UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Stabilizace a řízení inverzního kyvadla

Bc. Soňa Šafářová

Diplomová práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Soňa Šafářová**
Osobní číslo:	**I11413**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Stabilizace a řízení inverzního kyvadla**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Cílem diplomové práce je sestavit ze stavebnice Lego Mindstorms Education robota ve tvaru inverzního kyvadla. Kromě stabilizace robota pomocí fuzzy řízení je třeba vyřešit pohyb robota v prostoru (minimálně pokyny vpřed, vzad, otoč vlevo, otoč vpravo). Komunikace mezi robotem a operátorskou stanicí bude probíhat pomocí bluetooth. Rešeršní část práce bude obsahovat rozbor problému, stručný popis možných přístupů k řešení a především detailní rozbor způsobu řešení pomocí fuzzy řízení. Praktická část práce sestává z návrhu originálního robotu, jeho oživení, zajištění komunikace mezi robotem a operátorem a zejména implementace algoritmu řízení robotu. Operační systém robotu bude Lejos, veškerý programový kód bude psán v Javě, popř. její modifikaci NXJ. Práce bude obsahovat přehlednou uživatelskou příručku pro práci se všemi vytvořenými aplikacemi. Práce bude vypracována podle interních pokynů Fakulty elektrotechniky a informatiky v souladu s normami ČSN ISO 7144 a ČSN ISO 690.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

NGUYEN, H., PRASAD, N., WALKER, C. A First Course in Fuzzy and Neural Control. Chapman and Hall, 2003.
PASSINO, K M., YURKOVICH, S. Fuzzy Control, Addison-Wesley, 1998.

Lejos. [online]. [cit. 2012-09-21]. Dostupné z: http://lejos.sourceforge.net/.

Vedoucí diplomové práce: **Ing. Petr Doležel, Ph.D.**
Katedra řízení procesů

Datum zadání diplomové práce: **31. října 2012**
Termín odevzdání diplomové práce: **17. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2012

Prohlášení autora

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

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

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 17. 05. 2013 Soňa Šafářová

Poděkování

Děkuji Ing. Petru Doleželovi Ph.D. za jeho rady, připomínky, čas a konzultace, které přispěly k tvorbě diplomové práce. Velké poděkování patří i mé rodině za nepřetržitou podporu a trpělivost po celou dobu studia.

Diplomová práce vznikla v rámci řešení projektu „Podpora stáží a odborných aktivit při inovaci oblasti terciárního vzdělávání na DFJP a FEI Univerzity Pardubice, reg. č.: CZ.1.07/2.4.00/17.0107“, v týmu Moderní metody řízení – vývoj a aplikace metod prediktivního řízení s využitím umělé inteligence.

Anotace

Diplomová práce se zabývá stabilizací inverzního kyvadla, jehož fyzická podoba je realizována jako robot sestavený ze stavebnice LEGO Mindstorms Education, který balancuje na dvou motory poháněných kolech. Pro stabilizaci je použit fuzzy regulátor aplikující principy fuzzy logiky, který na základě vyhodnocení aktuálního stavu generuje akční signál.

V úvodu diplomové práce je rozebrán samotný problém inverzního kyvadla, dále pak možné způsoby měření potřebných vstupních hodnot a způsoby stabilizace. Druhá část práce se podrobně věnuje principu stabilizace robota fuzzy regulátorem, od struktur lingvistických proměnných až po samotný řídicí algoritmus. Následuje popis praktické části od nastavení proměnných prostředí počítače a vývojového prostředí až po ovládání vytvořených programových částí. V závěru práce jsou zhodnoceny výsledky fuzzy řízení.

Klíčová slova

Inverzní kyvadlo, fuzzy, Java, leJOS NXJ, LEGO Mindstorms Education, NXT

Title

Stabilization and control of inverted pendulum

Annotation

This thesis deals with stabilisation of inverted pendulum, physical appearance of which is realised as a robot built from LEGO Mindstorms Education and which balances on wheels powered by two motors. Fuzzy regulator applying principles of fuzzy logic, which generates action signal based on evaluation of current state, is used for stabilisation.

In the introduction of the thesis the problematic of inverted pendulum, possible methods of measuring of necessary input data and a way of stabilization are analysed. Next part deals in detail with principle of stabilization of the robot by fuzzy regulator, from structures of linguistic variables to the control algorithm itself. A description of the practical part follows from setting the computer environment variables and developing environment to controlling generated program part. In the conclusion of the thesis, the results of fuzzy control are evaluated.

Keywords

Inverted pendulum, fuzzy, Java, leJOS NXJ, LEGO Mindstorms Education, NXT

Obsah

Seznam zkratek

SISO	Single-input, single-output
MISO	Multi-input, single-output
MIMO	Multi-input, multi-output
COA	Center of Area
FoM	First of Maxima
MoM	Middle of Maxima
JVM	Java Virtual Machine
API	Application Programming Interface
IDE	Integrated Development Environment
USB	Universal Serial Bus
PC	Personal Computer

Seznam obrázků

Seznam tabulek

Úvod

S pojmem fuzzy logika se lidstvo setkává zhruba od poloviny 20. století, kdy byl tento obor odvozen od teorie množin. Vzhledem k tomu, že v tomto vědním oboru nejsme omezeni pouze logickými hodnotami „pravda“, „nepravda“, můžeme využít a dále pak pracovat i s pojmy jako „málo“, „trochu“, „hodně“. Z uvedených pojmů je zřejmé, že se ve fuzzy logice pracuje s mírou neurčitosti. Ne vždy je možné rozhodnout nebo zařadit určitou hodnotu nebo chování jednoznačně do jedné kategorie. Této skutečnosti se v dnešní době využívá u řady zařízení, která pracují se subjektivním pohledem jejich uživatelů na danou hodnotu. Může se jednat o běžné elektrospotřebiče, jako jsou chladničky, myčky, pračky nebo třeba o regulaci teploty v místnosti, kdy je fuzzy logikou řízeno otevírání a zavírání hlavic ventilů radiátorů podle toho, jestli je v místnosti teplo, zima nebo je teplota vzduchu akorát.

Vzhledem k tomu, že se ve fuzzy logice pracuje s neurčitostí, můžeme se na jednu hodnotu vstupu či výstupu dívat z více pohledů. Tyto vlastnosti jsou určující pro použití fuzzy řízení jako vhodného řešení problematiky stabilizace inverzního kyvadla.

1 Cíl práce

Řízení stabilizace inverzního kyvadla se obvykle řeší PID regulátorem, proto je cílem diplomové práce detailně popsat a prakticky vyzkoušet jinou, méně užívanou technologii a to konkrétně fuzzy řízení, které spadá do kategorie umělé inteligence.

Úkolem teoretické části diplomové práce je popsat a seznámit se s problémem inverzního kyvadla, způsoby měření potřebných vstupních hodnot a metodami využívajícími se pro jeho stabilizaci. Stěžejní část je pak detailní popis fuzzy řízení, které bylo využito v praktickém příkladu.

V praktické části diplomové práce je úkolem navrhnout a sestavit robota ze stavebnice LEGO Mindstorms Education, který svojí konstrukcí představuje inverzní kyvadlo. Kromě stabilizace robota fuzzy regulátorem je zapotřebí vyřešit pohyb robota v prostoru a schopnost komunikovat s operátorskou stanicí prostřednictvím technologie bluetooth.

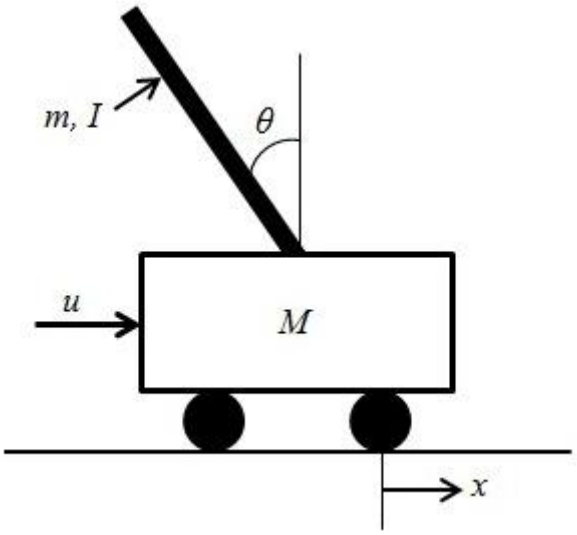
Součástí diplomové práce je uživatelská příručka, která obsahuje pokyny pro práci s vytvořenými programy.

2 Problém inverzního kyvadla

Problém inverzního kyvadla je jeden ze základních příkladů z dynamiky a z teorie řízení a slouží jako testovací prostředek pro různé druhy řízení. Klasické kyvadlo je například u hodin, kde kyvadlo visí dolů z otočného, ale svojí pozicí stabilního bodu. Těžiště kyvadla je pod otočným bodem, zatímco těžiště inverzního kyvadla se nachází nad otočným bodem, čímž se inverzní kyvadlo stává nestabilním.

Obvyklý model inverzního kyvadla se skládá z pohyblivé podložky, označované jako vozík a z tyče, která je přichycena v otočném kloubu umístěném na horní části podložky. Tyč tak svojí polohou tvoří naklánějící se kyvadlo. Úkolem vozíku je přímým horizontálním pohybem udržet tyč ve výchozí vertikální poloze. Bez pohybu vozíku (regulačního zásahu) není možné docílit stabilizace kyvadla, jelikož stabilitu kyvadla ovlivňují i malé vnější síly, které na vozík působí. Model inverzního kyvadla můžeme vidět na obr. 2.1 a praktický příklad z běžného života na obr. 2.2.

Pro úspěšné řešení stabilizace inverzního kyvadla je důležité vytvořit řídicí systém, který bude schopný řídit jak úhel natočení kyvadla, tak pozici vozíku.



Obr. 2.1 – Model inverzního kyvadla

Parametry modelu:

- M – hmotnost vozíku,
- m – hmotnost kyvadla,
- I – setrvačnost kyvadla,
- u – síla aplikovaná na vozík,
- x – pozice vozíku,
- θ – úhel, o který se kyvadlo vychýlilo.

Podrobný matematický model inverzního kyvadla je k dohledání v různých literárních zdrojích, např. ve zdroji [1].

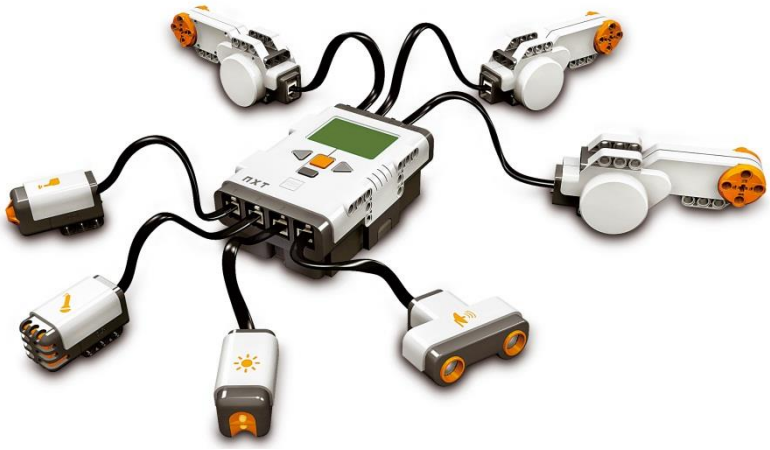
Obr. 2.2 – Praktický příklad inverzního kyvadla

2.1 Stavebnice LEGO Mindstorms Education

LEGO Mindstorms Education je stavebnice obsahující jak hardware, tak software pro vytváření programovatelných a uživatelsky přizpůsobivých robotů. Jak z názvu vyplývá, stavebnice je produktem vyvinutým společností LEGO.

Mozkem stavebnice je programovatelná kostka NXT, ke které je možné připojit až tři servomotory a čtyři senzory. V základní nabídce je k dispozici světelný, ultrazvukový, dotykový a zvukový senzor. Na trhu jsou také další typy senzorů, které jsou se stavebnicí plně kompatibilní (gyroskop, akcelerometr, barometr, atd.). Ke stavebnici je dodáván grafický programovací software NXT-G, který umožňuje intuitivně vytvářet jednoduché,

ale i složitější úlohy. Ovládání programu je založeno na operacích drag and drop a jednotlivé úlohy jsou sestaveny z navzájem propojených bloků, které obsahují nastavitelné parametry a vlastnosti. Velkou výhodou stavebnice je, že programování úloh, především těch složitých, může být realizováno i v programovacím jazyku Java nebo C.



Obr. 2.3 – Hardwarové vybavení stavebnice LEGO Mindstorms

První verze stavebnice byla vyvinuta ve spolupráci s laboratoří MIT a na trh byla uvedena v roce 1998 pod názvem Robotic Invention System. LEGO Mindstorms NXT byla druhou generací stavebnice, byla vydána v roce 2006 a oproti první verzi umožňovala připojení prostřednictvím technologie bluetooth. Následně v roce 2009 byla vydána druhá verze s označením LEGO Mindstorms NXT 2.0. Inovací oproti starší verzi byl nový barevný senzor a podpora operací s plovoucí desetinnou čárkou. Třetí generace stavebnice je LEGO Mindstorms EV3, která bude dostupná od podzimu 2013 a bude umožňovat připojení i prostřednictvím Wi-Fi. Více se o stavebnici můžete dozvědět z webových stránek LEGO [2].

LEGO Mindstorms Education je vzdělávací verze stavebnice, zahrnující různé, dle věku uživatelů rozlišené, sestavy. Tyto typy stavebnice se využívají pro rozšíření a podporu výuky robotiky na základních, středních i vysokých školách. Stavebnice je vhodným prostředkem k simulování reálných systémů, ale může přispívat i k rozvoji kreativity.

Na následujících obrázcích můžeme vidět příklady robotů sestavených ze stavebnice LEGO Mindstorms. Na obr. 2.4 je chodící aligátor, který zavírá a otvírá tlamu při detekci objektu v její blízkosti. Obr. 2.5 zobrazuje model robota střílejícího kuličky. A na obr. 2.6 můžeme vidět robota, schopného kráčet po dvou končetinách. Inspiraci je možné hledat na CD přiloženém ke stavebnici nebo na webových stránkách LEGO Mindstorms [3].

Obr. 2.4 – Robogátor

Obr. 2.5 – Shooterbot

Obr. 2.6 – Alpha-Rex (Humanoid)

2.2 Způsoby měření vstupních hodnot

Abychom mohli regulovat stabilitu inverzního kyvadla, je v první řadě zapotřebí získat vstupní parametry pro řídicí algoritmus. Představme si tedy způsoby jejich měření.

2.2.1 Světelný senzor

Světelný senzor je jeden ze dvou základních senzorů, které pomáhají robotovi vidět. Druhým z této skupiny je ultrazvukový senzor. Světelný senzor umožňuje robotu zaznamenávat změny intenzity světla, případně různé barvy světelného záření [4]. Ultrazvukový senzor umožňuje detekovat objekty a pohyb, ale i měřit vzdálenost od objektů v centimetrech nebo palcích až do vzdálenosti 255 centimetrů. Přesnost měření se pohybuje v rozmezí +/- 3 cm, což je velmi nedostačující pro potřeby stabilizace inverzního kyvadla a proto není možné použít ultrazvukový senzor pro měření vstupních hodnot [5].

Obr. 2.7 – Ultrazvukový senzor

Světelný snímač se skládá ze dvou hlavních částí. Z vlastního zdroje světla, jenž je v podobě IR LED diody a z fototranzistoru. LED diodu lze softwarově vypnout nebo zapnout, podle potřeby použití aktivního či pasivního módu. Senzor může měřit buď odraz světla od povrchu, generovaného LED diodou (aktivní mód) nebo jen intenzitu světla z okolí (pasivní mód). Naměřené hodnoty jsou reprezentovány v rozmezí 0 – 100 %.

Obr. 2.8 – Světelný senzor

Senzor tedy umožňuje robotovi reagovat na intenzitu odraženého světla od různě barevných povrchů, anebo vnímat míru intenzity světla v jeho okolí. Pokud je světelný snímač kalibrovaný, lze ho použít i jako senzor vzdálenosti a tedy určit, zdali se robot naklání dopředu nebo dozadu.

2.2.2 Gyroskop

Gyroskopický senzor je zařízení, které dokáže určit rotaci objektu v prostoru. Gyro senzor obsahuje jednoosý gyroskopický snímač, který detekuje rotaci a vrací hodnotu reprezentující počet stupňů za sekundu rotace a v jakém směru byla rotace provedena. Naměřená hodnota rotace se může pohybovat v rozmezí +/- 360° za sekundu. Měřená veličina se nazývá úhlová rychlost.

Gyroskop umožňuje konstruovat zařízení, která mohou balancovat, houpat se nebo vykonávat jiné činnosti, při kterých je měření rotace důležité.

Obr. 2.9 – Gyroskopický senzor

Osa měření je ve svislé rovině a gyroskop je potřeba umístit černou části nahoru (viz obr.2.9) [6]. Tento senzor není součástí stavebnice LEGO Minstorms Education.

2.2.3 Motory

Pro získání informací o úhlu a rychlosti natočení robota lze využít i hnací motory, ačkoliv se v první řadě používají k vykonávání regulačního zásahu. V programovacím jazyku leJOS, který bude přiblížen v jedné z dalších kapitol, jsou dostupné metody jak pro určení hodnoty úhlu, o který se motory otočily, tak pro získání hodnot úhlové rychlosti motorů. Hodnoty veličin se uvádějí ve stupních a ve stupních za sekundu.

2.3 Způsoby řízení stabilizace

Po úspěšném změření stavových veličin se můžeme věnovat řízení kyvadla. Zde také existuje několik způsobů, jak k problému inverzního kyvadla přistupovat.

2.3.1 PID regulátor

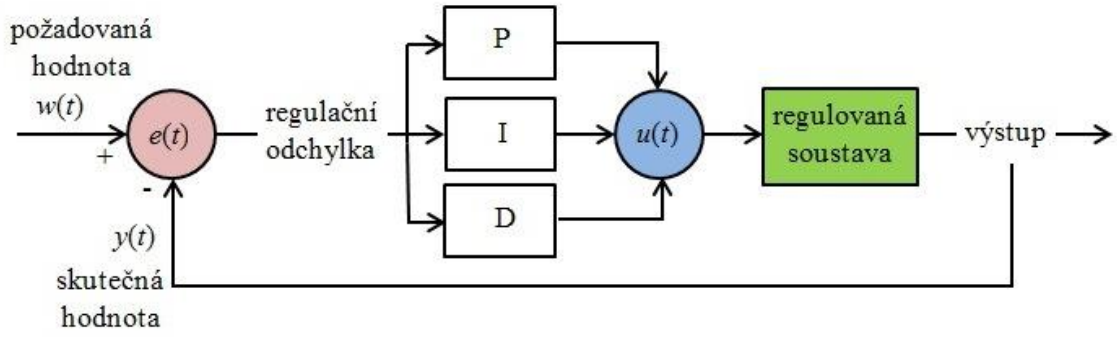
PID regulátor patří mezi základní regulátory, které se využívají k ovládání široké škály strojních zařízení, vozidel nebo robotů. Regulátory se konstruují, aby řízení procesu bylo částečně automatické a nebylo potřeba nepřetržitého dohledu a zásahu operátora.

Písmena v názvu představují proporcionální, integrační a derivační složku chování regulátoru. Regulátor automaticky nastavuje akční veličinu $u(t)$ tak, aby regulovaná

veličina $y(t)$ nabývala požadované hodnoty $w(t)$, kde t je čas. Jednotlivé složky regulátoru jsou tedy konstanty, jejichž výsledné hodnoty interpretují z časového hlediska závislost na:

- P – aktuální chybě,
- I – akumulaci předchozích chyb,
- D – predikci budoucích chyb.

Regulátor spolu s regulovanou soustavou tvoří regulační smyčku. Na vstup regulátoru je přivedena požadovaná hodnota $w(t)$ a spolu s ní i skutečná hodnota regulované veličiny $y(t)$ (výstup z regulované soustavy). Výstup regulátoru $u(t)$, který je akční veličinou, působí, po případné transformaci, na vstup do soustavy.



Obr. 2.10 – Schéma PID regulátoru

Cílem využití regulátoru je minimalizovat regulační odchylku a tím stabilizovat regulovanou soustavu. Regulační odchylka je definována jako rozdíl mezi žádanou a skutečnou (naměřenou) hodnotou regulované veličiny viz rovnice (2-1).

$$e(t) = w(t) - y(t) \tag{2-1}$$

Nenulová hodnota $e(t)$ může být způsobena poruchou v regulované soustavě nebo cíleně změnou požadované hodnoty. V závislosti na regulační odchylce se pak regulátorem upravuje akční veličina. Výsledná hodnota akční veličiny je tvořena poměrným součtem jednotlivých složek P, I a D [7].

Při tvorbě regulátoru není nutné využít všechny složky chování a tím vznikají další typy regulátorů, např. P, PI nebo PD regulátor.

2.3.2 Lineární kvadratický regulátor

Lineární kvadratický regulátor je v literatuře označován zkratkou LQR. Tento typ regulátoru se používá k nalezení optimální zpětné vazby ve stavovém prostoru. Máme nějaký aktuální stav regulovaného systému a chceme přejít do výchozího stavu. Ve výpočtech se používají konstantní váhové matice **Q** a **R**, kde matice **Q** obsahuje váhové faktory zpětné vazby a matice **R** je váhovým skalárním faktorem. Pro matice musí platit, že matice **Q** je pozitivně semidefinitní a matice **R** je pozitivně definitní.

Regulátor je definován rovnicí

$$u(t) = \mathbf{K}x(t) \tag{2-2}$$

kde K je maticí konstant, kterou lze získat řešením Riccatiho rovnice (2-3) obsahující matice **Q** a **R**.

$$\frac{dy}{dx} = \mathbf{P}(x) + \mathbf{Q}(x)\,y + \mathbf{R}(x)\,y^2 \tag{2-3}$$

2.3.3 Fuzzy řízení

Fuzzy řízení je postaveno na fuzzy logice, která je v dnešní době často aplikována v řízení a regulaci. I když se v této vědní disciplíně pracuje s neurčitostí, zůstává fuzzy logika přesnou matematickou disciplínou využívající se k modelování složitých procesů.

