 2 minutu,
- výpočet řešení – cca 1 minutu,
- složení kostky – cca 5 minuty.



Obrázek 31 – Vývojový diagram pro 2. úlohu



Obrázek 32 – Sekvenční diagram pro druhou úlohu

Na sekvenčním diagramu zobrazeném výše je znázorněn průběh programu pro úlohu číslo 2. Po spuštění je jako první zavolána metoda číslo tři pro oskenování celé kostky. Po naskenování se spustí výpočet řešení pomocí metody číslo 4, zároveň se kontroluje pomocí ultrazvukového senzoru, jestli je kostka stále na místě. Vypočítají se celkem tři různá řešení a v podmínce (opt) se zkontroluje jaké je lepší. Nejlepší řešení je takové, které vyžaduje nejméně pohybů. Po získání řešení je metodou číslo 5 provedeno složení kostky prováděné v cyklu (loop).

3.2.5 Závěr k úloze číslo 2

Skenování barev Rubikovy kostky, se kvůli různým světelným podmínkám někdy špatně načítají, z toho důvodu je možnost zvolit manuální zadávání barev. Robotovi se podařilo Rubikovu kostku složit v relativně rychlém čase. Omezujícím prostředkem na rychlost složení je rychlost motorů a také rychlost výpočtu řešení. Rychlejšího výpočtu řešení by mohlo být dosaženo připojením řídící jednotky NXT k počítači, kde po naskenování Rubikovy kostky by výsledky byly odeslány do počítače a ten by provedl rychlejší výpočet řešení.

Možnosti vylepšení:

- zaměnit světelný senzor za barevný senzor, který již měří konkrétní barvy,
- připojení řídící jednotky k počítači pro rychlejší výpočet.

.

4 ZÁVĚR

V této práci byly řešeny 2 úlohy za použití stavebnice LEGO Mindstorms NXT. I přesto, že v první úloze byl použit rozšiřující senzor, gyroskopický senzor, lze obě úlohy vypracovat i s použitím základní sady stavebnice. Tyto úlohy by mohly sloužit pro výuku robotiky a také programování v programovacím jazyku Java, a to jak na vysokých školách, tak i na středních školách. Při programování byly použity prvky objektově orientovaného programování a také programování s využitím více vláken.

Úloha číslo jedna se zabývala problematikou samo balančního robota. Cílem bylo dosáhnout, aby robot, který stojí pouze na dvou kolech, nespadl a udržel rovnováhu. Dosaženo toho bylo pomocí regulátoru PID a gyroskopického senzoru. Robot zvládl překonat i mírné překážky a dokázal udržet rovnováhu i při mírném nárazu. Součástí robota je i ultrazvukový senzor, který robot pomáhal „vidět“ aby se vyhnul objektům, který jsou přímo před ním. Robota z této úlohy by bylo možné rozšířit dalšími senzory, které by mohli pomoci robotovi při udržování stability, umožňovali robota ovládat na dálku nebo pomohli robotovi zvednout po pádu ze země. Možností, jak tohoto robota vylepšit nebo upravit je skutečně mnoho, jediným omezením je počet portů v řídící jednotce NXT.

Druhou řešenou úlohou bylo vyřešení hlavolamu Rubikovy kostky. Robot měl za úkol naskenovat kostku, poté vypočítat řešení, a nakonec kostku složit. Pro skenování kostky byl v této práci použit světelný senzor, který je ale citlivý na okolní světelné podmínky. Z toho důvodu nebylo možné zajistit při každém skenování stejných hodnot pro jednotlivé barvy na kostce. Proto byla přidána možnost zadat jednotlivé barvy manuálně, aby mohl robot korektně vypočítat řešení. Řešení je vypočítáno pomocí Ortegovy metody pro řešení Rubikovy kostky. Tato metoda byla vybrána, protože nevyžaduje vysoké nároky na paměť a řídící jednotka je ji schopna vypočítat. Tato metoda ale naproti ostatním vyžaduje větší počet pohybů potřebných pro složení kostky. Robota lze upravit nebo vylepšit mnoha způsoby, nejlepší úpravou by byla výměna světelného senzoru za barevný senzor pro lepší rozlišovací schopnosti barev. Robota by také bylo možné připojit k počítači, do kterého by robot odeslal hodnoty naskenované kostky a počítač by provedl výpočet řešení. Došlo by tím k urychlení výpočtu a byl by schopen vypočítat i jiné algoritmy, než který je zmíněn v této práci.