Ve fuzzy řízení se setkáváme s pojmy jako lingvistická proměnná, fuzzy množina, pravidlo nebo stupeň příslušnosti. Vše vyjmenované je potřebné k vytvoření regulátoru, avšak pravidla typu JESTLIŽE-PAK jsou stavebním kamenem fuzzy řízení. Po jejich aplikaci na vstupní veličiny a po provedení několika dalších operací dostaneme na výstupu regulátoru akční veličinu.

Podrobně se budeme fuzzy řízení věnovat v následující kapitole.

3 Fuzzy řízení

Fuzzy je slovo pocházející z angličtiny, které v překladu znamená *mlhavý*, *matný nejasný*, *neostrý*, *neurčitý*, *vágní*. Fuzzy řízení využívající se pro regulaci systémů je postaveno na principech a vlastnostech fuzzy logiky, kterou si dále podrobně popíšeme.

Pojem fuzzy logika byl zaveden v roce 1965 profesorem kalifornské univerzity v Berkeley Lotfi A. Zadehem. Vychází z teorie fuzzy množin, která představuje zobecnění klasických množin, kde zkoumaný prvek buď do množiny patří, nebo nepatří. Při využití fuzzy logiky můžeme definovat mimo krajních hodnot z intervalu [0,1] i další stupně příslušnosti prvku k množině.

Řízení systému klasickou regulací je možné, pokud máme k dispozici matematický popis procesu. Avšak nalezení matematického popisu v praxi může být velice obtížné nebo je popis velmi složitý a tudíž je návrh klasického regulátoru ať už z časového či finančního hlediska téměř nemožný. Tento problém se obvykle řeší použitím přibližné metody nebo se provádějí různá zjednodušení a výsledná regulace pak nemusí být spolehlivá. Častým regulačním prvkem u těchto typů procesů bývá operátor (člověk), který na základě svých zkušeností zajišťuje řízení procesu. K tomu mu obvykle stačí přibližná znalost chování systému, který obsluhuje, a znalost matematického modelu není potřeba. Zde vzniká možnost využití fuzzy regulátoru, jelikož jeho princip je postaven právě na vzniku situace (chování procesu) a k ní vykonávané činnosti (zásahu operátora). Vztahy mezi situacemi a k nim aplikovanými činnostmi mohou být definovány fuzzy pravidly, která umožňují celý proces automatizovat.

Při použití klasické regulace se setkáváme se situací, kdy definovaný problém nelze jednoznačně (přesně) určit, a proto je dále rozkládán na menší podproblémy, pro jejichž definici je možné opět použít jen dvouprvkové množiny {0,1}. V případech, kdy již není možné problém dále dělit, dochází ke vzniku chyb, které tvoří odchylky procesu od reality. Zde nastává příležitost pro fuzzy řízení, které je schopné zajistit funkčnost systému v míře odpovídající realitě, i když na úkor menších nepřesností vznikajících při definici problému. V důsledku těchto možných řešení regulace nám vzniká zásadní otázka, zdali při popisu procesu zachovat přesnost informace, která bude méně relevantní (klasická regulace) nebo jestli upřednostníme relevantní informaci, která bude méně přesná (fuzzy řízení). Vztah mezi relevancí a přesností informace byl definován L. A. Zadehem v roce 1973 a nazývá se *principem inkompatibility* [8]. Je však důležité poznamenat, že existují procesy, které nelze fuzzy řízením regulovat a fuzzy regulace se tak stává pouze doplňkem klasické regulace.

V následujícím textu se seznámíme s jednotlivými částmi a principy fuzzy regulátoru, který jak, již bylo řečeno, byl využit v praktické části diplomové práce.

3.1 Teorie množin

Než se začneme podrobně věnovat fuzzy množinám, zopakujeme si základní skutečnosti z teorie množin. Klasická množina je soubor prvků libovolného druhu. Ke každé množině existuje množina jejích podmnožin, zatímco množina všech množin neexistuje. Proto je při práci s množinami definována jedna pevně daná množina – ***univerzální množina U*** (univerzum). Druhou velmi důležitou množinou je ***prázdná množina*** $\varnothing$, jež neobsahuje žádné prvky.

Kardinalita (mohutnost) konečné množiny označuje počet jejich prvků. Mezi základní pojmy teorie množin patří také *supremum* a *infimum*. Pokud $A \subseteq B$ (viz dále), pak *horní závora* množiny A je prvek $y \in B$ takový, že pro každé $x \in A$ je $x \leq y$. Obdobně *dolní závora* je prvek $z \in B$ takový, že $z \leq x$ pro každé $x \in A$. Supremum je tedy označení pro nejmenší horní závoru a jeho zápis je

$$\sup_B A \tag{3-1}$$

a největší ze všech dolních závor se nazývá infimum

$$\inf_B A. \tag{3-2}$$

3.1.1 Vztahy mezi množinami

Jestliže A a B jsou množiny, potom:

- $A = B$, právě když obsahují stejné elementy,
- $A \subseteq B$ (A je podmnožinou B), když prvky množiny A jsou i prvky množiny B,
- $A \subset B$ (A je vlastní podmnožinou B), pokud prvky množiny A jsou také prvky množiny B, ale množina B navíc obsahuje i jiné prvky,
- $B \supseteq A$ má stejný význam jako $A \subseteq B$,
- $B \supset A$ vyjadřuje totéž jako $A \subset B$.

3.1.2 Operace nad množinami

Klasická teorie množin používá několik operací pro práci s množinami:

- *průnik* – $A \cap B = \{x : (x \in A) \wedge (x \in B)\}$,
- *sjednoceni* – $A \cup B = \{x : (x \in A) \vee (x \in B)\}$,
- *doplněk* množiny A, označený jako $\overline{A}$, je množina všech prvků, které do množiny A nepatří – $\overline{A} = \{x : x \in U, x \notin A\}$,
- *rozdíl množin* – $A - B = \{x : (x \in A) \wedge (x \notin B)\}$,
- *kartézský součin* je množina všech uspořádaných dvojic, kdy první prvek je z první množiny a druhý prvek z druhé množiny – $A \times B = \{(a,b) : a \in A, b \in B\}$.

Relace je jakákoliv podmnožina kartézského součinu, kde $R \subseteq (X \times Y)$ se nazývá zobrazení, jestliže pro každý prvek $x \in X$ existuje právě jedno $y \in Y$ takové, že $(x, y) \in R$. Je-li relace *R* zobrazením, její *množina vzorů* (*definiční obor*) je definována jako

$$P_1(R) = \{x \in X : (\exists y \in Y : (x, y) \in R)\} \tag{3-3}$$

a definice *množiny obrazů* (*oboru hodnot*) je

$$P_2(R) = \{y \in Y : (\exists x \in X : (x, y) \in R)\}. \tag{3-4}$$

3.1.3 Charakteristická funkce množiny

Při určování množiny můžeme využít několika způsobů. První z nich je vyjmenování všech prvků, které do množiny patří, druhý způsob je definice množiny výrokem $P(x)$ a třetí způsob je definice množiny *A* charakteristickou funkcí μ_A.

$\mu_A : U \to [0,1]$ je *charakteristickou funkcí* množiny *A* právě tehdy, jestliže pro všechna *x* platí

$$\mu_A(U) = \begin{cases} 1 & pro\ x \in A \\ 0 & pro\ x \notin A. \end{cases} \tag{3-5}$$

Důležitá je univerzální množina *U* jako definiční obor charakteristické funkce. Tento třetí způsob se využívá především pro zobecnění klasických množin na fuzzy množiny.

Zde jsme si uvedli pouze základní skutečnosti z teorie množin, které budou v následující teorii fuzzy množin zobecněny. Rozšiřující informace o tomto tématu jsou dostupně ve zdrojích [9] a [10].

3.2 Teorie fuzzy množin

Fuzzy množina představuje ve fuzzy logice jeden ze základních pojmů, a jak již bylo řečeno, jedná se o zobecnění klasických množin.

Základní myšlenka fuzzy množin spočívá v tom, že pokud nejsme schopni stanovit přesné hranice třídy vymezené vágním pojmem, nahradíme rozhodnutí o tom, jestli prvek patří nebo nepatří do množiny mírou vybíranou z nějakého rozsahu. Každý prvek, se kterým pracujeme, tak bude mít přiřazenou míru určující jeho pozici v dané třídě.

Celý princip si můžeme vysvětlit na příkladu, kdy máme za úkol specifikovat hranice množiny pro všechny *vysoké* lidi. Jako první si určíme, že vysoký člověk má výšku mezi 170 až 240 cm. Máme tedy vstupní množinu $U = [170, 240]$ (cm). Ve chvíli, kdy budeme mít skupinu lidí, které budeme chtít rozdělit do menších skupin dle výšky, narazíme na problém. Rozhodneme-li se, že velký člověk má alespoň 180 cm, pak člověk, jehož výška je 179,4 cm, by do skupiny vysokých lidi již nepatřil. Vzhledem k tomu, že běžným, v praxi používaným, měřením nejsme schopni od sebe tyto dvě výšky odlišit, je

lepší místo přesných čísel použít vágní slova jako „malý“, „velký“, „velmi velký“, apod. V teorii fuzzy množin je proto potřeba vyjít z množiny všech možných výšek, např. $U = [50, 240]$ (cm) a každé myslitelné výšce, jež spadá do naší základní množiny, přiřadíme číslo z intervalu $[0,1]$, které bude vyjadřovat *stupeň pravdivosti* tvrzení, že člověk s touto výškou je „vysoký“. Stupeň pravdivosti 0 znamená, že s daným tvrzením naprosto nesouhlasíme, zatímco stupeň 1 definuje naprostý souhlas. Čísla mezi těmito dvěma hodnotami vyjadřují míru částečného souhlasu, např. pravda, že člověk s výškou 167 cm je vysoký bude mít stupeň $0,3$, zatímco 195 cm vysoký člověk bude mít přiřazen stupeň pravdivosti 1.

Při definici fuzzy množiny musíme nejdříve definovat množinu *U*, která představuje množinu prvků libovolného druhu (lidé, auta, čísla). Fuzzy množina *A* je pak tvořena prvky *x*, které jsou vybírány z množiny *U*, $(x \in U)$ kde každý prvek má přiřazeno číslo $\mu_A \in [0,1]$ nazývané *stupeň příslušnosti* prvku *x* do fuzzy množiny *A*. Fuzzy množinu lze definovat charakteristickou funkcí

$$\mu_A : U \to [0,1] \tag{3-6}$$

jež je nazývána *funkcí příslušnosti*, v cizí literatuře pak jako *membership function*. Funkce příslušnosti mapuje univerzum na celý interval $[0,1]$ a stupeň příslušnosti $\mu_A(x)$ prvku *x* určuje, do jaké míry je *x* prvkem dané fuzzy množiny. Má-li fuzzy množina konečný počet dvojic, lze tento vztah vyjádřit výčtem těchto dvojic

$$A = \{\mu_A(x_1)/x_1, ..., \mu_A(x_n)/x_n\}, \tag{3-7}$$

kde $x_1, ..., x_n \in U$ jsou prvky, kterým jsou přiřazeny stupně příslušnosti $\mu_A(x_1), ..., \mu_A(x_n) \in (0,1]$, což znamená, že prvky se stupněm příslušnosti 0 nejsou v množině *A* zahrnuty.

Použijeme-li pro výčet znaménko „+“, lze konečnou množinu psát i jako

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2, ..., \mu_A(x_n)/x_n = \sum_{i=1}^{n} \mu_A(x_i)/x_i . \tag{3-8}$$

Je-li fuzzy množina definována na diskrétním konečném nebo spočetném univerzu *U*, můžeme zápis zkrátit na

$$A = \sum_{x \in U} \mu_A(x)/x . \tag{3-9}$$

Pokud je univerzum spojité nebo nespočetné, potom místo symbolu „+“ použijeme symbol integrálu

$$\int_U \mu_A(x)/x\,. \tag{3-10}$$

V případech, kdy funkce příslušnosti množiny má pouze dvě hodnoty 0 nebo 1, se pro takovou množinu používá výraz *ostrá množina* (crisp set).

3.2.1 Základní pojmy

Nosič (support) je množina všech prvků univerza, které mají kladnou funkci příslušnosti.

$$\mathrm{Supp}(A) = \{x \in U \mid \mu_A(x) > 0\} \tag{3-11}$$

Jádro (core) fuzzy množiny A je ostrá množina prvků, jejichž funkce příslušnosti je rovna 1.

$$\mathrm{core}(A) = \{x \in U : \mu_A(x) = 1\} \tag{3-12}$$

α-řez (α-cut) je množina prvků, které mají stupeň příslušnosti větší nebo roven zadanému stupni α.

$$\alpha(A) = \{x \in U : \mu_A(x) \geq \alpha\} \tag{3-13}$$

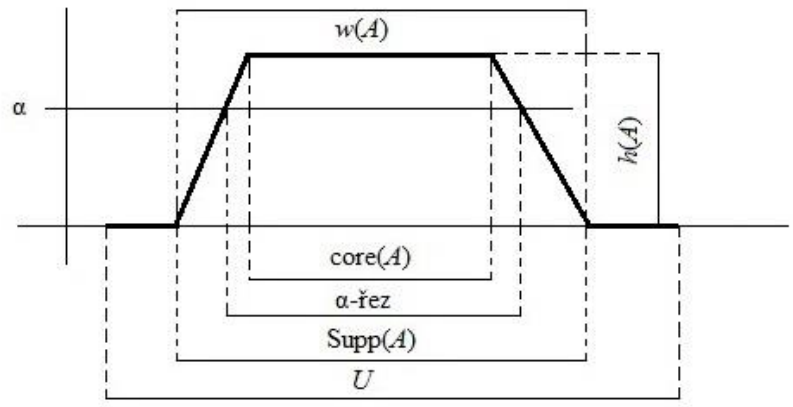
Šířka (width) množiny A s nosičem Supp(A) je definována jako

$$w(A) = \sup(\mathrm{Supp}(A)) - \inf(\mathrm{Supp}(A)) \tag{3-14}$$

a **výška** (height) množiny A jako

$$h(A) = \{x \in U : \sup(\mu_A(x))\}\,. \tag{3-15}$$

Všechny výše definované pojmy jsou grafický znázorněny na následujícím obrázku.



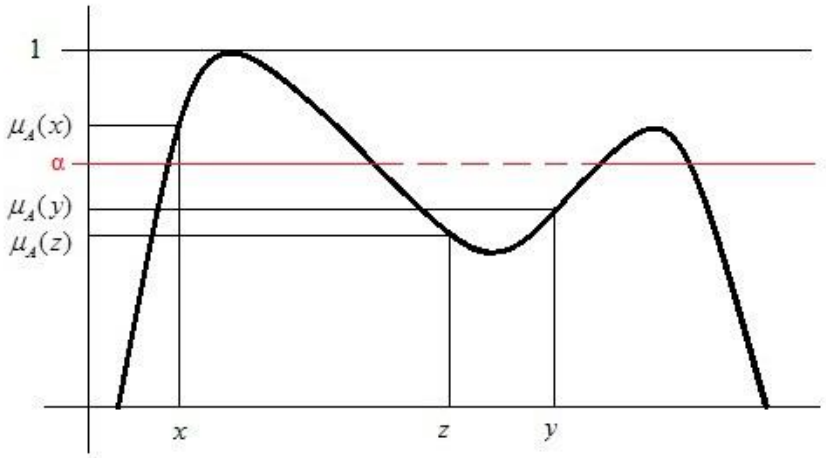
Obr. 3.1 – Základní pojmy fuzzy množin

Fuzzy množina A je **normální**, je-li její výška rovna 1, jinak se nazývá *subnormální*.

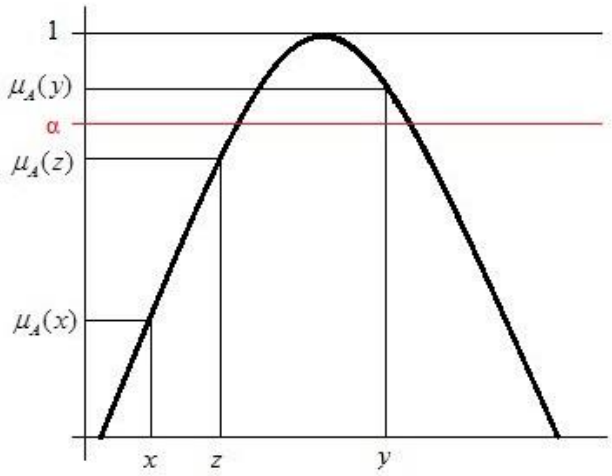
Fuzzy množina A je **konvexní**, jestliže pro každé dva prvky $x, y \in U$ a pro každé $\lambda \in [0,1]$ platí

$$\mu_A(\lambda \cdot x + (1-\lambda) \cdot y) \geq \min(\mu_A(x), \mu_A(y)). \tag{3-16}$$

To znamená, že hodnota funkce příslušnosti (např. $\mu_A(z)$) v libovolném bodě z, který se nachází mezi body x a y, je větší než menší z obou hodnot $\mu_A(x), \mu_A(y)$. Celá situace je vysvětlena na obr. 3.2 a obr. 3.3. Dalším pravidlem je, že fuzzy množina je konvexní, pokud každý její α-řez tvoří souvislý interval.



Obr. 3.2 – Nekonvexní fuzzy množina



Obr. 3.3 – Konvexní fuzzy množina

3.2.2 Typy fuzzy množin

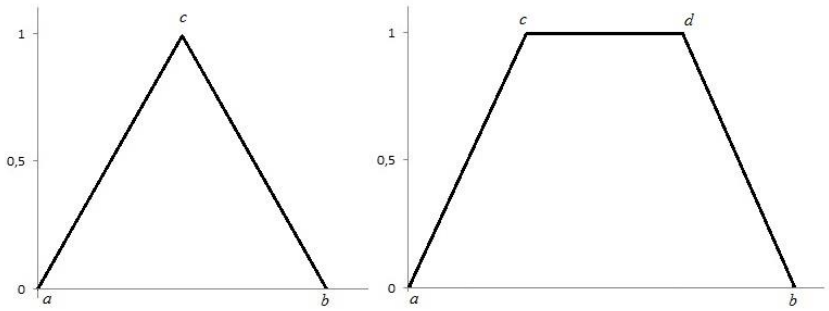
Pokud pracujeme s numerickými daty, univerzum bude definováno např. jako interval reálných čísel. Typy funkce příslušnosti, které jsou v teorii řízení nejčastěji používány, jsou trojúhelníková (triangular), lichoběžníková (trapezoidal), Gaussova (Gaussian), S-křivka a Z-křivka.

Trojúhelníková křivka je určena dvěma krajními body $(a,0)$ a $(b,0)$ a vrcholem (c,α), pro který platí

$$a < c < b. \tag{3-17}$$

Průběh funkce je pak definován

$$\mu_A(x) = \begin{cases} 0 & x \le a \\ \dfrac{x-a}{c-a} & a \le x \le c \\ \dfrac{x-b}{c-b} & c \le x \le b \\ 0 & b \le x \end{cases}. \tag{3-18}$$



Obr. 3.4 – Trojúhelníková (vlevo) a lichoběžníková (vpravo) množina

Čtyřúhelníková funkce je tvořena dvěma krajními body $(a,0)$ a $(b,0)$ a dvěma vrcholy (c,α) a (d,α) a je určena

$$\mu_A(x) = \begin{cases} 0 & x \le a \\ \dfrac{x-a}{c-a} & a \le x \le c \\ \alpha & c \le x \le d \\ \dfrac{x-b}{d-b} & d \le x \le b \\ 0 & b \le x \end{cases}. \tag{3-19}$$

Gaussova křivka je definována

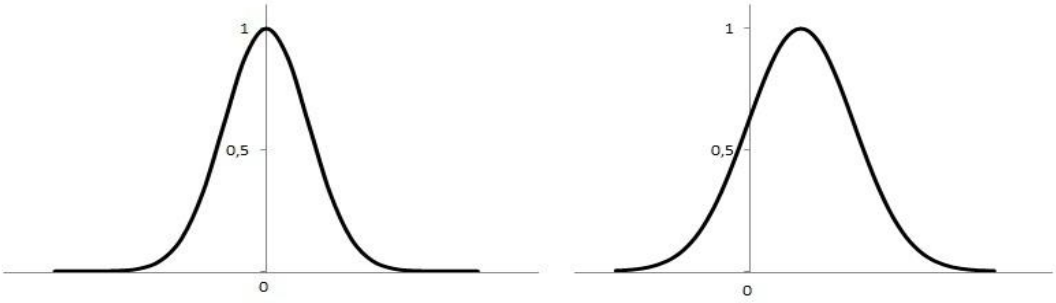
$$\mu_A(x) = e^{-\frac{(x-c)^2}{2\sigma^2}}, \tag{3-20}$$

kde parametry *c* a σ určují střed a tvar funkce. Pokud $c = 0$ a $\sigma = 0$ dostaneme standardní funkci příslušnosti ve tvaru

$$\mu_A(x) = e^{-\frac{x^2}{2}} \tag{3-21}$$

která je na obr. 3.5 vlevo. Plocha pod křivkou odpovídá $\sqrt{2\pi}$. Průběh funkce na obr.3.5 vpravo je definován vztahem

$$\mu_A(x) = e^{-\frac{(x-3)^2}{20}}. \tag{3-22}$$

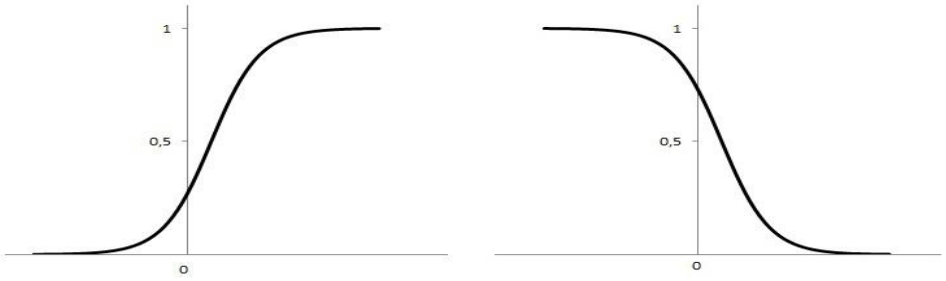


Obr. 3.5 – Gaussovy funkce příslušnosti

S-křivka a Z-křivka jsou sigmoidální funkce ve tvaru

$$\mu_A(x) = \frac{1}{1+e^{-(x-m)\sigma}}, \tag{3-23}$$

kde parametr σ určuje, jestli funkce bude rostoucí $(\sigma = 1)$ nebo klesající $(\sigma = -1)$, zatímco parametr *m* posouvá funkci doprava nebo doleva.



Obr. 3.6 – S-křivka (vlevo) a Z-křivka (vpravo)

S-křivka na obr. 3.6 je definována rovnicí (3-24) a Z-křivka rovnicí (3-25).

$$\mu_A(x) = \frac{1}{1+e^{-x+1}} \tag{3-24}$$

$$\mu_A(x) = \frac{1}{1+e^{x-1}} \tag{3-25}$$

3.2.3 Operace s fuzzy množinami

Ve fuzzy řízení pracujeme s kolekcemi fuzzy množin, a proto je nutné mít prostředky, které umožňují práci s nimi. Stejně jako s klasickými množinami lze s fuzzy množinami provádět základní operace sjednocení, průnik a doplněk. Existují ale i další operace jako *t-norma*, *t-konorma*, *negace*, které v klasické teorii množin nemají smysl nebo jejich výsledek je stejný jako při provedení některé ze základních operací.

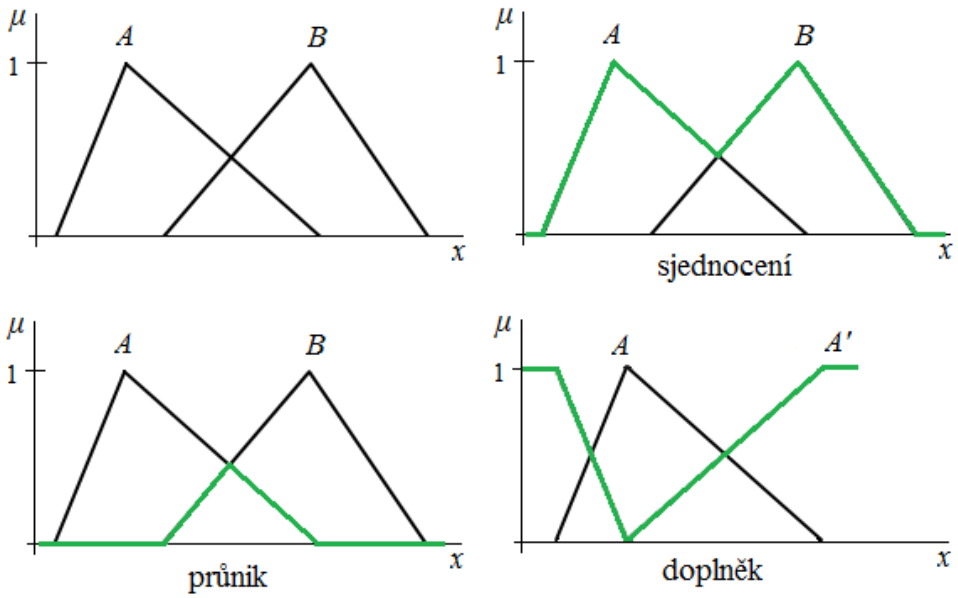
Operace **průniku** (intersection), **sjednocení** (union) nad množinami A a B a unárního **doplňku** (complement) pro všechna $x \in U$ jsou ve stejném pořadí definována následovně

$$\mu_{A\cap B}(x) = \min\left(\mu_A(x), \mu_B(x)\right), \tag{3-26}$$

$$\mu_{A\cup B}(x) = \max\left(\mu_A(x), \mu_B(x)\right), \tag{3-27}$$

$$\mu_{A'}(x) = 1 - \mu_A(x). \tag{3-28}$$

Grafické zobrazení těchto operací můžeme vidět na následujícím obrázku.



Obr. 3.7 – Operace s fuzzy množinami

Jednou z rozšiřujících operací s fuzzy množinami je **triangulární norma**, obvykle označovaná jako **t-norma**, v cizojazyčné literatuře pak jako triangular norm nebo t-norm. Jedná se o zobecnění operace AND, kdy t-norma je binární operace průniku

$$\circ : [0,1] \times [0,1] \to [0,1], \tag{3-29}$$

která musí pro všechna $x, y, z \in [0,1]$ splňovat následující pravidla:

- $x \circ y = y \circ x$ – komutativní zákon,
- $x \circ (y \circ z) = (x \circ y) \circ z$ – asociativní zákon,
- $y \le z \Rightarrow x \circ y \le x \circ z$ – monotónnost,
- $x \circ 1 = 1 \circ x = x$ – okrajová podmínka.

Pojem „binární operace“ znamená, že se jedná o funkci dvou proměnných. Nejpoužívanější t-norma je operace minima, která byla uvedena již v rovnici (3-26) při definici průniku fuzzy množin. Mezi další často používané t-normy patří součinová konjunkce (algebraický součin) (3-31), Łukasiewiczova konjunkce (omezený součin) (3-32) a drastický součin, který je definován následovně

$$\mu_{A \cap B}(x) = \begin{cases} \mu_A(x) & \mu_B(x) = 1 \\ \mu_B(x) & \mu_A(x) = 1 \\ 0 & \mu_A(x) \vee \mu_B(x) < 1 \end{cases}. \tag{3-30}$$

$$\mu_{A \cap B}(x) = \mu_A(x) \cdot \mu_B(x) \tag{3-31}$$

$$\mu_{A \cap B}(x) = \max\left(0, \mu_A(x) + \mu_B(x) - 1\right) \tag{3-32}$$

Operace t-normy jsou znázorněny na obr. 3.8.

Další fuzzy operací je **triangulární konorma**, označovaná také jako **t-konorma** nebo **s-norma**, v cizojazyčné literatuře pak jako triangular conorm nebo t-conorm a odpovídá zobecnění operace OR. T-konorma je binární operace sjednocení

$$* : [0,1] \times [0,1] \to [0,1], \tag{3-33}$$

Která musí pro všechna $x, y, z \in U$ splňovat tyto pravidla:

- $x * y = y * x$ – komutativní zákon,
- $x * (y * z) = (x * y) * z$ – asociativní zákon,
- $y \le z \Rightarrow x * y \le x * z$ – monotónnost,
- $x * 0 = 0 * = x$ – okrajová podmínka.

Nejpoužívanější t-konorma je operace maxima, která byla uvedena již v rovnici (3-27) při definici sjednocení fuzzy množin. Mezi další často používané t-konormy patří součinová disjunkce (algebraický součet) (3-35), Łukasiewiczova disjunkce (omezený součet) (3-36) a drastický součet, který je definován následovně

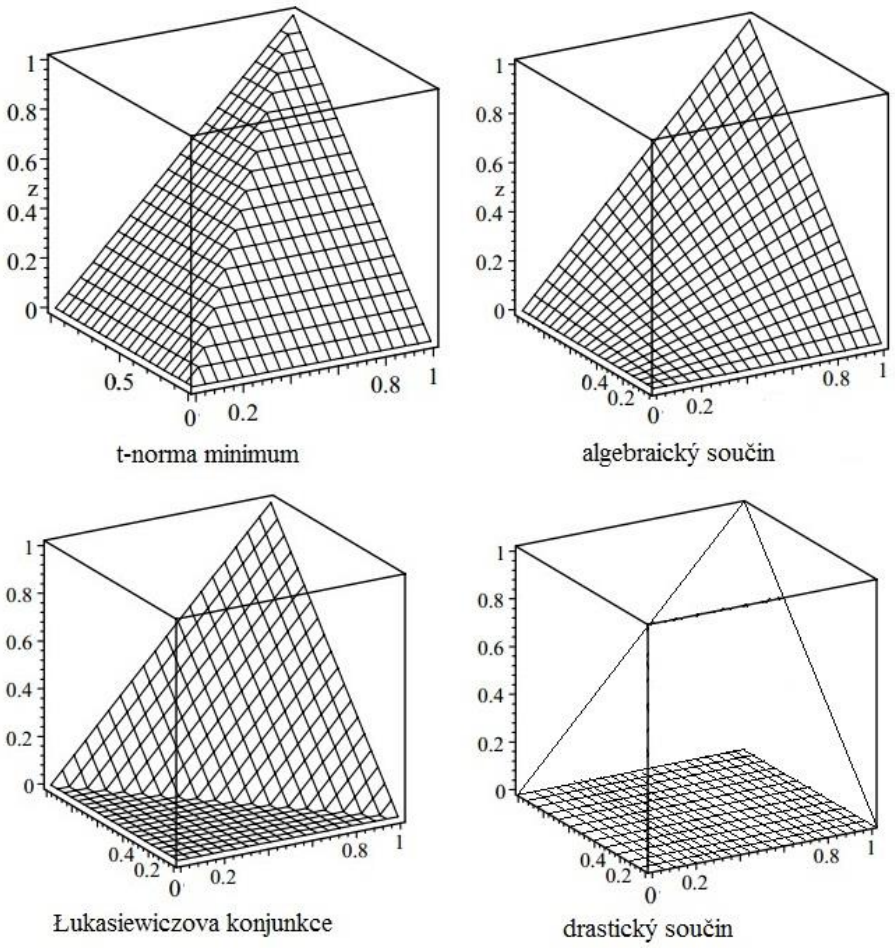
$$\mu_{A\cup B}(x)=\begin{cases}\mu_A(x) & \mu_B(x)=0\\ \mu_B(x) & \mu_A(x)=0\\ 1 & \mu_A(x)\vee\mu_B(x)>0\end{cases}. \tag{3-34}$$

$$\mu_{A\cup B}(x)=\mu_A(x)+\mu_B(x)-\mu_A(x)\cdot\mu_B(x) \tag{3-35}$$

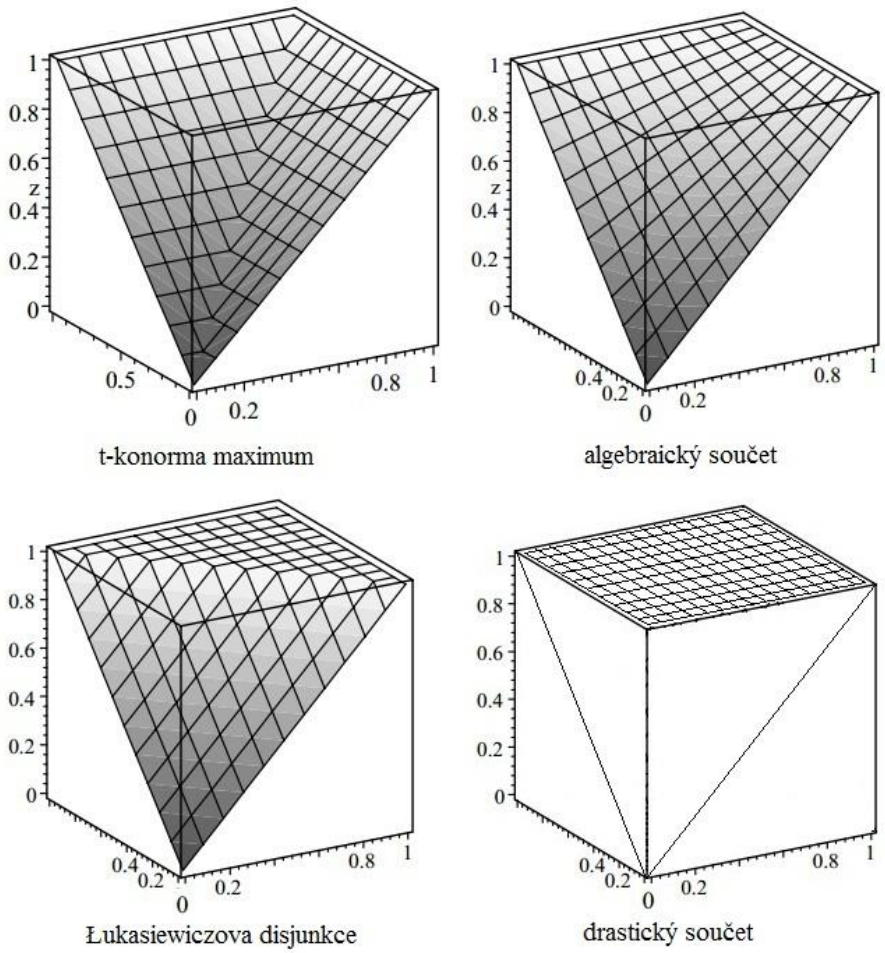
$$\mu_{A\cup B}(x)=\min\left(1,\mu_A(x)+\mu_B(x)\right) \tag{3-36}$$

Operace t-konormy jsou znázorněny na obr. 3.9.

Jakou z uvedených variant pro danou operaci použít záleží na konkrétní řešené úloze.



Obr. 3.8 – Operace t-normy



Obr. 3.9 – Operace t-konormy

Poslední fuzzy operace, kterou si definujeme se **negace**. Jedná se o unární operaci

$$\neg : [0,1] \to [0,1], \tag{3-37}$$

která splňuje následující tvrzení:

- $\neg(\neg(x)) = x$,
- $\neg(1) = 0$,
- $\neg(0) = 1$,
- $x \le y \Rightarrow \neg y \le \neg x$.

Rozšiřující informace o operacích s fuzzy množinami můžeme dohledat v knize „Základy fuzzy modelování“ [11] od strany 24, ve zdroji [12] od strany 28 anebo ve skriptech „Základy fuzzy množin“ [13].

3.3 Fuzzy relace

V klasické teorii množin byla relace definována jako množina uspořádaných dvojic, stejně tak fuzzy relace představuje fuzzy množinu uspořádaných dvojic. Máme-li diskrétní konečná univerza U a V a funkci příslušnosti $\mu_R : U \times V \to [0,1]$, představující kartézský součin na intervalu $[0,1]$, pak

$$R = \sum_{U \times V} \mu_R(u,v)/(u,v) \tag{3-38}$$

je binární relace na kartézském součinu $U \times V$. Jsou-li daná univerza spojitá, je binární relace definována

$$R = \int_{U \times V} \mu_R(u,v)/(u,v). \tag{3-39}$$

3.3.1 Operace s fuzzy relacemi

Stejně jako byly u fuzzy množin zavedeny operace průniku a sjednocení, budou stejné operace zavedeny i pro fuzzy relace.

Příklady si uvedeme na binárních relacích *R* a *S*, jež jsou definované na kartézském součinu $X \times Y$. Funkce příslušnosti *průniku relací* je pro všechna *x, y* definována jako

$$\mu_{R \cap S} = \min\left(\mu_R(x,y), \mu_S(x,y)\right). \tag{3-40}$$

Funkce příslušnosti *sjednocení relací* je pro všechna *x, y* definována jako

$$\mu_{R \cup S} = \max\left(\mu_R(x,y), \mu_S(x,y)\right). \tag{3-41}$$

Ve funkci příslušnosti průniku relací lze místo operace minima použít libovolnou t-normu, stejně tak u funkce příslušnosti sjednocení relací lze místo operace maxima použít libovolnou t-konormu. Obě definice mohou být rozšířeny na *n*-nární relace.

Dalšími důležitými operacemi na fuzzy relacích jsou operace ***projekce*** (*projection*) a operace ***cylindrického rozšíření*** (*cylindric extension*). Máme-li binární relaci *R* definovanou na kartézském součinu $X \times Y$, potom *projekce R* na *Y* je fuzzy množina

$$proj\ R\ na\ Y = \int_Y \sup_{\forall x} \mu_R(x,y) \Big/ y. \tag{3-42}$$

Operace projekce vytváří obecně z *n*-nární relace *l*-nární relaci, např. z ternární relace relaci binární nebo z binární relace unární relaci, což je fuzzy množina.

Opačnou operací k projekci je operace *cylindrické rozšíření*. Uvažujeme-li fuzzy množinu *A* definovanou na univerzu *Y*, pak cylindrické rozšíření *A* na $X \times Y$, je množina všech dvojic $(x, y) \in X \times Y$ s funkcí příslušnosti $\mu_{ce(A)}(x, y)$. Tento vztah lze vyjádřit jako

$$ce(A) = \int_{X \times Y} \mu_A(y)/(x, y). \tag{3-43}$$

Kombinace cylindricky rozšířené fuzzy množiny, fuzzy relace a následná projekce se nazývá ***kompozice*** (*composition*). Je dána fuzzy množina *A*, definovaná na univerzu *X* a fuzzy relace *R* definovaná na kartézském součinu $X \times Y$. Kompozice *A* a *R* je potom fuzzy množina *B* definovaná na *Y* a platí pro ni

$$B = A \circ R = proj\left(ce(A) \cap R\right) na\ Y. \tag{3-44}$$

Jestliže je průnik $\circ$ vytvořen operací *min* a projekce operací *max*, potom funkce příslušnosti množiny *B* – kompozice má tvar

$$\mu_B(y) = \max_{\forall x}\left(\min\left((\mu_A(x), \mu_R(x, y))\right)\right), \tag{3-45}$$

Který se nazývá *max-min kompozice* nebo *kompozice relací*.

3.4 Jazyková proměnná

Jazyková proměnná není proměnnou v klasickém slova smyslu, ale slouží jako základní jednotka pro reprezentaci znalostí. V literatuře ji můžeme nalézt i pod názvem lingvistická proměnná. Hodnoty jazykové proměnné nejsou čísla, ale slova nebo věty přirozeného nebo umělého jazyka a jejich význam je reprezentován fuzzy množinami v nějakém univerzu. Např. rychlost je jazykovou proměnnou, pokud její hodnoty jsou vyjádřeny slovy *pomalu, rychleji, rychle, velmi rychle* a nikoliv čísly 10, 20, 30. Ve fuzzy řízení se jako jazyková proměnná často používá odchylka, která může nabývat *velkých, středních, malých, kladných* nebo *záporných* hodnot.

V pravidlech, o kterých si více řekneme dále, představuje jazyková proměnná výroky. Nejde však o výroky, jejichž výsledky jsou omezeny pouze na „pravda“ či „nepravda“, ale o výroky, jejichž pravdivostní hodnota je v intervalu $[0,1]$, tedy fuzzy výroky.

Z předešlého popisu je tedy zřejmé, že jazyková proměnná má dva parametry. Prvním je její symbolické jméno, které budeme označovat písmenem *x* a druhým parametrem je množina slovních hodnot *T*(*x*), kterých může nabývat. Dalším parametrem jazykové proměnné je rozsah čísel, univerzum U_x, ve kterých se popisovaná veličina pohybuje. V případě zmiňované rychlosti to je např. $U_x = [0,120]$, u výšky lidí např. $U_x = [30, 240]$. Dále musí být daná funkce M_x, která mapuje slovní hodnoty na hodnoty

univerza

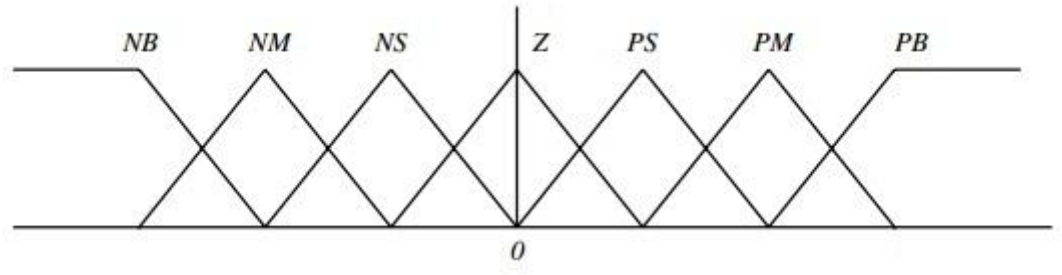
$$M_x : T(x) \rightarrow \mu_{T(x)}\,. \tag{3-46}$$

Jazykovou proměnnou lze zapsat jakou uspořádanou čtveřici následujícím způsobem

$$\langle x, T(x), U_x, M_x \rangle. \tag{3-47}$$

Ve fuzzy řízení se obvykle pracuje se skupinou množin, které jsou uspořádány v jazykové proměnné. Skupinu mohou tvořit množiny s různými funkcemi příslušnosti. Jako vzorový příklad jazykové proměnné, s typickými názvy množin a tvary funkcí příslušnosti, se používá model na obr. 3.10, kde označení množin má následující význam:

- ***NB*** (Negative Big) – „negativní velký“,
- ***NM*** (Negative Medium) – „negativní střední“,
- ***NS*** (Negative Small) – „negativní malý“,
- ***Z*** (Zero) – „nulový,
- ***PS*** (Positive Small) – „pozitivní malý“,
- ***PM*** (Positive Medium) – „pozitivní střední“,
- ***PB*** (Positive Big) – „pozitivní velký“.



Obr. 3.10 – Model jazykové proměnné

Při přibližném usuzování, kdy rozhodujeme, jestli nějaká skutečnost nastala nebo ne, pracujeme se znalostmi v jazykové formě a ty je potřeba např. pro fuzzy řízení převést do jazykové proměnné. Převedení znalosti do jazykové proměnné se provádí ve třech krocích:

- zvolíme symbol pro popisovanou fyzikální proměnnou,
- pro hodnoty, kterých může fyzikální hodnota nabývat, zvolíme označení,
- vyjádříme skutečnost, že fyzikální proměnná má nějakou hodnotu.

Např.:

- pro fyzikální veličinu *odchylka* zvolíme symbol *e*,
- pro její *kladnou střední* hodnotu zvolíme označení *PM*,
- vyjádříme slovní spojení „odchylka je středně velká“ – *e* je *PM* nebo také *e* = *PM*.