5 POUŽITÁ LITERATURA

[1] History of LEGO Robotics. *LEGO Mindstorms* [online]. [cit. 2017-03-20]. Dostupné z: https://www.lego.com/cs-cz/mindstorms/history

[2] Lego Mindstorms. *Lego Mindstorms NXT* [online]. [cit. 2017-03-20]. Dostupné z: https://en.wikipedia.org/wiki/Lego_Mindstorms

[3] JAKEŠ, Ph.D. Tomáš. Řídící jednotka NXT. *Robotické vzdělávání LEGO Mindstorms* [online]. 2013 [cit. 2017-03-21]. Dostupné z: https://lego.zcu.cz/web/ridici-jednotka

[4] JAKEŠ, Ph.D. Tomáš. Lego Mindstorms NXT Education. *Robotické vzdělávání LEGO Mindstorms* [online]. 2013 [cit. 2017-03-21]. Dostupné z: https://lego.zcu.cz/web/moduly-systemu-lego-nxt

[5] JAKEŠ, Ph.D. Tomáš. Základní modulu systému LEGO MINDSTORMS. *Robotické vzdělávání LEGO Mindstorms* [online]. [cit. 2017-03-21] Dostupné z: https://lego.zcu.cz/web/zakladni-moduly

[6] BAGNALL, Brian. Maximum Lego NXT. Winnipeg: Variant Press, 2007. ISBN 9780973864915.

[7] Lego Mindstorms. *Lego Mindstorms NXT* [online]. [cit. 2017-04-23]. Dostupné z: https://de.wikipedia.org/wiki/Datei:Lego_mindstorms_nxt_normal_sound_sensor.jpg

[8] NXT Gyro Sensor. *HiTechnic Products* [online]. © 2001-2012 [cit. 2017-03-10] Dostupné z: http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NGY1044

[9] NXT Color Sensor V2. *HiTechnic Products* [online]. © 2001-2012 [cit. 2017-04-24]. Dostupné z: http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NCO1038

[10] JAKEŠ, Ph.D. Tomáš. Spojovací kabely LEGO MINDSTORMS. *Robotické vzdělávání LEGO Mindstorms* [online]. 2013 [cit. 2017-03-17] Dostupné z: https://lego.zcu.cz/web/zakladni-moduly/spojovacikabely

[11] LEGO Ke stažení. *LEGO Group* [online]. © 2017 [cit. 2017-03-10] Dostupné z: https://www.lego.com/cs-cz/mindstorms/downloads

[12] Introduction. *leJOS Java for LEGO Mindstorms* [online]. [cit. 2017-03-13] Dostupné z: http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/Intro.htm

[13] LEGO Nauč se programovat. *LEGO Group* [online]. © 2017 [cit. 2017-03-10] Dostupné z: https://www.lego.com/cs-cz/mindstorms/learn-to-program

[14] Error Handling and Debugging. *leJOS Java for LEGO Mindstorms* [online]. [cit 2017-04-28]. Dostupné z: http://www.lejos.org/nxt/nxj/tutorial/ErrorHandlingAndDebugging/ErrorHandling_and_debugging.htm

[15] leJOS Getting Started on Microsoft Windows. *leJOS Java for Lego Mindstorms* [online]. [cit. 2017-03-13] Dostupné z: http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/GettingStartedWindows.htm

[16] leJOS Using a Java IDE with leJOS NXJ. *leJOS Java for Lego Mindstorms* [online]. [cit. 2017-03-27] Dostupné z: http://www.lejos.org/nxt/nxj/tutorial/Preliminaries/UsingIDEs.htm

[17] ŠAFÁŘOVÁ, Soňa. Stabilizace a řízení inverzního kyvadla. Pardubice 2013, Diplomová práce. Univerzita Pardubice, Fakulta Elektrotechniky a Informatiky.