Toto slovní spojení představuje nejjednodušší fuzzy výrok, který se nazývá *atomický* a jeho význam určuje funkce příslušnosti μ_{PM}, jež je definována na univerzu U_e. Hodnota funkce příslušnosti, která je z intervalu $[0,1]$, udává míru, s jakou hodnota fyzikální veličiny *odchylka* patří do fuzzy množiny *PM*.

Atomické fuzzy výroky můžeme spojovat spojkami *and*, *or*, *not* a vytvářet tak složitější fuzzy výroky (viz dále).

Máme-li jazykovou proměnnou *v* na univerzu U_v a fuzzy množinu *B*, která reprezentuje slovní hodnotu „big“, jejich vztah lze definovat jako

$$B = \int_{U_v} \mu_B(v)/v\,. \tag{3-48}$$

Pokud ve fuzzy řízení pracujeme s více vstupními veličinami, které jsou definované na různých univerzech, nebudou nám atomické fuzzy výroky stačit a bude potřeba použít *složené výroky*. Uveďme si příklad se dvěma jazykovými proměnnými, úhlem *a* a úhlovou rychlostí *v*, jejichž univerza jsou U_a a U_v. Dále musíme mít dva fuzzy výroky, např. „*a* je *PS*“ a „*v* je *NM*“. Fuzzy množiny *PS* a *NM* reprezentující slovní hodnoty „pozitivní velká“ a „negativní střední“ jsou definovány jako

$$PS = \int_{U_a} \mu_{PS}(a)/a, \quad NM = \int_{U_v} \mu_{NM}(v)/v\,. \tag{3-49}$$

V tomto případě je potřeba nejprve provést cylindrické rozšíření fuzzy množin na kartézský součin $U_a \times U_v$. Složený fuzzy výrok *r*: „(*a* je *PS*) *and* (*v* je *NM*)“ je reprezentován fuzzy relací definovanou na kartézském součinu $U_a \times U_v$

$$\mu_r(a,v) = \int_{U_a \times U_v} \min\left(\mu_{PS}(a), \mu_{NM}(v)\right)/(a,v)\,, \tag{3-50}$$

kde místo operátoru *min* můžeme použít jinou t-normu.

3.5 Fuzzy pravidla

V klasické logice, ale i v oblastech regulace, řízení či rozpoznávání objektů se používají podmíněné výroky typu

$$if\ (\ fuzzy\ výrok)\ then\ (\ fuzzy\ výrok), \tag{3-51}$$

které tvoří fuzzy pravidla, jež určují chování systému v daných situacích. S fuzzy pravidly typu JESTLIŽE-PAK, se setkáváme i v každodenním lidském životě. Fuzzy pravidla jsou ve formě logické implikace a fuzzy výroky v nich obsažené mohou být atomické nebo složené, pokud pracujeme s více vstupními parametry, viz kapitola 3.4.

První fuzzy výrok ve fuzzy pravidle, před částí *then*, se nazývá *antecedent* (*předpoklad, premisa*), druhý výrok, za částí *then*, se nazývá *konsekvent* (*závěr*) fuzzy implikace.

Pravidlo *if-then* představuje vztah mezi výroky. Vezmeme-li si např. případ, kdy fuzzy řízením regulujeme nějaký dynamický systém, vstupní veličinou systému bude *odchylka* systému od požadovaného stavu *e* a výstupem bude *akční zásah u*, který má být na systém aplikován, pak pravidlo ve tvaru

$$if\ (e\ je\ NM)\ then\ (u\ je\ PS) \tag{3-52}$$

vyjadřuje, že pokud je hodnota odchylky „záporná střední“, je třeba, aby hodnota akčního zásahu spadala do množiny „pozitivní malá“. Tento vztah představuje relaci mezi veličinami *e* a *u* a je možné je vyjádřit ve formě fuzzy relace, která bude reprezentována fuzzy implikací.

Ve chvíli, kdy budeme chtít popsat veškeré chování regulované soustavy, se znalostí jednoho pravidla nevystačíme. Z toho vyplývá, že ve fuzzy řízení pracuje se souborem (množinou) pravidel ve tvaru (3-51), který se nazývá *jazykový popis*. Pokud si fuzzy pravidlo označíme písmenem *R*, potom zápis množiny fuzzy pravidel je následující

$$R = \{R_1, R_2, \ldots, R_m\}. \tag{3-53}$$

3.6 Typy fuzzy systémů

I když na fuzzy systémy můžeme nahlížet z mnoha hledisek, uvedeme si dvě stěžejní pro fuzzy regulaci.

Prvním je počet vstupních a výstupních veličin systému. Rozdělení je pak následující:

- SISO (Single-input, single-output) – jeden vstup a jeden výstup,
- MISO (Multi-input, single-output) – více vstupů a jeden výstup,
- MIMO (Multi-input, multi-output) – více vstupů a více výstupů.

Druhým hlediskem je tvar pravidel. V literatuře se nejčastěji setkáváme s fuzzy systémy typu *Mamdani* nebo *Sugeno*, které si dále popíšeme.

3.6.1 Fuzzy systém Mamdani

Fuzzy systém Mamdani je postaven na souboru pravidel, ve tvaru JESTLIŽE-PAK popsaném již dříve vztahem (3-51). V konsekventu pravidel se nachází fuzzy výrok, jehož výsledkem je příslušná fuzzy množina.

3.6.2 Fuzzy systém Sugeno

Model Sugeno můžeme v literatuře nalézt také pod názvem *Takagi-Sugeno-Kang model* nebo pod zkratkou TSK. Oproti předchozímu systému se v konsekventu pravidel nenachází fuzzy výrok, ale nějaká lineární funkce ve tvaru

$$f_k = (x_1, \ldots, x_n) = a_{k0} + a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kn}x_n \quad k = 1, 2, \ldots, r, \qquad (3\text{-}54)$$

kde *k* je číslo pravidla a *r* označení pro celkový počet pravidel.

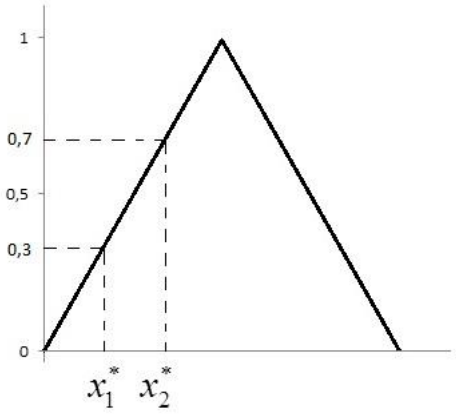
Pravidlo je pak ve tvaru

$$f\,(x_1 = A_1^k)\ and \ldots and\ (x_n = A_n^k)\ then\ y = f_k(x_1, \ldots, x_n) \quad k = 1, 2, \ldots, r. \qquad (3\text{-}55)$$

Rozšiřující informace či popis dalších typů modelů je ve zdroji [1] od strany 110.

3.7 Fuzzifikace

Při fuzzifikaci dochází k převodu ostré hodnoty x^* proměnné *x* na fuzzy množinu. Cílem je přiřadit vstupní veličině nabývající hodnotu x^*, odpovídající stupeň příslušnosti, viz obr. 3.11. Ten je závislý na tvaru funkce příslušnosti dané množiny.



Obr. 3.11 – Fuzzifikace

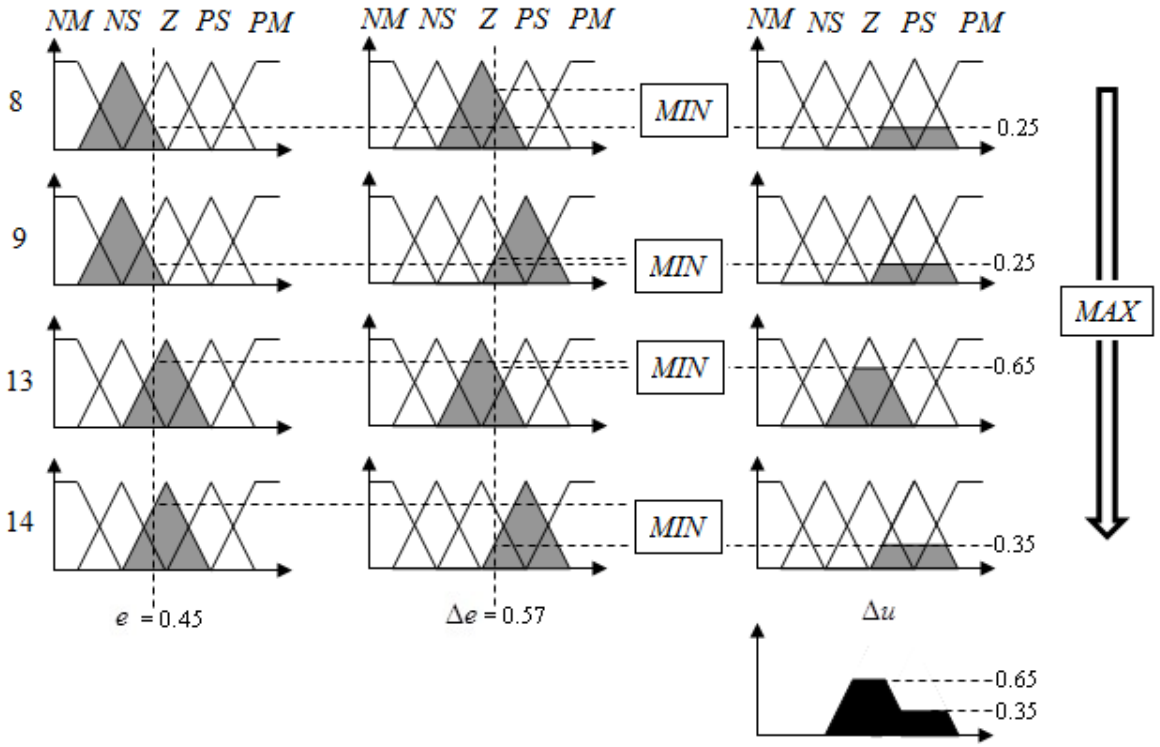
3.8 Inferenční mechanizmus

Budeme-li uvažovat fuzzy model Mamdani, inferenční mechanizmus je proces, jehož výstupem je výsledná fuzzy množina odpovídající akčnímu zásahu. Nejdříve fuzzifikací získáme stupeň příslušnosti pro naměřenou hodnotu vstupní veličiny. Pokud pracujeme s fuzzy systémem typu MISO (viz kapitola 3.6), pak provádíme fuzzifikaci pro každou vstupní veličinu zvlášť.

Dalším krokem je vyhodnocení každého pravidla zvlášť. Jako výsledek každého pravidla získáme fuzzy množinu, která je následně, na základě stupňů příslušnosti vstupních veličin, ořezána operací *min* (fuzzy logické *and*). Upravená množina reprezentuje akční zásah odpovídající danému pravidlu.

Výsledná fuzzy množina je vytvořená agregací (operace *max*, fuzzy logické *or*) dílčích výsledků. Abychom z výsledné fuzzy množiny dostali ostrou hodnotu akčního zásahu, je potřeba na ni aplikovat některou z metod defuzzifikace (viz následující kapitola).

Celý průběh inferenčního mechanizmu je znázorněn na obr. 3.12, kde pracujeme se dvěma vstupními veličinami *odchylkou e* a *změnou odchylky* Δe. Výstupem je *změna akčního zásahu* Δu. Na začátku každého pomyslného řádku procesu je číslo pravidla, které bylo uplatněno. Všechna pravidla jsou znázorněna v tab. 1, kde u každého konsekventu je v horním indexu číslo daného pravidla. Pro přehlednost jsou zelenou barvou vyznačena pravidla, která jsou pro konkrétní ostré hodnoty vstupních veličin aktivní.



Obr. 3.12 – Inferenční mechanizmus

Tab. 1 – Fuzzy pravidla

		0	0	0,65	0,35	0
	e\Δe	NM	NS	Z	PS	PM
0	NM	NM [1]	NM [2]	NS [3]	Z [4]	PS [5]
0,25	NS	NM [6]	NS [7]	PS [8]	PS [9]	PM [10]
0,3	Z	NM [11]	NS [12]	Z [13]	PS [14]	PM [15]
0	PS	NS [16]	Z [17]	PS [18]	PM [19]	PM [20]
0	PM	NS [21]	Z [22]	PS [23]	PM [24]	PM [25]

3.9 Defuzzifikace

Cílem defuzzifikace je získat ostrou hodnotu výstupní veličiny $y*$ z výsledné fuzzy množiny, která byla vypočítaná v inferenčním mechanizmu. I když ve fuzzy řízení pracujeme s nepřesností, je potřeba z fuzzy regulátoru získat dostatečně přesnou hodnotu, aby bylo možné systém regulovat.

Některé metody defuzzifikace si popíšeme v následujícím textu.

3.9.1 Metoda středu plochy

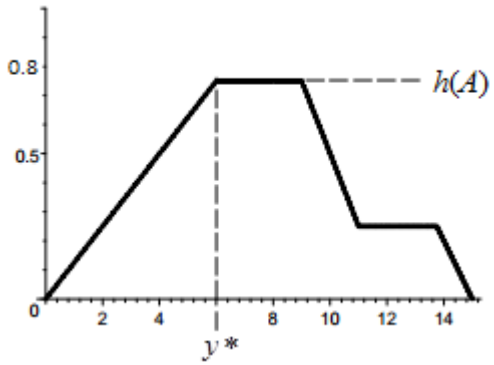
Metoda středu plochy bývá často nazývána jako *metoda těžiště*, v cizojazyčné literatuře pak *Center of Area*, *Center of Gravity* nebo *Centroid*. Její zkratka je COA. I přes velkou výpočetní složitost se jedná se nejpoužívanější metodu defuzzifikace. Jak je z názvu patrné, ostrá hodnota výstupní veličiny $y*$ je určena jako souřadnice středu těžiště výstupní fuzzy množiny. Pokud je výsledná fuzzy množina A spojitá, pro výpočet ostré hodnoty se používá následující vztah

$$y* = \frac{\int_U x \cdot \mu_A(x) dy}{\int_U \mu_A(x) dy}. \tag{3-56}$$

3.9.2 Metoda prvního maxima

Metoda prvního maxima (FoM, *First of Maximum*) vyhodnocuje jako ostrou hodnotu nejmenší hodnotu z univerza U, která má maximální stupeň příslušnosti ve výsledné fuzzy množině A a je definována jako

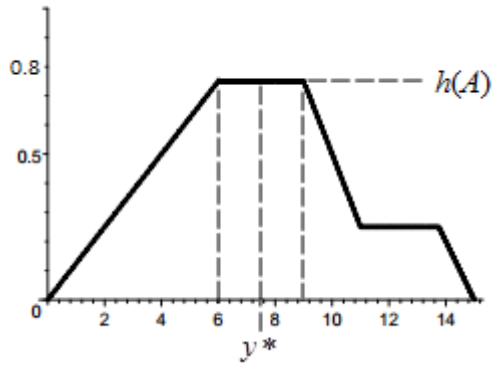
$$y^* = \inf \inf_{x \in U} \{x \in U / \mu_A(x) = h(A)\}. \tag{3-57}$$



Obr. 3.13 – Metoda prvního maxima

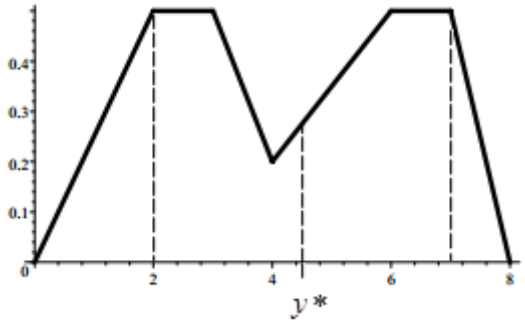
3.9.3 Metoda středu maxima

Metoda středu maxima (MoM, *Middle of Maxima*) je podobná předchozí defuzzifikační metodě. Rozdíl je, že ostrá hodnota v této metodě je vypočítaná jako aritmetický průměr hodnot prvního a posledního maxima. Situace je zobrazena na obr. 3.14.



Obr. 3.14 – Metoda středu maxima

Při defuzzikaci touto metodou můžeme dospět k nežádoucím výsledkům a to pokud je maximální výška výsledné fuzzy množiny rozdělena do více části, viz obr. 3.15.



Obr. 3.15 – Chyba metody středu maxima

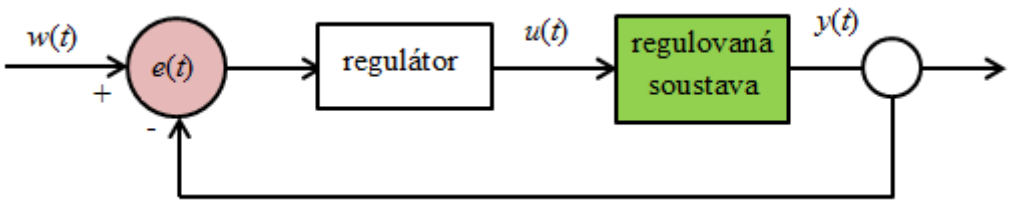
Další defuzzifikační metody jsou popsány v knihách „A First Course in Fuzzy and Neural Control“ [1], na straně 120, Základy fuzzy modelování [11], strana 95 nebo ve zdroji [12] na straně 66.

3.10 Fuzzy regulátor

Základem všech regulačních a řídicích systémů je regulace v uzavřené smyčce se zpětnou vazbou, která je znázorněna na obr. 3.16. Význam jednotlivých proměnných je následující:

- $e(t)$ – regulační odchylka v čase t,
- $u(t)$ – akční zásah regulátoru,
- $y(t)$ – výstup z regulované soustavy,
- $w(t)$ – požadovaná hodnota.

Požadovaná hodnota představuje stav, ve kterém by se měl regulovaný systém udržovat. V průběhu regulace může docházet ke změnám této hodnoty, ale obvykle hodnota zůstává, alespoň po určitý časový úsek konstantní.



Obr. 3.16 – Schéma regulace se zpětnou vazbou

Struktura regulovaného systému, viz obr. 3.16, zůstává stejná při použití klasického řízení i při fuzzy řízení. Rozdíl je pouze v návrhu regulátoru.

Jak již bylo v úvodu kapitoly 3 řečeno, hlavní princip fuzzy regulátoru nespočívá ve schopnosti regulovat systém postavený na jeho matematickém popisu, ale ve schopnosti řídit systém na základě znalostí o něm. Tedy jak se chovat, když nastane určitá situace.

Strategie řízení je postavena na pravidlech, která byla popsána v kapitole 3.5. Postup vyhodnocení naměřených hodnot vstupních veličin systému za účelem získání hodnoty akčního zásahu je podrobně popsán v kapitole 3.8.

3.10.1 Struktura fuzzy regulátoru

Fuzzy regulátor se skládá z několika modulů:

- fuzzifikace,
- inferenční mechanizmus,
- báze pravidel,
- báze dat,
- defuzzifikace.

Některé z nich byly částečně popsány již v předchozích kapitolách. Strukturu fuzzy regulátoru můžeme vidět na obr. 3.17.

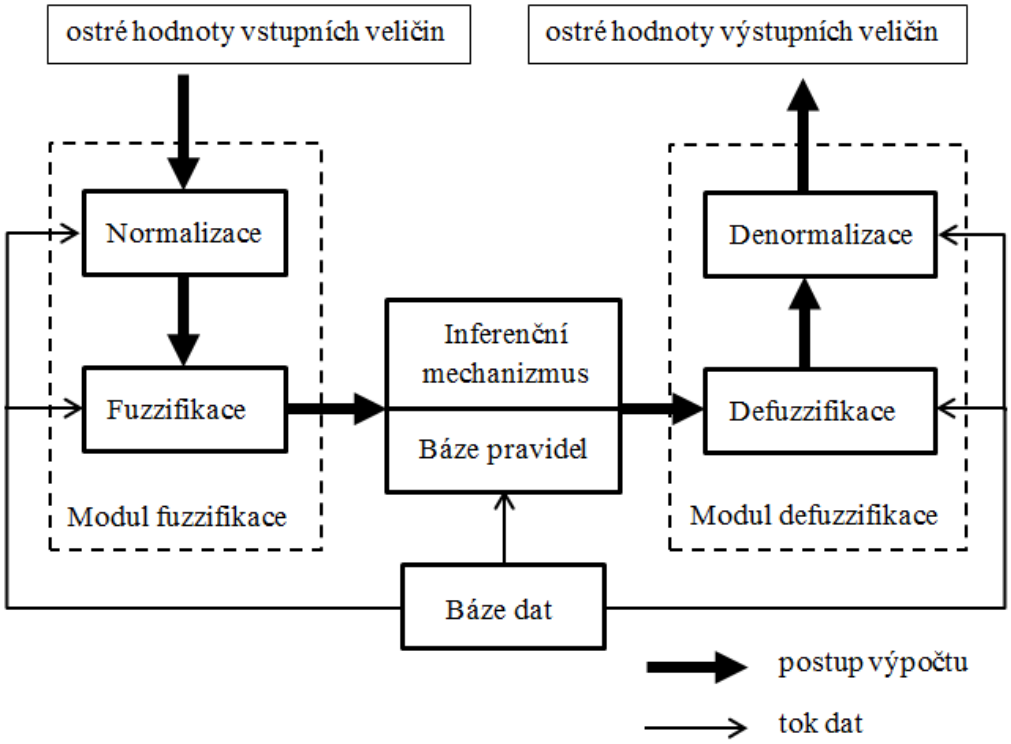
Modul fuzzifikace

V modulu fuzzifikace jsou prováděny dvě operace, a to normalizace a fuzzifikace. Normalizace představuje proces, kdy ostré hodnoty vstupních veličin, které se mohou pohybovat v různém rozsahu, jsou transformovány tak, aby nabývaly hodnot z určitého univerza.

Normalizované hodnoty jsou následně zpracovány fuzzifikací, kdy je ostré hodnotě vstupní veličiny přiřazen stupeň příslušnosti ke konkrétní fuzzy množině. Tyto fuzzy množiny pak stávají antecedenty fuzzy pravidel.

Inferenční mechanizmus

Výsledkem inferenčního mechanizmu je fuzzy množina, jež je agregovaná z výsledků pravidel (dílčí fuzzy množiny), která byla hodnotami vstupních veličin aktivována. Podrobně je tento proces popsán v kapitole 3.8.



Obr. 3.17 – Schéma struktury fuzzy regulátoru

Modul defuzzifikace

Z předchozího kroku, inferenčního mechanizmu, jsme dostali fuzzy množinu, kterou je třeba některou z metod defuzzifikace (viz kapitola 3.9) upravit tak, abychom získali ostrou hodnotu akčního zásahu.

Stejně jako bylo potřeba normalizovat hodnoty vstupních veličin, je potřeba výslednou hodnotu akčního zásahu, získanou defuzzifikací, denormalizovat. Denormalizace představuje přepočet výstupní veličiny na fyzikální výstupní veličinu.

Báze dat

Báze dat obsahuje informace o fuzzy množinách (funkcích příslušnosti), které reprezentují každou slovní hodnotu jednotlivých jazykových proměnných. Další informace uložené v bázi dat jsou rozsahy univerz potřebné při normalizaci vstupních hodnot a fyzikální rozsahy výstupních veličin potřebné při denormalizaci.

Báze pravidel

Jak z názvu vyplývá, báze pravidel obsahuje soubor pravidel a ty by měla reprezentovat znalosti nutné k úspěšné regulaci systému. Abychom mohli bázi pravidel naplnit, je potřeba definovat následující parametry:

- vstupní veličiny, reprezentující stav systému – jazykové proměnné,
- hodnoty jazykových proměnných (*NB, NM, NS,* apod.),
- definovat všechny pravidla (*if-then*).

3.10.2 Typy fuzzy regulátorů

Nejčastěji se při regulaci systémů pracuje s odchylkami stavových veličin, tedy jak se stav systému v určitém časovém okamžiku liší od stavu, ve kterém by se měl nacházet. Stejně jako v klasickém řízení (viz kapitola 2.3.1), lze ve fuzzy řízení použít regulátory typu P, PI, PD a PID.

Podrobněji si popíšeme fuzzy PD regulátor, který byl použit v praktické části diplomové práce. Popis ostatních typů fuzzy regulátorů je dostupný ve zdroji [11] na straně 116 a ve zdroji [12] na straně 78.

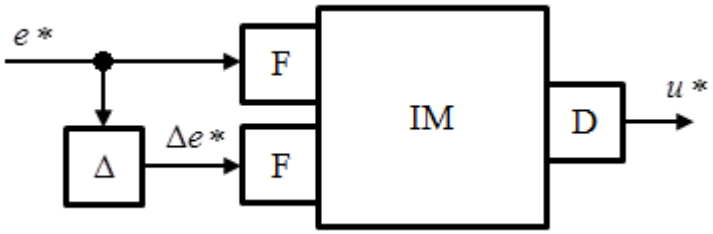
Popis klasického diskrétního PD regulátoru je dán následující rovnicí

$$u(t) = K_R e(t) + K_D \Delta e(t), \tag{3-58}$$

kde K_R je zesílení a K_D je derivační časová konstanta. Vstupy regulátoru jsou odchylka e a její změna Δe. Fuzzy pravidla pro PD regulátor mají tvar

$$if\left(e(k) = FM\right) and \left(\Delta e(k) = FM\right) then \left(u(k) = FM\right), \tag{3-59}$$

kde *FM* je jméno hodnoty jazykové proměnné. Blokové schéma fuzzy PD regulátoru je na obr. 3.18. F je označení pro fuzzifikaci, D pro defuzzifikaci a IM pro inferenční mechanizmus.



Obr. 3.18 – Schéma struktury fuzzy regulátoru

4 Praktické řešení

Principy popsané v kapitole Fuzzy řízení jsou aplikovány na model inverzního kyvadla – robot balancující na dvou motory poháněných kolech, jenž je sestaven ze stavebnice LEGO Mindstorms Education.

V této kapitole je popsána instalace potřebného softwarového vybavení, konstrukce robota, způsob komunikace, složení a nastavení regulátoru a způsob, kterým je realizován pohyb robota v prostoru.

4.1 LeJOS

LeJOS je open source firmware vytvořený pro obsluhu programovatelných kostek Lego Mindstorms. LeJOS NXJ je určen pro NXT kostky, kam byl poprvé nainstalován v roce 2006. Jedná se o programovací jazyk založený na platformě Java a jeho součástí je malá JVM (Java Virtual Machine), díky které je možné programovat aplikace v Javě. LeJOS NXJ obsahuje mnoho tříd a metod, které umožňují při práci se stavebnicí velkou variabilitu. Jejich vlastnosti jsou přehledně popsány v API (Application Programming Interface – rozhraní pro tvorbu aplikací). Po instalaci jsou navíc dostupné i různé nástroje pro kompilaci, nahrávání či ladění programů, ale také řada vzorových aplikací. Ty jsou dostupné i na webových stránkách leJOS [14].

LeJOS NXJ nemá vlastní vývojové prostředí, ale existuje programový doplněk do předních Java IDE (Integrated Development Environment) a to do NetBeans a Eclipse. Rozšiřující informace o leJOS NXJ jsou na webových stránkách leJOS [15].

Návod na instalaci

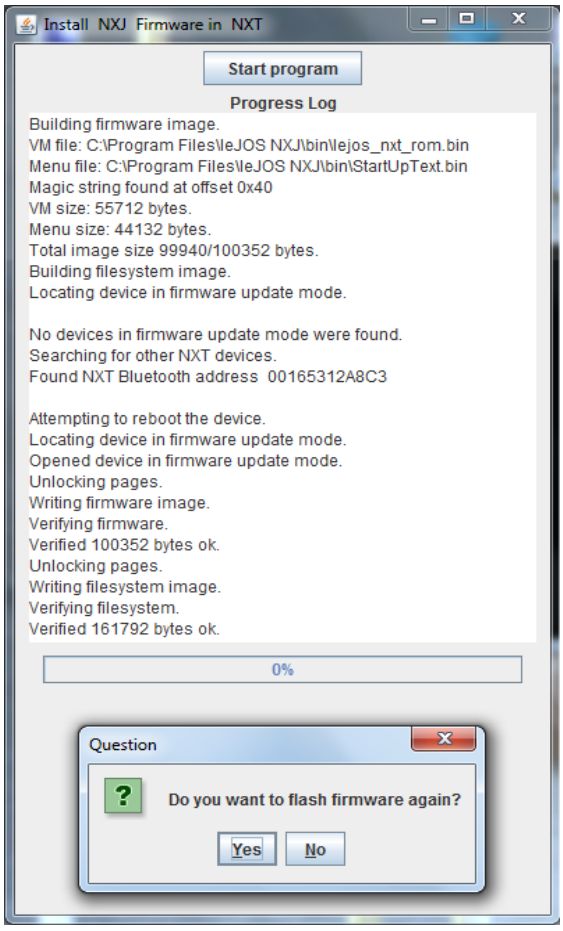
Před instalací firmwaru leJOS je potřeba nainstalovat ovladač na připojení NXJ přes USB rozhraní (Fantom Driver). Ten je obvykle dodáván se softwarem LEGO Mindstorms nebo jej lze stáhnout z webových stránek [16].

Dále je potřeba mít nainstalovaný Java Development Kit (JDK), jeho nejnovější verze je dostupná na webových stránkách [17] nebo bývá součástí instalačního balíku vývojového prostředí NetBeans. Po instalaci JDK následuje kontrola správnosti nastavení proměnných prostředí počítače. Tento krok zajistí, že další programy „uvidí“, že je JDK již nainstalován. Ve Windows lze proměnné prostředí editovat z *Ovládací panely* → *Systém* → *Změnit nastavení* → *Upřesnit* → *Proměnné prostředí*. Nastavení by mělo odpovídat hodnotám v tab. 2.

Tab. 2 – Nastavení proměnných prostředí

Proměnná	Hodnota	Příklad
JAVA_HOME	Cesta k adresáři, kde je nainstalován JDK	C:\Program Files\Java\jdk1.7.0_17
PATH	Cesta k adresáři *bin*, nainstalovaného JDK	C:\Program Files\leJOS NXJ\bin;
NXJ_HOME	Cesta k adresáři, kde je nainstalován leJOS	C:\Program Files\leJOS NXJ

Nyní můžeme spustit instalaci softwaru leJOS, který je dostupný na webových stránkách výrobce [18]. Během instalace se řídíme pokyny instalačního průvodce. Po dokončení instalace se otevře nové okno, které zajistí nahrání firmwaru do NXT kostky robota. Toto okno můžeme vyvolat i z příkazové řádky příkazem *nxjflashg*. Průběh instalace je opět velice intuitivní. Stav instalačního průvodce po úspěšném flashnutí firmwaru na NXT kostku je zobrazen na obr. 4.1.



Obr. 4.1 – Instalační průvodce po flashnutí firmwaru

Průběhy jednotlivých instalací i potřebná nastavení, jsou podrobně popsány na stránkách leJOS [19].

Jelikož po instalaci nejaktuálnější verze firmwaru (0.9.1beta) chybí adresář, ve kterém je plugin pro vývojové prostředí nebo Java dokumentace, je praktická část realizována ve verzi „0.9.0beta“

4.2 Nastavení vývojového prostředí NetBeans

Praktická část diplomové práce byla vyvíjena ve vývojovém prostředí NetBeans IDE 7.3. NetBeans představují komplexní, open source, nástroj pro tvorbu, kompilaci a správu projektů. Jeho instalační software je dostupný na webových stránkách [20] a samotná instalace opět probíhá dle pokynů průvodce instalací.

Abychom mohli v NetBeans používat třídy leJOS NXJ, dokumentaci (Javadoc) nebo jednoduše uploadovat vytvořený program na NXT kostku, je potřeba provést několik kroků.

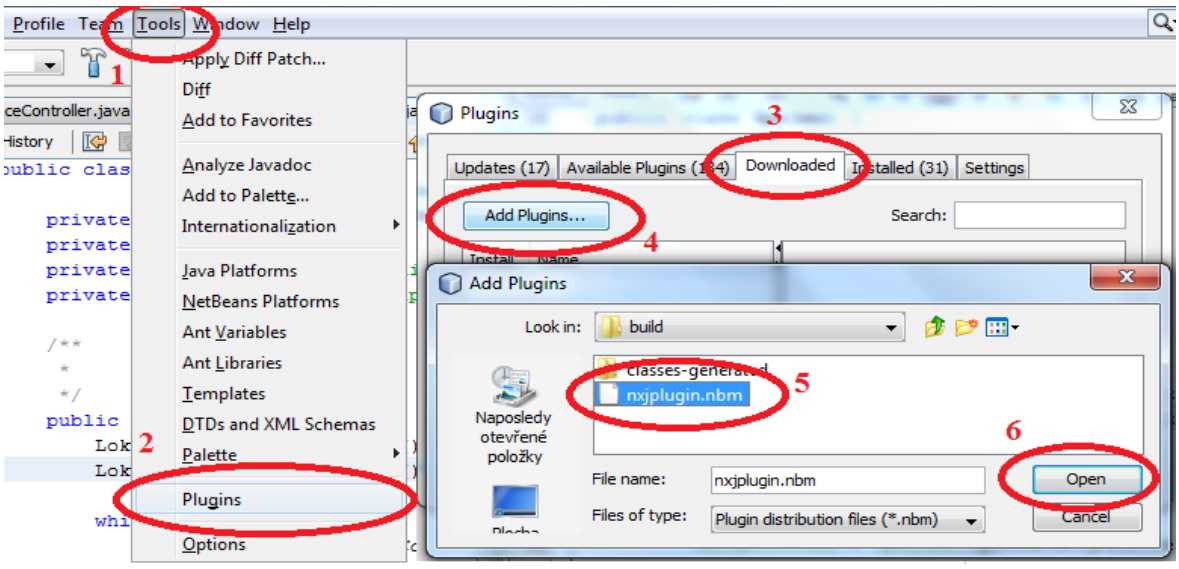
Jako první je potřeba do NetBeans nahrát NXJ plugin, ten najdeme v adresářové struktuře defaultního leJOS NXJ adresáře – *leJOSNXJProjects*. Jeho umístění bylo možné nastavit v jednom z kroků instalace softwaru leJOS NXJ. Pokud jsme při instalaci zachovali výchozí nastavení, je adresář umístěn v

$$C:\backslash Users\backslash \text{“}Uzivatel\text{“},$$

kde „*Uzivatel*“ představuje jméno uživatele, pod kterým byl software nainstalován. Cesta k pluginu je pak

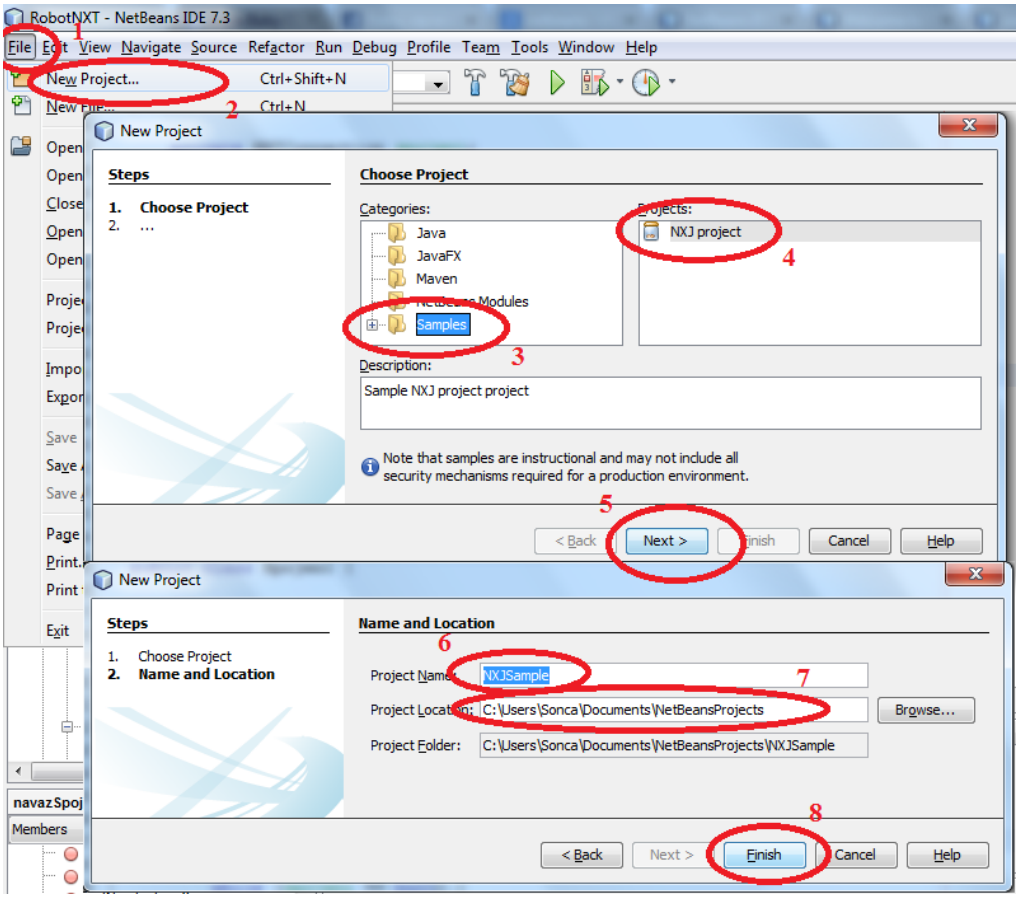
$$leJOSNXJProjects \backslash NXJPlugin \backslash build \backslash nxjplugin.nbm.$$

Do vývojového prostředí NetBeans plugin nainstalujeme následujícím způsobem. V menu klikneme na položku *Tools* → Plugins → Downloaded → Add Plugins → a zde zadáme cestu k *nxjplugin.nbm*, viz obr. 4.2.



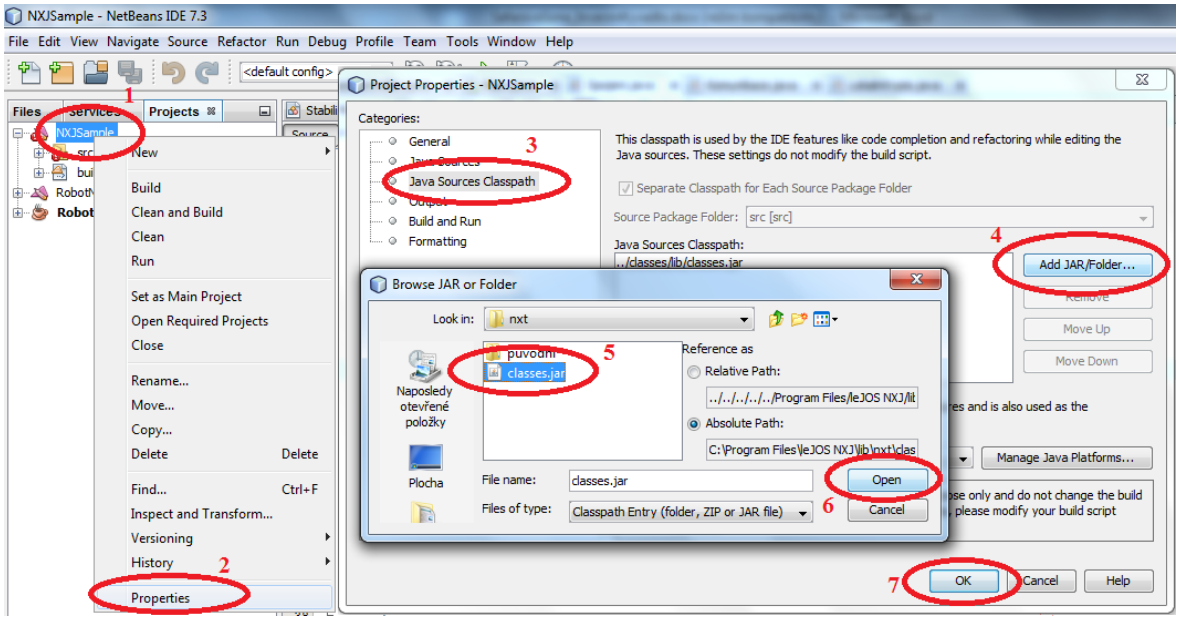
Obr. 4.2 – Postup při instalace NXJ pluginu v NetBeans

Nyní je možné v NetBeans vytvářet NXJ projekty. Postup je *File → New Project → Samples → NXJ Project → Next → zvolíme název a místo uložení projektu → Finish*.



Obr. 4.3 – Postup pro vytvoření nového NXJ projektu

Abychom mohli pracovat s leJOS třídami a metodami, je potřeba do nově vytvořeného NXJ projektu zadat cestu k leJOS balíčkům. Klikneme pravým tlačítkem myši na název projektu a postupujeme *Properties → Java Sources Classpath → Add JAR/Folder → classes.jar*. Soubor *classes.jar* nalezneme v adresáři *lib*, v místě, kam jsme nainstalovali software leJOS NXJ. Obvykle to je *C:\Program Files\leJOS NXJ\lib\nxt*. Postup můžeme vidět na obr. 4.4.



Obr. 4.4 – Postup pro přidání leJOS balíčku

Další věc, kterou je potřeba nastavit je cesta k leJOS balíčku pro *build.xml*, který zajišťuje nahrání projektu do NXT kostky. Otevřeme si adresář, kam jsme projekt uložili a otevřeme i adresář našeho projektu. Zde se nachází soubor *build.properties*, který otevřeme např. v programu *Word Pad* nebo *PSPad* (není součástí Windows). Pokud jsme v kapitole 4.1 správně nastavili proměnné prostředí, bude na prvním řádku v souboru napsáno

$$nxj.home = \$\{env.NXJ\_HOME\}.$$

V tomto případě není potřeba nic měnit a práci se souborem můžeme ukončit. Pokud ale obsah prvního řádku souboru vypadá následovně

$$nxj.home = ../snapshot,$$

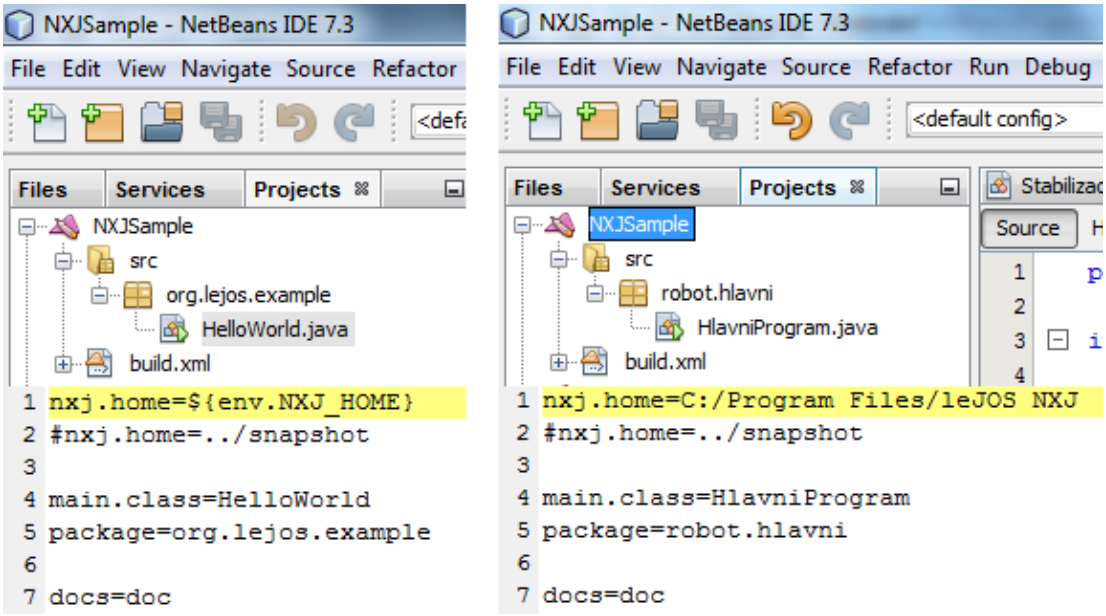
je potřeba část za rovnítkem nahradit cestou k adresáři leJOS NXJ, která bude místo zpětných lomítek obsahovat lomítka klasická (/). Takže výsledná podoba prvního řádku bude např.

$$nxj.home = C:/Program\ Files/leJOS\ NXJ.$$

Před uzavřením souboru je důležité se ujistit, že měněný řádek nekončí mezerou.