[18] Jak funguje Segway? *Segway Česká republika* [online]. © 2014 [cit. 2017-03-27] Dostupné z: http://www.segwaypt.cz/segway-funkce

[19] Segway i2 SE. *Segway Česká republika* [online]. [cit. 2017-04-23]. Dostupné z: http://www.segwaypt.cz/modelova_rada/Segway-i2-SE

[20] Základy teorie ON/OFF a PID regulace. *Dixell.cz Elektronické měřicí a regulační přístroje Dixell* [online]. [cit. 2017-03-28] Dostupné z: https://www.dixell.cz/teorie-regulace/

[21] Stavíme kvadrokoptéru: PID regulátor. *Root.cz – informace nejen ze světa Linuxu* [online]. 2015 [cit. 2017-04-23]. Dostupné z: https://www.root.cz/clanky/stavime-kvadrokopteru-pid-regulator/

[22] NOVÁK, Vilém. *Základy fuzzy modelování*. Praha: BEN – technická literatura, 2000. ISBN 80-7300-009-1.

[23] Fuzzy neuronové řídicí systémy – vysvětlení. *Control Engineering Česko* [online]. 2007 [cit. 2017-04-23] Dostupné z: http://www.controlengcesko.com/hlavni-menu/artykuly/artykul/article/fuzzy-neuronove-ridici-systemy-vysvetleni

[24] HTWay A Segway type robot. *HiTechnic Products* [online]. 2010 [cit. 2017-03-27] Dostupné z: http://www.hitechnic.com/blog/gyro-sensor/htway/

[25] History of the Rubik’s Cube puzzle toy. Rubik’s Cube and Twisty Puzzle Portal – Ruwix [online]. [cit. 2017-04-01] Dostupné z: https://ruwix.com/the-rubiks-cube/history-rubiks-cube/

[26] Metody skládání Rubikovy kostky. *Hlavolamy.info* [online]. 2014 [cit. 2017-04-01] Dostupné z: http://mojehlavolamy.webnode.cz/news/metody-skladani-rubikovy-kostky/

[27] God's Number is 20. *God's Number is 20* [online]. [cit. 2017-04-01] Dostupné z: http://www.cube20.org/

[28] ORTEGA, Victor a Josef JELINEK. Ortega Corners-First Solution Method for Rubik's Cube. *Rubik's Cube Solution Methods* [online]. [cit. 2017-04-23]. Dostupné z: http://rubikscube.info/ortega.php

[29] SCHILDT, Herbert. Java 8: výukový kurs. Přeložil Jakub GONER. Brno: Computer Press, 2016. ISBN 978-80-251-4665-1.