Vzhledem k tomu, že hlavní třída projektu se jmenuje *HelloWorld*, je pravděpodobné, že ji budeme chtít přejmenovat. V prostředí NetBeans třídu přejmenujeme kliknutím pravým tlačítkem myši na soubor *HelloWorld.java* a zvolíme *Refactor* → *Rename*. Nový název je potřeba zapsat do souboru *build.properties*. To stejné platí, i pokud budeme měnit umístění hlavní třídy v projektu. Názorná ukázka je na obr. 4.5, kde

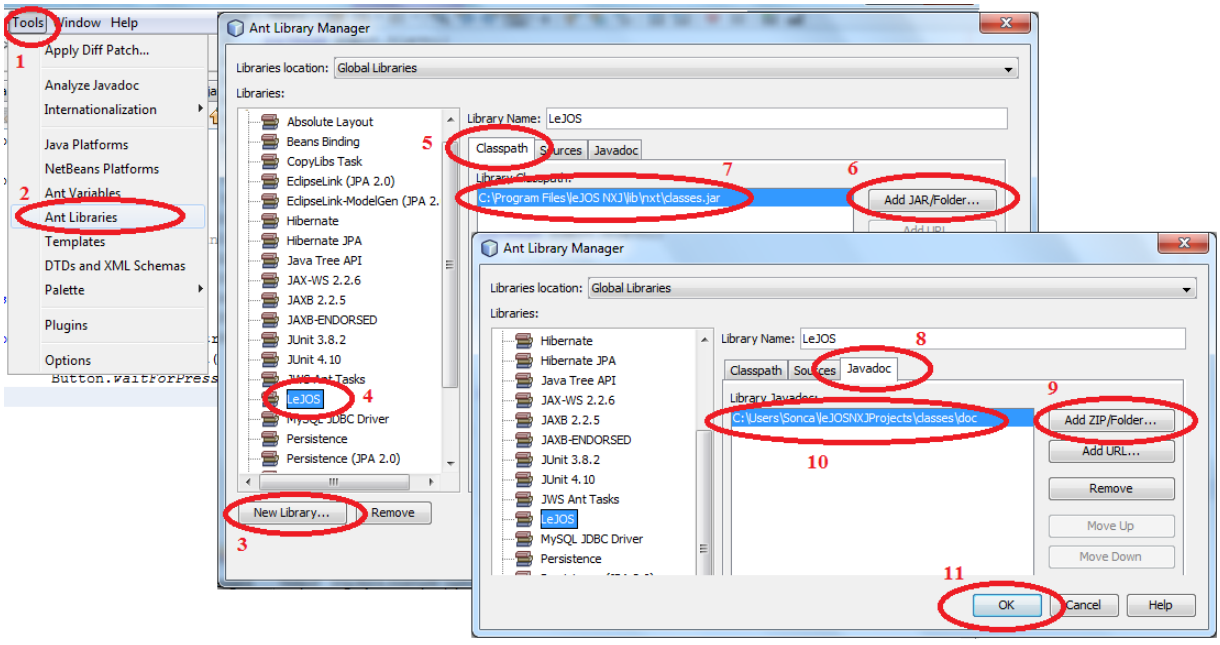
můžeme vidět stav adresářové struktury projektu a souboru *build.properties* před změnou a po změně.



Obr. 4.5 – Adresářová struktura projektu a soubor *build.properties* před (vlevo) a po změně (vpravo)

Poslední věc, kterou si nastavíme, je dokumentace JavaDoc, která nám bude pomáhat při psaní zdrojového kódu. Klikneme na položku hlavního menu *Tools* → *Ant Libraries* → *New Library* a knihovnu pojmenujeme *LeJOS*. Do sekce *Classpath* přidáme stejný balíček (*classes.jar*), jako jsme přidali do projektu, viz obr. 4.4. Do sekce *Javadoc* přidáme cestu k adresáři *doc*,

$$C:\backslash Users\backslash Sonca\backslash leJOSNXJProjects\backslash classes\backslash doc.$$



Obr. 4.6 – Přidání dokumentace

Nyní máme vše nastaveno. Nahrání projektu do NXT kostky se provede tak, že klikneme pravým tlačítkem myši na projekt a vybereme položku *Run*. Předpokladem úspěšného nahrání projektu je, že máme počítač a NXT kostku propojeny USB kabelem a NXT kostka je aktivovaná. Aktivace se provede po stisknutí oranžového tlačítka.

4.3 Konstrukce LEGO robota

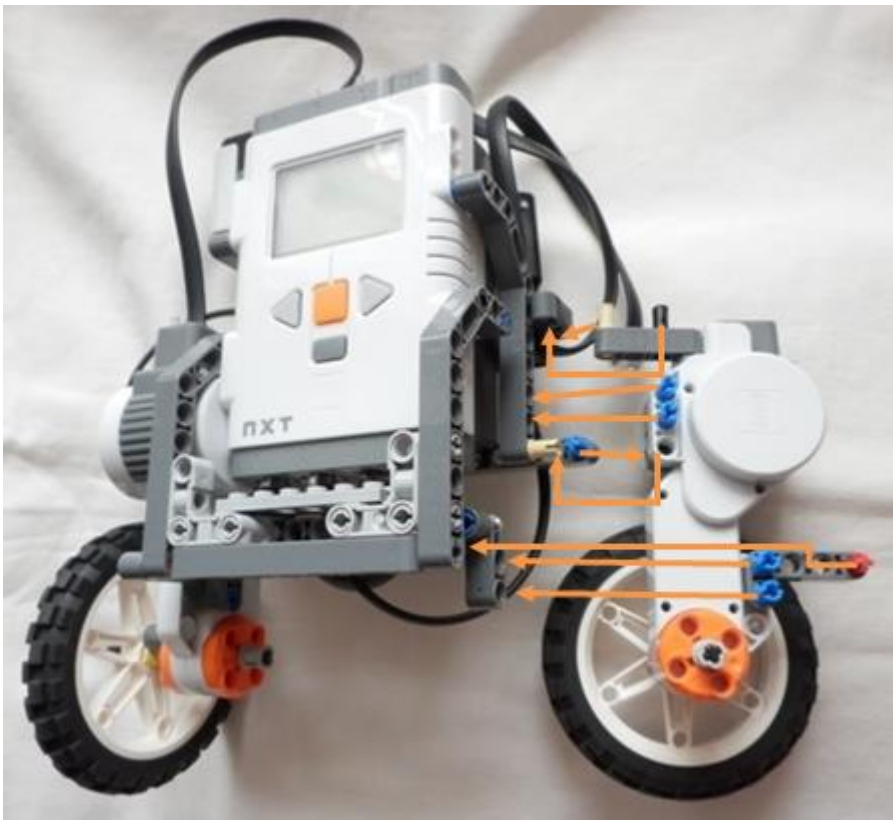
Konstrukce LEGO robota je zachycena na několika následujících obrázcích. Při sestavení odlišného modelu robota a při využití regulátoru popsaného v kapitole 4.5, je důležité zachovat pozici a umístění gyroskopického senzoru. Pokud by byl gyroskop umístěn např. na opačné straně robota, naměřené hodnoty ty byly s opačným znaménkem a regulace by byla neúspěšná.

Obr. 4.7 – Robot čelní pohled

Obr. 4.8 – Robot zezadu

Obr. 4.9 – Pravá strana robota

Obr. 4.10 – Uchycení gyroskopu



Obr. 4.11 – Připevnění motoru k NXT kostce

4.4 Komunikace robota s PC

V zadání diplomové práce je požadováno zajistit komunikaci robota a počítače prostřednictvím technologie bluetooth, jež je blíže popsaná v kapitole 4.4.1. Z technických důvodů nebyl tento požadavek dodržen. Důvodem je velký vzorkovací interval, za který se provedou následující operace:

- naměření a výpočet vstupních veličin,
- odeslání vstupů aplikaci spuštěné na PC,
- výpočet akčního zásahu fuzzy regulátorem v operační paměti počítače,
- odeslání výstupu regulátoru na NXT kostku robota,
- aplikace akčního zásahu na motory robota.

Průměrná hodnota vzorkovacího intervalu, při provedení 100 regulačních cyklů, dosahuje 0,0625 sekund, což je pro potřeby regulace nedostatečné.

Jako možná varianta řešení předešlého problému je využívat bluetooth komunikaci pouze pro druhotné předávání naměřených a vypočítaných dat z NXT kostky na PC, aby mohla být na PC zobrazena v okně aplikace a pro posílání instrukce zvoleného pohybu z počítače na NXT kostku. Pro realizaci této varianty je celý řídicí algoritmus převeden do programové části robota. Průměrná doba vzorkovacího intervalu pro tuto variantu je 0,0829 sekund, opět se jedná o průměr ze 100 hodnot. Při porovnání hodnoty s hodnotou z první varianty je zřejmé, že došlo ke zhoršení – prodloužení vzorkovacího intervalu a tedy ani tato varianta není vhodná pro stabilizaci inverzního kyvadla.

Z předchozích důvodů je komunikace technologií bluetooth nahrazena komunikací přes USB kabel, kde je vzorkovací interval v průměru 0,0147 sekund.

4.4.1 Bluetooth

Bluetooth je bezdrátová komunikační technologie, která byla vytvořena v roce 1994 firmou Ericsson. Umožňuje propojení dvou a více elektronických zařízení, jako např. mobilní telefon, PDA, počítač nebo i NXT kostka.

Jedná se o technologii pro komunikaci na krátké vzdálenosti, jejími výhodami jsou robustnost, nízká spotřeba a nízké náklady na realizaci. Spojení mezi dvěma zařízeními je realizováno prostřednictvím krátkého dosahu, sítěmi ad hoc, které jsou nazývány *piconety*. *Piconety* jsou voleny dynamicky a automaticky podle vzdálenosti, takže se můžeme snadno připojit kdykoliv a kdekoliv. Každé zařízení může v rámci jednoho *piconetu* komunikovat až se sedmi dalšími zařízeními a jedno zařízení může současně patřit do několika *piconetů*.

Maximální dosah signálu mezi zařízení je závislý na typu zařízení a na konkrétní implementaci výrobce zařízení. Minimální vzdálenost dosahu je stanovena na 10 metrů.

Rozšiřující informace o této technologii jsou dostupné na webových stránkách Bluetooth [21].

4.4.2 Navázání a ukončení spojení

Při navázání spojení je vytvořen mezi NXT kostkou a počítačem komunikační kanál, v němž poté prostřednictvím vstupního a výstupního datového proudu dochází k přenosu dat z jednoho zařízení na druhé. I když je v práci použito připojení USB kabelem, ukážeme si i způsob navázání spojení přes bluetooth.

Navázání USB spojení NXT:

```
NXTConnection spojeni;
while (spojeni == null) {
      spojeni = USB.waitForConnection();
}
```

Navázání USB spojení PC:

```
NXTConnector spojeni = new NXTConnector();
boolean jePripojeno = spojeni.connectTo("usb://");
```

Navázání bluetooth spojení NXT:

```
BTConnection spojeni;
while (spojeni == null) {
      spojeni = Bluetooth.waitForConnection();
}
```

Navázání bluetooth spojení PC:

```
NXTConnector spojeni = new NXTConnector();
boolean jePripojeno = spojeni.connectTo("btspp://");
```

Ve chvíli, kdy máme navázané spojení, můžeme vytvořit datové proudy.

NXT:

```
DataInputStream vstupniProud = spojeni.openDataInputStream();
DataOutputStream vystupniProud = spojeni.openDataOutputStream();
```

PC:

```
DataInputStream  vstupniProud = new
      DataInputStream(spojeni.getInputStream());
DataOutputStream vystupniProud = new
      DataOutputStream(spojeni.getOutputStream());
```

Při uzavírání spojení mezi NXT kostkou a počítačem je potřeba nejdříve zavřít datové proudy a až poté ukončit samotnou komunikaci. Postup pro ukončení spojení je pro obě zařízení stejné, rozdíly nejsou ani u jednotlivých technologií.

Ukončení spojení:

```
vstupniProud.close();
vystupniProud.close();
spojeni.close();
```

4.4.3 Příjem a odesílání dat

Při přijímání nebo odesílání dat mezi jednotlivými zařízeními se využívají datové proudy, jejichž vytvoření jsme si ukázali v kapitole 4.4.2. Pro obě zařízení je princip přenosu dat stejný, záleží jen na operaci čtení či zápis dat a na datovém typu dat.

Příjem (čtení) dat:

```
double uhel;
double uhlovaRychlost;
double vzorkovaciInterval;

vzorkovaciInterval = vstupniProud.readDouble();
uhel = vstupniProud.readDouble();
uhlovaRychlost = vstupniProud.readDouble();
```

Odeslání dat:

```
byte kodInstrukce = 1;
float vykon = 80;

vystupniProud.writeByte(this.kodInstrukce);
vystupniProud.writeFloat(vykon);
vystupniProud.flush();
```

Metoda *flush()* přinutí všechny vyrovnávací paměti, aby zapsaly svůj obsah do výstupního proudu.

4.5 Fuzzy regulátor

Jak již bylo v kapitole 3.10 popsáno, fuzzy regulátor se skládá z několika částí. Zde si popíšeme jejich praktickou realizaci.

4.5.1 Měření vstupních veličin

Pro zjištění aktuálního náklonu robota je využit gyroskopický senzor. Ten ale neměří přímo hodnotu vychýlení ve stupních, ale úhlovou rychlost ve stupních za sekundu. Pokud známe hodnotu vzorkovacího intervalu, ve kterém probíhají jednotlivé cykly regulace, lze z naměřené úhlové rychlosti vypočítat úhel náklonu ve stupních.

Položíme-li robota do vodorovné polohy a naměříme gyroskopem hodnotu úhlové rychlosti, zjistíme, že není nulová, i když by měla být. Proto je nutné, před prvním měřením, provést kalibraci gyroskopického senzoru. Při té je určena hodnota offsetu, která se v dalších výpočtech používá pro nastavení správné úhlové rychlosti. Offset je vypočítán jako průměr ze 100 naměřených hodnot, kdy rozdíl mezi maximální a minimální naměřenou hodnotou nesmí být větší než jedna. Takto vypočítaná hodnota offsetu je ale pouze počáteční a dále je nutné ji v každém regulačním cyklu upravovat, jelikož hodnota je proměnná v závislosti na čase. Pro průběžnou úpravu offsetu je použit exponenciální klouzavý průměr.

Kalibrace gyroskopického senzoru:

```
double lastOffset = 0;

do {
      gyroSoucet = 0.0;
      gyroMin = 1000;
      gyroMax = -1000;
      for (i = 0; i < vzorku; i++) {
            g = gyroPort.readValue();
            if (g > gyroMax) {
                  gyroMax = g;
            }
            if (g < gyroMin) {
                  gyroMin = g;
            }
            gyroSoucet += g;
            delay(5);
      }
} while ((gyroMax - gyroMin) > 1);
lastOffset = gyroSoucet / vzorku;
```

Měření hodnoty gyroskopickým senzorem:

```
double lastOffset = 0;
final double WEIGHTOFFSET = 0.9995;

public double getUhlovaRychlost() {
	double hodnota = gyroPort.readValue();
	lastOffset = lastOffset * WEIGHTOFFSET + (1.0 - WEIGHTOFFSET) *
		hodnota;
	double angleVelocity = hodnota - lastOffset;

	return angleVelocity;
}
```

Výpočet úhlu z naměřené úhlové rychlosti:

```
public double getUhel() {
	int now = (int) System.currentTimeMillis();
	if (cyclusLoop == 0) {
		vzorkovaciInterval = 0.0055f;
		timeStart = (int) System.currentTimeMillis();
	} else {
		vzorkovaciInterval = ((int) now - timeStart) /
			(cyclusLoop * 1000.0);
	}
	cyclusLoop ++;
	uhel += getUhlovaRychlost() * vzorkovaciInterval;

	return uhel;
}
```

4.5.2 Jazykové proměnné

Vstupy fuzzy regulátoru jsou jazykové proměnné *uhel* a *uhlovaRychlost*, výstupem je jazyková proměnná *vykon*. Každá jazyková proměnná je složena ze sedmi funkcí příslušnosti (fuzzy množin), které jsou buď ve tvaru trojúhelníku, nebo lichoběžníku.

Příklad vytvoření a nastavení jazykové proměnné *uhel*:

```
LingvistickaPromenna uhel = new LingvistickaPromenna();
uhel.pridatMnozinu(new LichobeznikovaMnozina(OznaceniMnoziny.NB,
      -45, -44, -13, -8.5));
uhel.pridatMnozinu(new TrojuhelnikovaMnozina(OznaceniMnoziny.NM,
      -13, -8, -3));
uhel.pridatMnozinu(new TrojuhelnikovaMnozina(OznaceniMnoziny.NS,
      -8.5f, -4, 0));
uhel.pridatMnozinu(new TrojuhelnikovaMnozina(OznaceniMnoziny.Z,
      -3, 0,3));
uhel.pridatMnozinu(new TrojuhelnikovaMnozina(OznaceniMnoziny.PS,
      0, 4, 8.5f));
uhel.pridatMnozinu(new TrojuhelnikovaMnozina(OznaceniMnoziny.PM,
      3, 8, 13));
uhel.pridatMnozinu(new LichobeznikovaMnozina(OznaceniMnoziny.PB,
      8.5f, 13, 44, 45));
```

4.5.3 Pravidla

V praktické části byl použit Mamdaniho model, viz kapitola 3.6.1. Pravidla jsou tedy ve formátu

$$if\ \big((uhel = hodnota)\ and\ (uhlovaRychlost = hodnota)\big)\ then\ (vykon = hodnota).$$

Příklad fuzzy pravidel:

```
FuzzyInferencniMechanismus fim;
fim.pridejPravidlo(OznaceniMnoziny.NB, OznaceniMnoziny.NB,
      OznaceniMnoziny.NB);
fim.pridejPravidlo(OznaceniMnoziny.NB, OznaceniMnoziny.NM,
      OznaceniMnoziny.NB);
fim.pridejPravidlo(OznaceniMnoziny.NB, OznaceniMnoziny.NS,
      OznaceniMnoziny.NB);
fim.pridejPravidlo(OznaceniMnoziny.NB, OznaceniMnoziny.Z,
      OznaceniMnoziny.NB);
```

Celou bázi pravidel, která vychází z pravidel ve zdroji [22], na str. 193, můžeme vidět v tab. 3. Slovní označení úhlů a úhlových rychlostí v menu tabulky představují

konsekventy pravidel, které se při vyhodnocení spojují logickou spojkou *and* a průnik příslušného řádku a sloupce definuje antecedent pravidla, tedy jaký bude výstup daného pravidla. Označení množin je popsáno v kapitole 3.4.

Tab. 3 – Pravidla pro fuzzy regulátor

úhlová rychlost \ úhel	NB	NM	NS	Z	PS	PM	PB
NB	NB	NB	NB	NB	NS	PS	PB
NM	NB	NB	NM	NM	Z	PM	PB
NS	NB	NB	NM	NS	PS	PM	PB
Z	NB	NM	NS	Z	PS	PM	PB
PS	NB	NM	NS	PS	PM	PB	PB
PM	NB	NM	Z	PM	PM	PB	PB
PB	NB	NS	PS	PB	PB	PB	PB

4.5.4 Nastavení akčního zásahu

Po defuzzifikaci výsledné fuzzy množiny metodou středu plochy, viz kapitola 3.9.1, získáme hodnotu akčního zásahu, která bude aplikována na motory robota.

Stabilizace robota – balancování je řízeno regulační smyčkou, kdy výsledný výkon na motory robota je složen z akčního zásahu vypočítaného regulátorem a dalších třech korekčních členů, které zachycují aktuální stav motorů. Jejich úhel natočení, který reprezentuje, o kolik stupňů se motory otočily od výchozí nulové hodnoty (start programu), a druhý korekční člen představuje rychlost otáčení motorů ve stupních za sekundu.

Pokud chceme, aby se robot krom balancování také pohyboval v prostoru, je potřeba přidat do výpočtu výsledného výkonu ještě jeden regulační člen, ten pak zajišťuje správnost jízdy.

Následující ukázky zdrojového kódu demonstrují zjednodušené části regulační smyčky, kde dochází ke složení akčního zásahu z hodnoty vypočítané regulátorem a korekčních členů motorů.

Klasický PD regulace:

```
static final double KUHEL = 4.35;
static final double KUHLRYCHLOST = 0.6;
static final double KPOZICE = 0.07;
static final double KRYCHLOST = 0.1;
static final double KJIZDA = -0.02;
```

```
float vykon;

vykon = (float) ((KUHEL * uhel) + (KUHLRYCHLOST * uhlovaRychlost) +
      (KPOZICE * mtrPozice) + (KRYCHLOST * mtrRychlost) +
      (KJIZDA * kontrolaPohybu));
```

Fuzzy regulace:

```
static final double KPOZICE= 0.07;
static final double KRYCHLOST = 0.1;
static final double KJIZDA = -0.02;
float vykon;

vykon = (float) (vykonFuzzy + (KPOZICE * mtrPozice) + (KRYCHLOST *
      mtrRychlost) + (KJIZDA * kontrolaPohybu));
```

- *uhlovaRychlost* – hodnota naměřená gyroskopickým senzorem [°/s].
- *uhel* – náklon robota vypočítaný z úhlové rychlost [°]. Hodnota je kladná, když se robot naklání dopředu a záporná když dozadu (záleží na umístění gyroskopu).
- *mtrPozice* – pozice motorů [°], která je dána připočtením rozdílu součtu hodnot z obou snímačů motorů (*mtrSum*) a součtu hodnot snímačů z předchozího kroku (*mtrSumPred*). Touto hodnotou se zhruba udržuje startovní pozice robota.
- *mtrRychlost* – rychlost motorů [°/s], jde o průměrnou hodnotu ze čtyř posledních rozdílů (*mtrDelta*). Pokud je robot nadměrně rozkmitán, hodnota proměnné zajistí zmírnění tohoto efektu.
- *kontrolaPohybu* – reguluje správnost směru jízdy při požadavku na jízdu dopředu nebo dozadu a zajišťuje udržení stability, ve chvíli, kdy robot přechází z pohybu do stavu *STOP* (pouze balancování).

```
mtrLevy = motorC.getTachoCount();
mtrPravy = motorB.getTachoCount();
mtrSum = mtrPravy + mtrLevy;
mtrDelta = mtrSum - mtrSumPred;
mtrPozice += mtrDelta;

mtrRychlost = (float) ((mtrDelta + mtrDeltaPozice1 + mtrDeltaPozice2 +
      mtrDeltaPozice3) / (4 * vzorkovaciInterval));
```

Výchozí hodnoty jednotlivých konstant pro fuzzy regulátor byly nastaveny dle vzorového příkladu PD regulátoru na stránkách výrobce gyroskopického senzoru [23] a dále pak upravovány pro zajištění co nejlepší stabilizace.