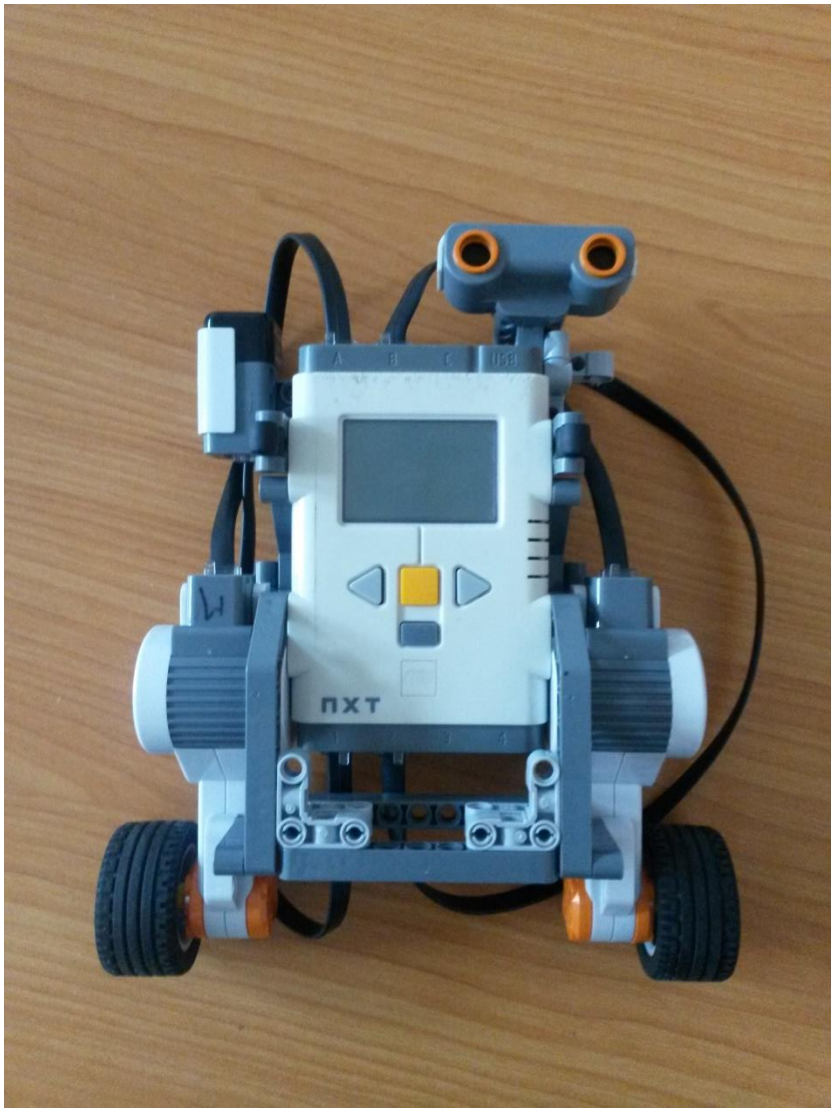
6 PŘÍLOHY

Příloha A – *Uživatelská příručka k ovládání robota k 1. úloze*

Před zapnutím programu je potřeba zkontrolovat. aby motory a senzory byly zapojeny ve správných portech. Při čelním pohledu na robotu musí být levý motor zapojen v portu A a pravý motor v portu B. Gyroskopický senzor musí být umístěn v portu 1 a ultrazvukový senzor v portu 2. Gyroskopický senzor musí být připevněn na levé straně robota, a nápis, který označuje, že se jedná o gyroskopický senzor, musí směřovat směrem k robotovi. Před spuštěním je potřeba zkontrolovat stav baterie, pokud by měl robot nedostatečně nabytou baterii, nedokázal by udržovat stabilitu.

Po zapnutí programu, je uživatel vyzván k zadání průměru kol, které jsou na robotovi použity. Tento údaj je důležitý, aby robot správně balancoval, velikost kol lze zjistit například z boční strany pneumatik. Pokud se zde údaj nenachází je možné údaj zjistit z internetových stránek http://wheels.sariel.pl/ anebo je potřeba kola přeměřit. Zadávané hodnoty do programu jsou uváděny v centimetrech a na výběr je ze 3 základních velikostí, které jsou dodávané společně se stavebnicí LEGO Mindstorms NXT. Výběr v menu se provádí stisknutím levé nebo pravé šipky a potvrzení stisknutím oranžového tlačítka.

Po potvrzení velikosti kol, je uživatel vyzván, aby položil robota na zem do vodorovné polohy, tak aby řídící jednotka ležela monitorem směrem nahoru. Je důležité, aby byl robot co nejvíce ve vodorovné poloze, protože po stisknutí oranžového tlačítka dojde ke kalibraci gyroskopického senzoru. Během kalibrace by se nemělo s robotem hýbat, kalibrace trvá zhruba 4 vteřiny. Po kalibraci bude na displeji probíhat odpočet 5 sekund, během kterých je potřeba robota zvednout do svislé polohy na kola, po odpočtu začne robot sám balancovat. Při přidržování robota ve svislé poloze nesmí být robot pevně držen, pouze přidržován, aby se po zahájení balancování robot mohl začít pohybovat. Robot poté bude chvíli balancovat na místě, aby získal co největší stabilitu. Po krátké době se začne pohybovat směrem dopředu, dokud nenarazí na překážku, která od něj bude vzdálená zhruba 15 cm. Po zaregistrování překážky robot začne couvat a otočí se o 90 °. Tento cyklus se stále opakuje, dokud robot nespadne. Po pádu robota se zastaví veškeré jeho činnosti.