Před aplikací akčního zásahu na motory robota si musíme dát pozor, jestli hodnota výkonu nepřesáhla interval $[-100,100]$, jelikož ten je omezující pro metodu *setPower*, kterou se výkon motorů nastavuje. Akční zásah je pak proveden, dle kladné nebo záporné hodnoty výsledného výkonu, metodami *forward* pro jízdu dopředu nebo *backward* pro jízdu dozadu.

```
NXTMotor motorB = new NXTMotor(MotorPort.B);
NXTMotor motorC = new NXTMotor(MotorPort.C);

if (vykon > 100) {
      vykon = 100;
}
if (vykon < -100) {
      vykon = -100;
}
motorB.setPower((int) Math.abs(vykon));
motorC.setPower((int) Math.abs(vykon));

if (vykon > 0) {
      motorB.forward();
      motorC.forward();
} else {
      motorB.backward();
      motorC.backward();
}
```

4.6 Pohyb robota

Pohyb robota v prostoru je v případě jízdy dopředu a dozadu zajištěn úpravou proměnné *mtrPozice*, která určuje aktuální pozici motorů. Jelikož je proměnná *mtrPozice* součástí jednoho regulačního členu je potřeba provést její úpravu před skládáním akčního zásahu z hodnoty vypočítané fuzzy regulátorem a regulačních členů.

Otáčení robota doprava a doleva docílíme úpravou výsledného akčního zásahu.

Pohyb dopředu a dozadu:

```
if (instrukce == DOPREDU) {
      kontrolaPohybu = KDOPREDU;
      mtrPozice -= kontrolaPohybu * vzorkovaciInterval;
} else if (instrukce == DOZADU) {
      kontrolaPohybu = KDOZADU;
      mtrPozice -= kontrolaPohybu * vzorkovaciInterval;
} else {
      kontrolaPohybu = 0;
}

vykon = (float) (vykonFuzzy + (KPOZICE * mtrPozice) + (KRYCHLOST *
      mtrRychlost) + (KJIZDA * kontrolaPohybu));
```

Otáčení robota doprava:

```
motorB.setPower((int) Math.abs(vykon - 10));
motorC.setPower((int) Math.abs(vykon + 10));
```

Otáčení robota doleva:

```
motorB.setPower((int) Math.abs(vykon + 10));
motorC.setPower((int) Math.abs(vykon - 10));
```

V rámci objevování možností, které stavebnice LEGO Mindstorms Education nabízí, byla na model robota připevněna dioda, která v průběhu stabilizace robota bliká.

```
RCXLightSensor dioda = new RCXLightSensor(SensorPort.S2);

Thread diodaBlikani = new Thread(new Runnable() {
      public void run() {
            while (true) {
                  dioda.activate();
                  delay(300);
                  dioda.passivate();
                  delay(100);
            }
      }
});
```

Důkazem úspěšnosti stabilizace robota fuzzy regulátorem je následující obrázek, obr. 4.12.

Obr. 4.12 – Důkaz úspěšné fuzzy regulace

5 Vyhodnocení regulačního pochodu

V této kapitole jsou vyhodnoceny výsledky kvality regulace systému inverzního kyvadla. Hlavním cílem diplomové práce bylo stabilizovat LEGO robota s využitím fuzzy regulátoru, což bylo úspěšné.

Abychom mohli zhodnotit výsledky regulace, jsou v průběhu stabilizace ukládány data o stavu robota. V jednotlivých průchodech regulačního cyklu se do souboru ukládá číselná hodnota vzorkovacího intervalu, úhel vychýlení, úhlová rychlost náklonu a výkonu motorů, který představuje akční zásah.

Výsledky ze systému, který je řízen fuzzy regulátorem, jsou porovnány s výsledky systému, kde je použit klasický PD regulátor. Zdrojový kód PD regulátoru, jež je aplikovatelný na stavebnici LEGO Mindstorms Education, je uveřejněn na webových stránkách výrobce gyroskopického senzoru [23].

Kritérium kvality regulace je definováno jako absolutní integrační plocha

$$I = \int_0^{t_{end}} |e(t)| \, \mathrm{d}t \,, \tag{5-1}$$

což znamená, že plocha je vymezená rozdílem skutečného a požadovaného úhlu inverzního kyvadla. Proměnná $e(t)$ je regulační odchylka.

Kritérium je třeba vyhodnotit pro stejný časový úsek jak pro fuzzy regulátor, tak pro PD regulátor. Data zobrazená v grafech na obr. 5.1 a obr. 5.2 jsou vyhodnocena pro zvolené kritérium v časovém intervalu [0,10] sekund. Na ose

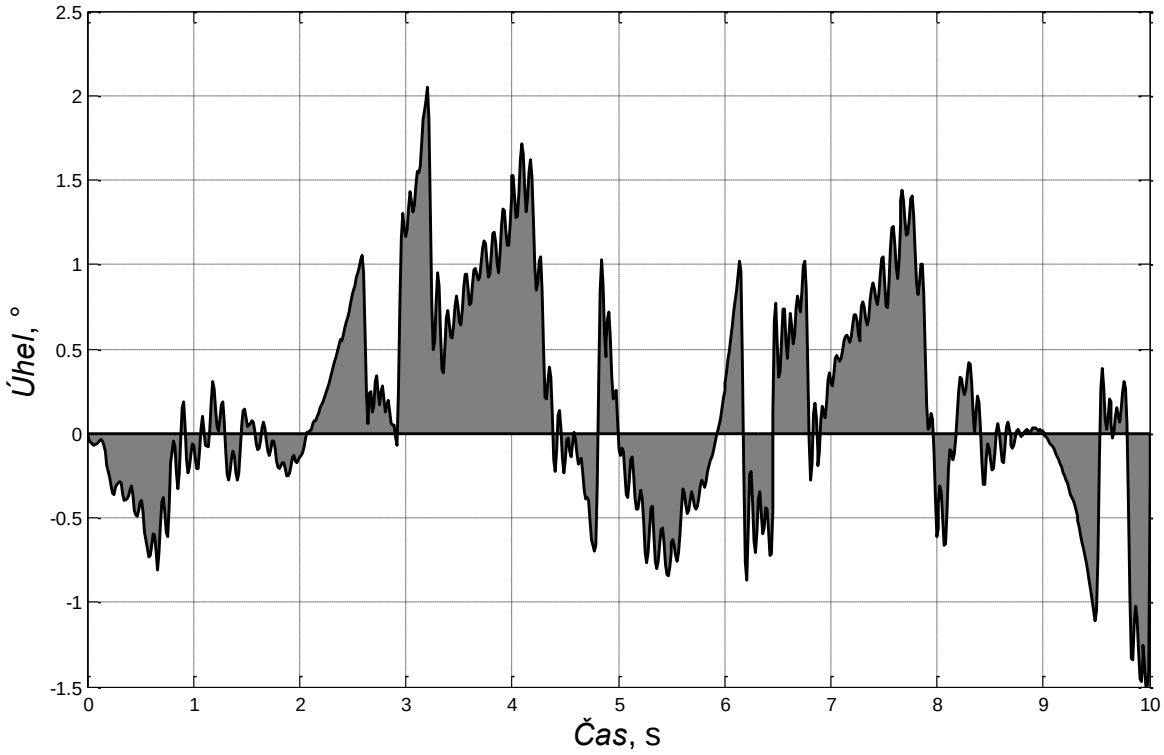
V numerickém výpočtu určitého integrálu je využita obdélníková metoda

$$I = \sum_{i=0}^{n-1} |e_i| \cdot (t_i - t_{i-1}) \,. \tag{5-2}$$

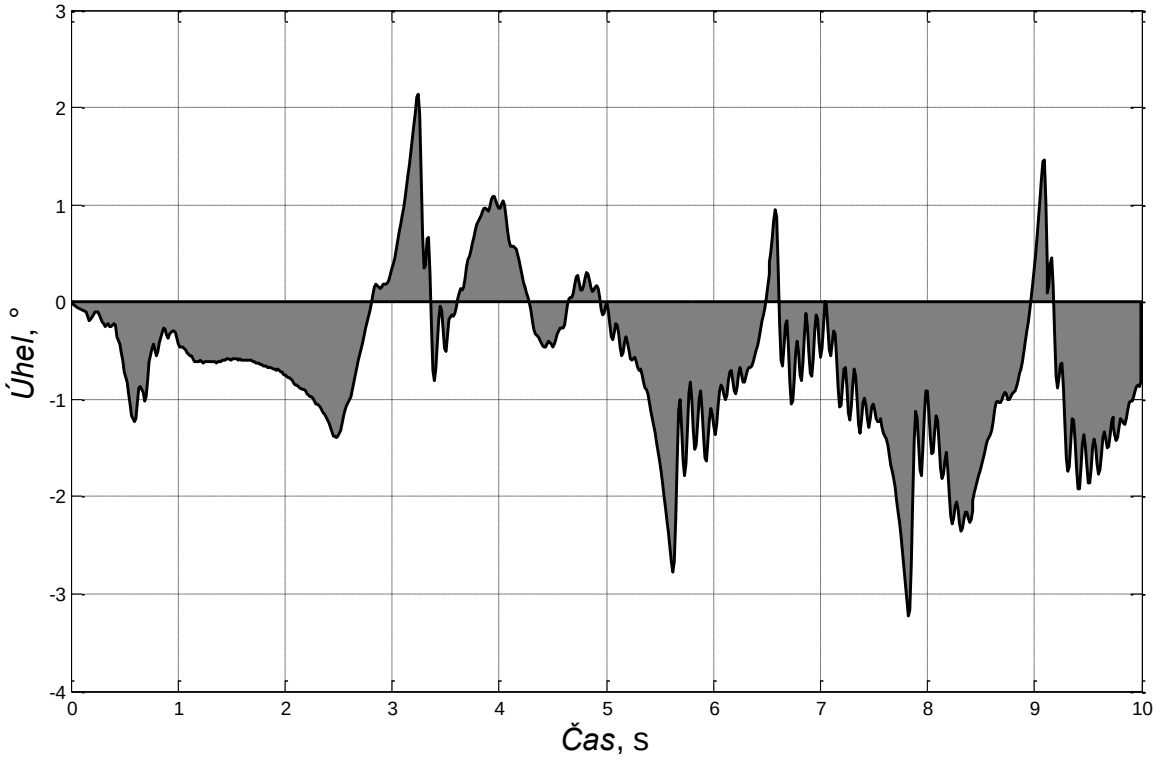
Z naměřených dat jsme výpočtem, dle předchozího vzorce, dostali následující výsledky (obsahy integračních ploch):

- $I_{fuzzy} = 4{,}8825$,
- $I_{PD} = 8{,}512$.

Je tedy zřejmé, že fuzzy řízení dosahuje při stabilizaci inverzního kyvadla lepších výsledků než klasická regulace s PD regulátorem. V tomto případě je fuzzy řízení lepší o 42,64 %.



Obr. 5.1 – Integrační plocha regulačního pochodu fuzzy regulátoru



Obr. 5.2 – Integrační plocha regulačního pochodu PD regulátoru

Závěr

Diplomová práce byla zaměřena na stabilizaci inverzního kyvadla, jehož model v podobě robota balancujícího na dvou motory poháněných kolech byl sestaven ze stavebnice LEGO Mindstorms Education. Stavebnice je vhodným prostředkem k simulování reálných systémů, jelikož její hlavní součástí je programovatelná kostka NXT, ke které je možné připojit a následně tak ovládat celou řadu periferií. Těmi jsou motory a různé druhy senzorů, např. ultrazvukový, světelný, dotykový nebo gyroskopický senzor.

Stabilizace inverzního kyvadla byla, dle zadání diplomové práce, realizována fuzzy řízením. Aplikace fuzzy regulátoru na systém robota byla úspěšná a kvalita regulace je o 42,64 % vyšší než při stabilizaci robota klasickým PD regulátorem. Ovšem nevýhodou fuzzy regulace oproti klasické regulaci je její vyšší výpočetní náročnost a tudíž je prakticky nepoužitelná se současným hardwarovým vybavením NXT kostky. Průměrný vzorkovací interval jedné regulační smyčky při výpočtu fuzzy regulace na kostce NXT je 0,0829 sekund. Při přemístění regulačního výpočtu do aplikace, jež je spouštěna na počítači a s využitím bluetooth komunikace pro posílání dat mezi zařízeními, klesne průměrná hodnota vzorkovacího intervalu na 0,0625 sekund. I když došlo ke zlepšení o více jak 24 %, jsou obě hodnoty pro potřeby regulace nedostatečné. Z tohoto důvodu musela být komunikace postavená na bluetooth technologii nahrazena komunikací přes USB kabel, kde průměrná hodnota vzorkovacího intervalu je 0,0147 sekund.

Na podzim roku 2013 má být na trh uvedena nová verze stavebnice s označením LEGO Mindstorms EV3, která má mít oproti současné verzi mnoho vylepšení. Je tedy pravděpodobné, že nový hardware si již s náročnějšími výpočty poradí a fuzzy regulace bude moct běžet v kostce NXT. Velkou výhodou bude i nová možnost komunikace prostřednictvím Wi-Fi.

Součástí diplomové práce je uživatelská příručka, která obsahuje návody, jak pracovat s vytvořenými aplikacemi. Kompletní okomentované zdrojové kódy jsou k dispozici na přiloženém CD.

Literatura

[1] NGUYEN, H., PRASAD, N., WALKER, C., WALKER, E. *A First Course in Fuzzy and Neural Control*. Boca Raton: Chapman & Hall/CRC, 2003, 301 s. ISBN 15-848-8244-1

[2] THE LEGO GROUP. *LEGO.com* [online]. ©2013 [cit. 2013-05-19]. Dostupné z: http://www.lego.com/

[3] Support. *LEGO.com MINDSTORMS* [online]. ©2012 [cit. 2013-05-11]. Dostupné z: http://mindstorms.lego.com/en-us/support/default.aspx

[4] Světelný senzor. *EDUXE: LEGO education* [online]. ©2012 [cit. 2013-05-05]. Dostupné z: http://www.eduxe.cz/product/9844-svetelny-senzor-341/

[5] NXT Sensors. *LEGO Engineering* [online]. © 2013 [cit. 2013-05-11]. Dostupné z: http://legoengineering.com/nxt-sensors-2.html

[6] NXT Gyro Sensor. *HiTechnic Products* [online]. © 2001-2012, [cit. 2013-05-05]. Dostupné z: http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NGY1044

[7] Co znamená PID. *Automa: časopis pro automatizační techniku* [online]. Praha: FCC Public, 2003, č. 03 [cit. 2013-05-06]. ISSN 1210-9592. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=28768

[8] ZADEH, Lotfi A. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Transactions on Systems, Man, and Cybernetics* [online]. 1973, č. 01 [cit. 2013-05-09]. ISSN 0018-9472. Dostupné z: http://www.cs.berkeley.edu/~zadeh/papers/1973-Outline%20of%20a%20New%20Approach%20to%20the%20Analysis%20of%20Complex%20Systems%20and%20Decision%20Processes.pdf

[9] BALCAR, Bohuslav a Petr ŠTĚPÁNEK. *Teorie množin*. 2. oprav. a rozš. vyd. Praha: Academia, 2000, 462 s. ISBN 80-200-0470-X

[10] VOPĚNKA, Petr. *Úvod do klasické teorie množin*. 1. vyd. Plzeň: Vydavatelství Západočeské univerzity v Plzni, 2011, 205 s. ISBN 978-802-5312-513. Vydáno ve spolupráci s nakl. Fragment

[11] NOVÁK, Vilém. *Základy fuzzy modelování*. 1. vydání. Praha: BEN - technická literatura, 2000, 175 s. ISBN 80-730-0009-1

[12] JURA, Pavel. *Základy fuzzy logiky pro řízení a modelování*. Vyd. 1. Brno: VUTIUM, 2003, 132 s. ISBN 80-214-2261-0

[13] NAVARA, Mirko a Petr OLŠÁK. *Základy fuzzy množin*. Vyd. 2., přeprac. Praha: Nakladatelství ČVUT, 2007, 150 s. ISBN 978-80-01-03668-6

[14] The leJOS NXJ Tutorial. *LeJOS, Java for Lego Mindstorms* [online]. 1997-2009 [cit. 2013-05-12]. Dostupné z: http://lejos.sourceforge.net/nxt/nxj/tutorial/index.htm

[15] Introduction. *LeJOS, Java for Lego Mindstorms* [online]. 1997-2009 [cit. 2013-05-12]. Dostupné z: http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/Intro.htm

[16] Files - Drivers - Fantom Driver (PC & MAC). *LEGO.com MINDSTORMS* [online]. ©2012 [cit. 2013-05-12]. Dostupné z: http://mindstorms.lego.com/en-us/support/files/Driver.aspx/

[17] Java SE Downloads. *Oracle | Hardware and Software, Engineered to Work Together* [online]. [16.4.2013] [cit. 2013-05-12]. Dostupné z: http://www.oracle.com/technetwork/java/javase/downloads/index.html

[18] NXJ Downloads. *LeJOS, Java for Lego Mindstorms* [online]. 1997-2009 [cit. 2013-05-12]. Dostupné z: http://lejos.sourceforge.net/nxj-downloads.php

[19] Getting Started on Microsoft Windows. *Java for Lego Mindstorms* [online]. 1997-2009 [cit. 2013-05-12]. Dostupné z: http://lejos.sourceforge.net/nxt/nxj/tutorial/Preliminaries/GettingStartedWindows.htm#7

[20] NetBeans IDE 7.3 Download. *NetBeans IDE* [online]. © 2012 [cit. 2013-05-13]. Dostupné z: https://netbeans.org/downloads/

[21] *Bluetooth Technology Website* [online]. © 2013 [cit. 2013-05-14]. Dostupné z: http://www.bluetooth.com/Pages/Bluetooth-Home.aspx

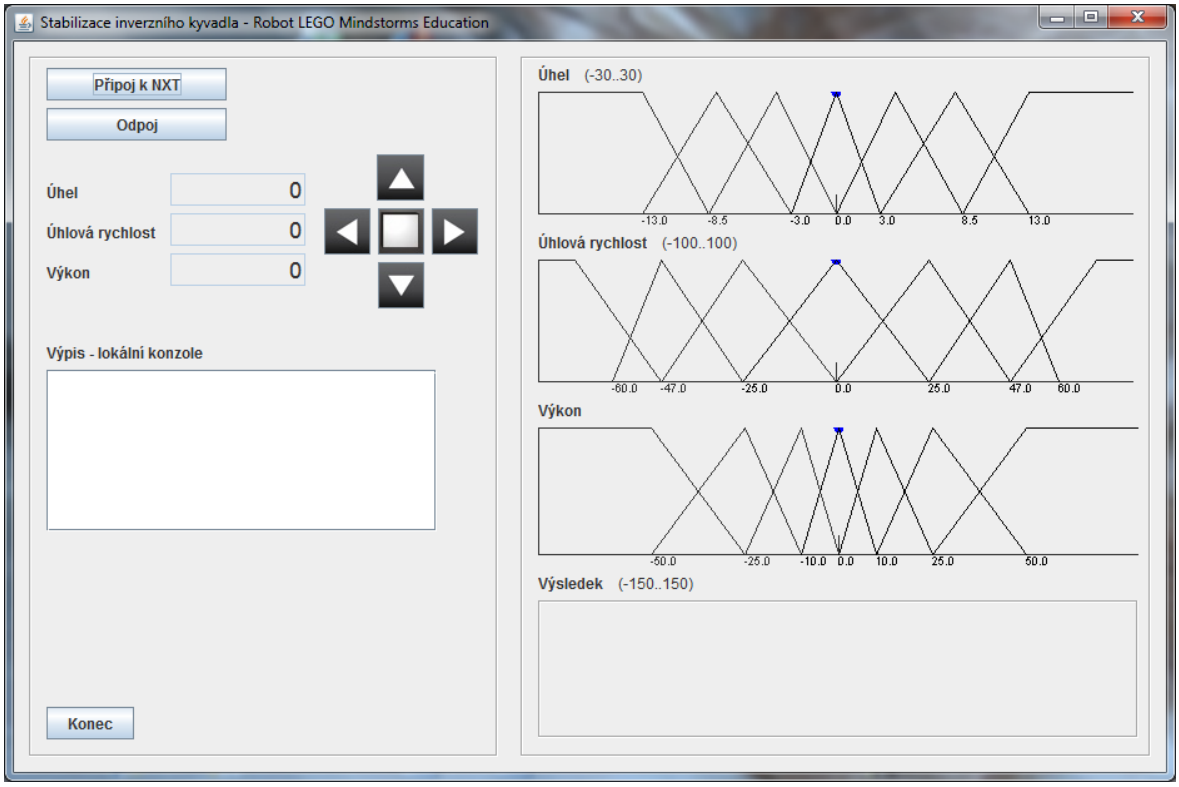
[22] PASSINO, Kevin M. a Stephen YURKOVICH. *Fuzzy control*. Menlo Park, Calif.: Addison-Wesley, 1998, xviii, 475 s. ISBN 02-011-8074-X

[23] HTWay – A Segway type robot. *HiTechnic Products* [online]. © 2001-2012 [cit. 2013-05-16]. Dostupné z: http://www.hitechnic.com/blog/gyro-sensor/htway/

Příloha A – Uživatelská příručka, část PC

Systém pro stabilizaci inverzního kyvadla sestaveného ze stavebnice LEGO Mindstorms Education se skládá ze dvou programových částí. První část je aplikace uložená a spuštěná na stolním počítači nebo notebooku (dále označovaná jako „Robot PC“), druhá část je aplikace uložená a spuštěná v NXT kostce robota („Robot NXT“).