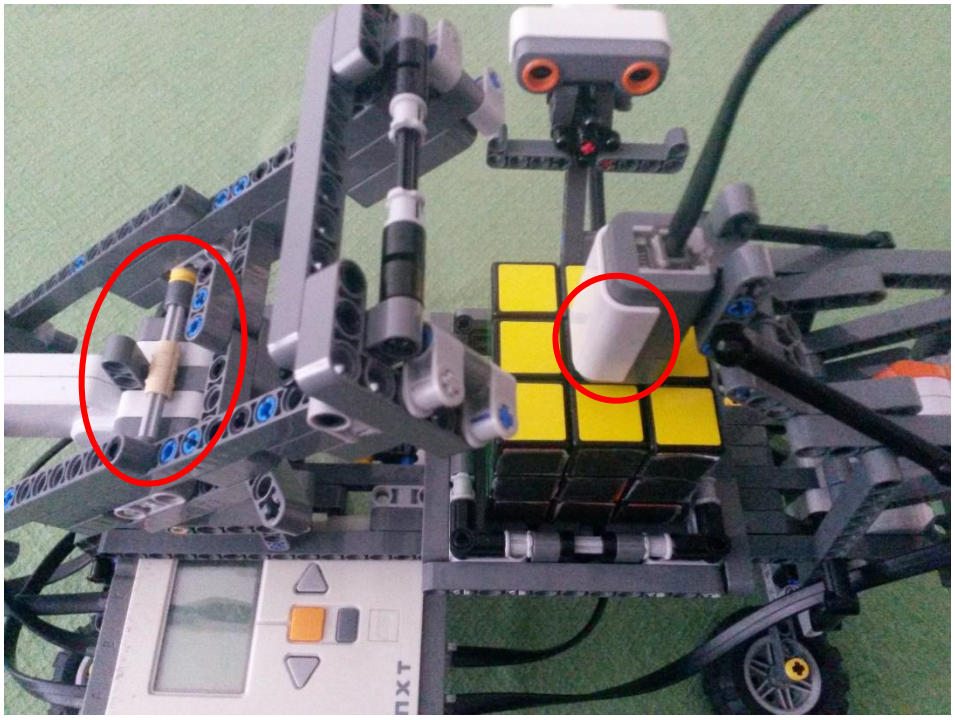


Obrázek 33 – Poloha pro kalibraci robota

Příloha B – *Uživatelská příručka k ovládání robota k 2. úloze*

Nejdříve je potřeba zkontrolovat správné zapojení motorů a senzorů do portů. Motor se světelným senzorem je potřeba zapojit do portu A. V portu B se nachází motor, který převrací kostku a motor, který drží Rubikovu kostku je potřeba zapojit do portu C. Světelný senzor se zapojí do portu 1 a ultrazvukový senzor je zapojen v portu 2. Robot se po celou dobu provádění programu nehýbe, a proto je možné, aby byl po celou dobu připojen k napájecímu adaptéru.

Před zapnutím programu se umístí Rubikova kostka na podložku, přičemž nezáleží, jakou stranou se kostka umístí. Po vložení je potřeba nastavit rameno se světelným senzorem do středu kostky, tak aby světelný senzor mířil na středovou kostku vrchní strany. Rameno na překlápění kostky musí být v nakloněné poloze a opřené o LEGO kostku nad motorem, viz obrázek 34.



Obrázek 34 – Robot ve výchozí poloze před skenováním

Po spuštění programu je uživatel vyzván, aby vybral způsob zadávání barev. Na výběr je automatické zadávání barev, kdy veškeré barvy z Rubikovy kostky jsou naskenovány pomocí světelného senzoru. Manuální zadávání, znamená, že uživatel veškeré barvy zadává sám, tzn. světelný senzor se zastaví nad vybranou kostkou a uživatel pro danou kostku vybere barvu. Barvy jsou vybírány pomocí tlačítek (levé a pravé tlačítko) na řídící jednotce, pro výběr barvy uživatel stiskne ENTER. Manuální zadávání se doporučuje za zhoršených světelných

podmínek, nebo pokud jsou barvy na Rubikově kostce popsané, pomalované, nebo jinak poškozené. Dále je důležité, aby barvy na Rubikově kostce nebyly podobné, světelný senzor by pak tyto barvy od sebe nerozlišil.

Pokud v průběhu zadávání barev, v manuálním režimu, dojde k zadání špatné barvy, uživatel stiskne tlačítko ESCAPE (malé obdélníkové tlačítko, pod oranžovým tlačítkem). Na displeji se zobrazí zpráva o přerušení skenování a po stisknutí klávesy ENTER se kostka vrátí do původní polohy. Světelný senzor se posune na středovou kostku a poté uživatel znovu zadá barvy pro stranu kostky.

Po naskenování celé kostky, začne robot vypočítávat řešení, jak kostku složit. Výpočet trvá v závislosti na míře rozložení kostky, průměrně trvá robotovi výpočet do 1 minuty. Během výpočtu je možné na displeji sledovat stav a výsledky řešení. Když robot dokončí výpočet, je uživatel vyzván, aby stiskl ENTER. Poté již začne robot kostku skládat, na displeji je možné sledovat počet zbývajících pohybů, než bude kostka složená. Počet se zmenší o jednu pokaždé když robot chytí kostku a otočí její spodní stranou.