Po spuštění „Robot PC“ se otevře okno aplikace, které můžeme vidět na následujícím obrázku.



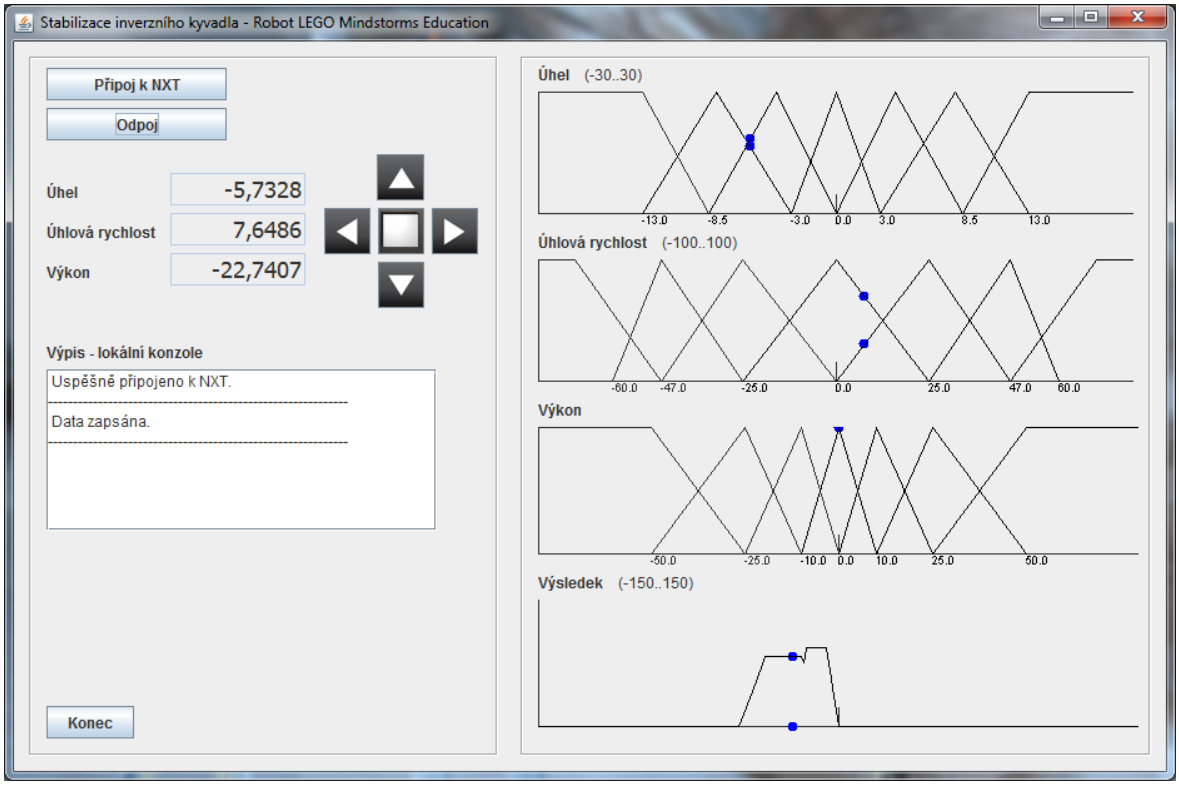
Obr. A.1 – Layout programové části „aplikace PC"

Okno aplikace lze pomyslně rozdělit na dvě části, na levou, kde se nachází ovládací prvky programu (tlačítka), textová pole pro výpis hodnot vstupních a výstupních veličin a textové pole pro informativní výpisy. V pravé části okna aplikace je zobrazeno rozložení jednotlivých jazykových proměnných (vstupy a výstup regulačního systému), ve kterých dochází ke grafickému vykreslování procesů fuzzifikace a defuzzifikace.

Po stisknutí tlačítka „Připoj k NXT“ a v případě, že je robot propojený s počítačem USB kabelem a „Robot NXT“ na kostce již běží, dojde k navázání spojení a spustí se regulační výpočet. Informace o spojení se objeví v textovém poli „Výpis – lokální konzole“, kde po určitém časovém intervalu přibyde informace o úspěšném či neúspěšném zapsání a uložení naměřených a vypočítaných dat do souboru. Hodnoty v textových polích vstupních veličin regulátoru *úhel* a *úhlová rychlost* jsou měněny dle aktuálně přijatých dat, jež byla naměřena gyroskopickým senzorem připevněným na robotovi. Data jsou

předávána na základě vytvořené komunikace mezi „Robot PC“ a „Robot NXT“. Hodnota akčního zásahu *výkon* je aktualizována dle hodnoty vypočtené fuzzy regulátorem.

Současně dochází k překreslování hodnot v pravé části okna, kde můžeme sledovat, které fuzzy množiny jsou vstupními veličinami aktivovány, jak vypadá tvar výsledné fuzzy množiny a kde se nachází z ní defuzzifikovaná hodnota akčního zásahu. Všechny skutečnosti popsané v tomto odstavci můžeme vidět na obr. A.2.

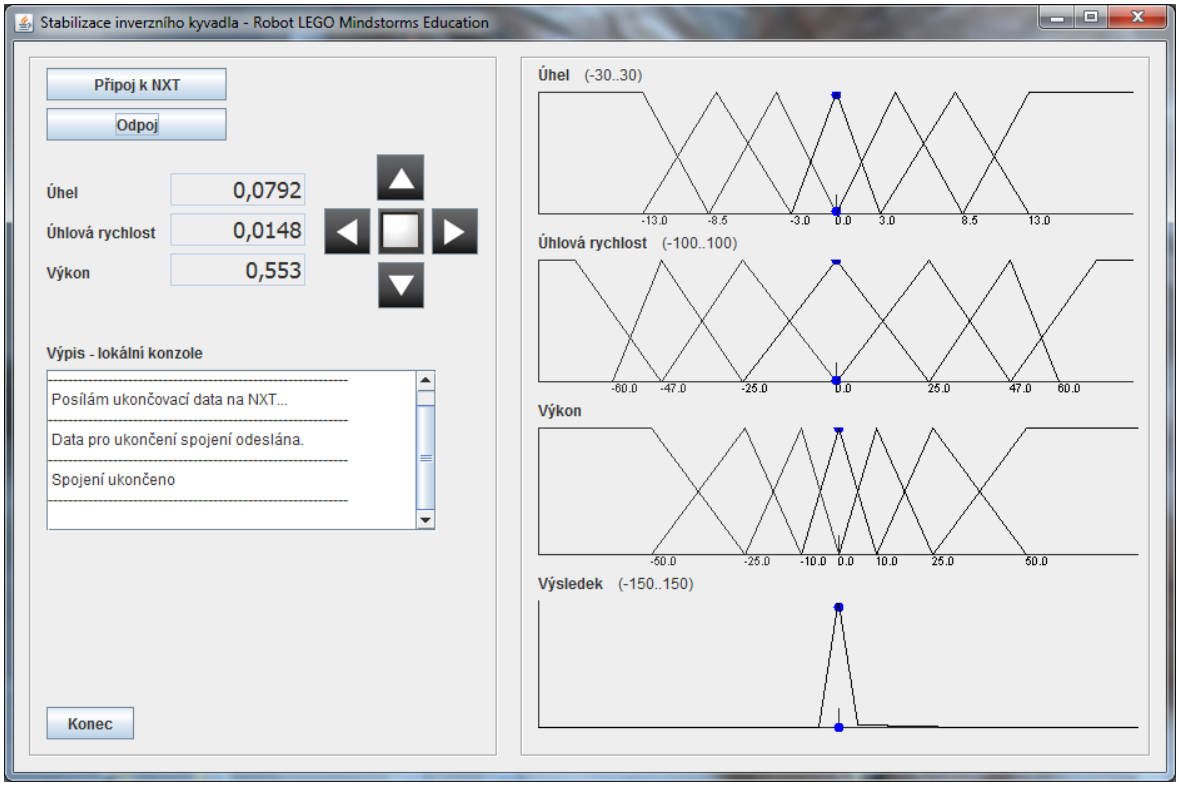


Obr. A.2 – Okno aplikace po spuštění regulace

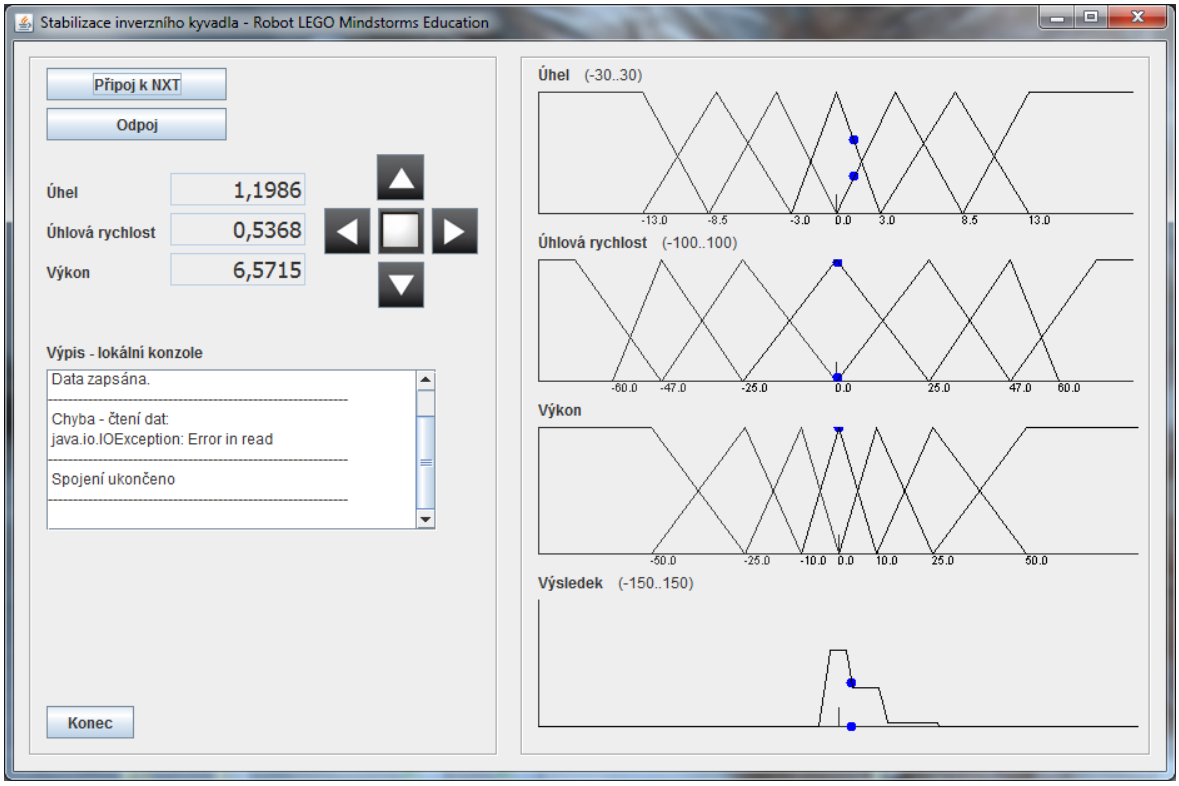
V levé části okna aplikace jsou umístěna i tlačítka, která slouží k ovládání pohybu robota v prostoru. Robot je schopný jet dopředu, dozadu a otáčet se doleva a doprava. Pokud je robot v jakémkoliv z uvedených pohybů a je zmáčknuto tlačítko stop, pohyb robota je ukončen a dále je prováděno pouze balancování na místě.

Pokud chceme práci s robotem ukončit, zmáčkneme tlačítko „Odpoj“, které zajistí ukončení spojení mezi počítačem a robotem. O úspěšném ukončení spojení jsme opět informování v textovém poli „Výpis – lokální konzole“, jak můžeme vidět na obr. A.3. Pokud během provádění regulace dojde k výpadku navázaného spojení, nebo jsme úmyslně ukončili „Robot NXT“, dojde k automatickému zastavení regulačního výpočtu a k vypsání informace o této skutečnosti, viz obr. A.4.

Po ukončení spojení je možné navázat nové spojení, aniž by bylo potřeba aplikaci „Robot PC“ restartovat.

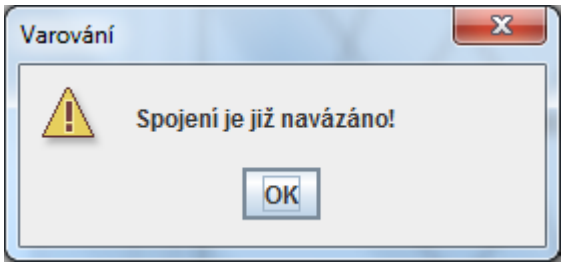


Obr. A.3 – Regulérní ukončení spojení

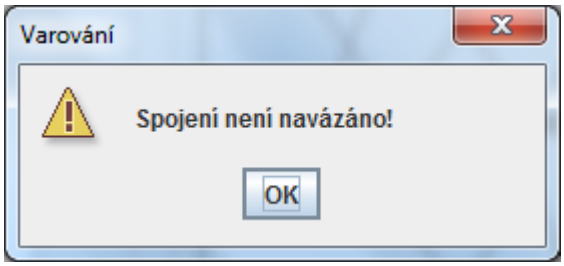


Obr. A.4 – Ukončení spojení při chybě

Pokud chceme navázat mezi aplikacemi „Robot PC“ a „Robot NXT“ nové spojení, je nutné nejdříve ukončit to současné. V opačném případě jsme varování, že spojení je stále navázáno, jak můžeme vidět na obr. A.5. To stejné platí i ve chvíli, kdy již došlo k ukončení spojení a my se pokusíme o jeho znovu ukončení, obr. A.6.



Obr. A.5 – Varování o již navázaném spojení



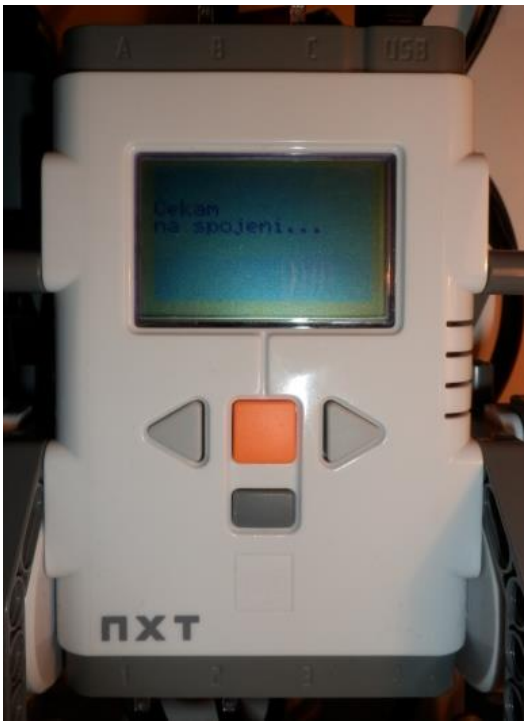
Obr. A.6 – Varování o nenavázané komunikaci

Program „Robot PC“ lze ukončit dvěma způsoby, buď tlačítkem „křížek“ v pravém horním rohu okna aplikace nebo tlačítkem „Konec“, které se nachází v levém dolním rohu okna aplikace.

Příloha B – Uživatelská příručka, část NXT

Jak již bylo řečeno, program, který zajišťuje stabilizaci robota, se skládá ze dvou částí. Jedna část je spuštěna na počítači, její ovládání a grafické prvky jsme si popsali v Příloze A a druhá část, která bude popsána zde, je spuštěna na NXT kostce robota, kam byla nahrána z vývojového prostředí NetBeans.

Programovatelná kostka NXT se aktivuje po stisknutí oranžového tlačítka, které bude dále označováno jako *ENTER*. Toto tlačítko slouží i jako aktivační prvek při listování v položkách menu nebo při spouštění programů, které jsou v kostce nahrány. V okolí tlačítka *ENTER* se nacházejí další tři tlačítka, viz obr. B.1, šipky doleva a doprava umožňující listování v menu kostky a tmavě šedé tlačítko, které slouží k návratu z vnořených nabídek menu, k ukončení aktuálně běžícího programu nebo k vypnutí kostky. Toto tlačítko bude dále označováno jako *STOP*.



Obr. B.1 – Programovatelná kostka NXT

V okamžiku, kdy máme kostku aktivní, nalistujeme si v menu položku „Files“ a tlačítkem *ENTER* do ní vstoupíme. V této složce se nachází seznam souborů, které jsou uloženy v paměti kostky. Pokud je v kostce uloženo více programů, vybereme si šipkami ten, se kterým chceme pracovat. V našem případě to je „Main.nxj“ a stiskneme *ENTER*. Objeví se nám další nabídky, kterými můžeme námi zvolený program spustit (Execute program), vymazat (Delete file) nebo nastavit jako výchozí (Set as Default). Tato funkcionalita je velmi užitečná, pokud s některým programem pracujeme často. Po aktivací kostky pak stačí pouze potvrdit aktivní položku menu – „Run Default“ a program je spuštěn.

Po spuštění programu se na display NXT kostky objeví nápis „`Cekam…`“. Aplikace „Robot NXT“ je tedy připravena navázat spojení s „Robot PC“. V okamžiku, kdy je navázáno spojení, se na display objeví nápis „`Pripojeno…`“ a hned poté je vypsána instrukce, „`Pro kalibraci gyra stisknete ENTER`“, spolu s hodnotou naměřenou gyroskopem. Ta je cyklicky aktualizována až do stisknutí *ENTER*. O tom, že probíhá kalibrace gyroskopu, jsme opět informování výpisem na display. Před spuštěním kalibrace je vhodné umístit robota do stabilní vodorovné polohy.

Poslední instruktážní výpis, který se před spuštěním samotné regulace objeví na display je „`Postavte robota a zmáčkněte ENTER`“. Ve chvíli, kdy stiskneme *ENTER*, je spuštěna regulace a robot začne balancovat. Na display můžeme sledovat hodnoty veličin reprezentující aktuální stav robota:

- úhel vychýlení robota [°],
- úhlová rychlost [°/s],
- výkon aplikovaný na motory,
- součet úhlů natočení motorů [°],
- rychlost otáčení motorů [°/s],
- vzorkovací interval [s].

Ukončení programu provedeme stisknutím *STOP*.

V případě, kdy je robot hodně rozkmitán a spadne, ukončíme běh aplikace „Robot NXT“ stisknutím *STOP*. Poté je možné navázat nové spojení s aplikací „Robot PC“, aniž by ji bylo třeba restartovat.

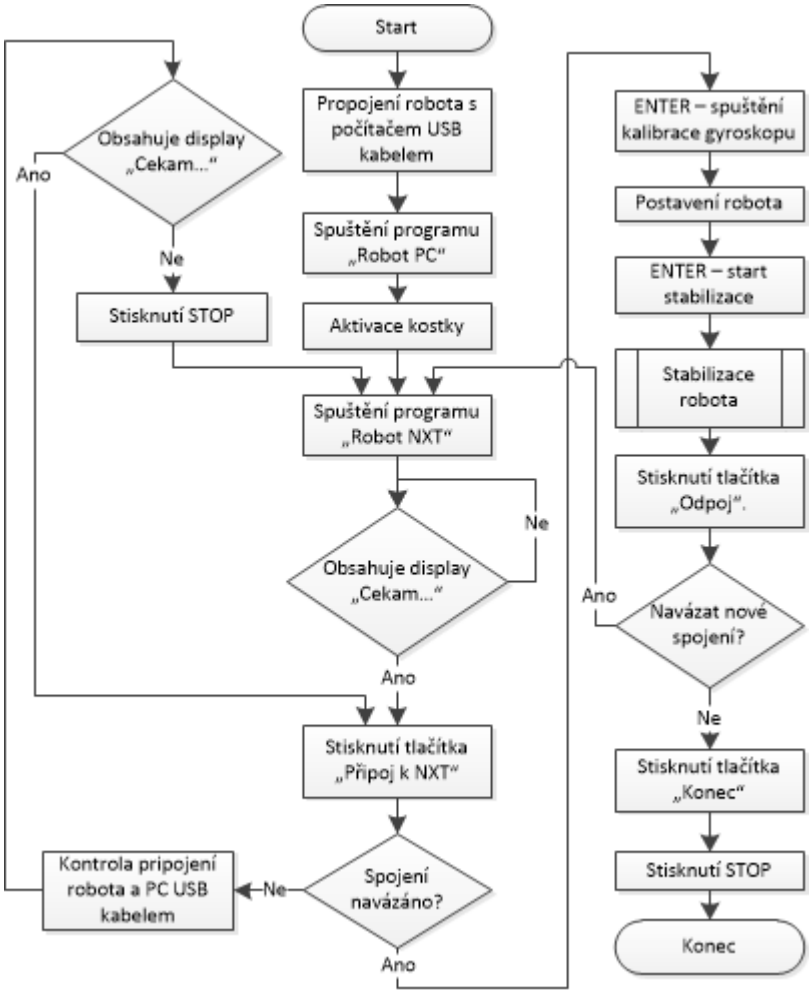
Příloha C – Uživatelská příručka, postup

Programové části:

- „Robot PC“ – aplikace umístěná na počítači.
- „Robot NXT“ – aplikace umístěná na NXT kostce robota.

Operace:

- ENTER – stisknutí oranžového tlačítka na NXT kostce robota.
- STOP – stisknutí tmavě šedého tlačítka na NXT kostce robota.
- Stiskneme tlačítko – rozumí se tlačítko v „Robot PC“.



Obr. B.2 – Postup při manipulaci s robotem