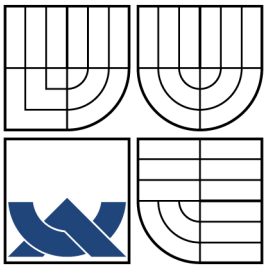
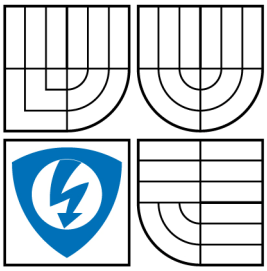


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ŘÍZENÍ 6-TI OSÉHO ROBOTA V RTOS

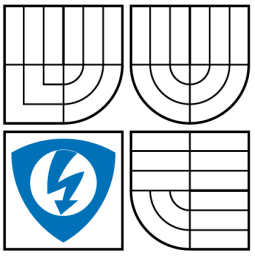
6 AXIS ROBOT CONTROL IN RTOS

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. MIROSLAV VÁCLAVEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. Pavel Kučera, Ph.D.
SUPERVISOR

BRNO, 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Miroslav Václavek	***ID:***	83125
Ročník:	2	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Řízení 6-ti osého robota v RTOS

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s operačním systémem na bázi WIN32 API - RTX. Seznamte se s 6-ti osým laboratorním robotem a prostudujte problematiku řízení servomechanismů. Seznamte se s problematikou řízení systémů s více stupni volnosti. Navrhněte a realizujte řídicí systém laboratorního robota pod operačním systémem reálného času RTX. Navrhněte a realizujte vhodné rozhraní mezi tímto řídicím systémem a nadřazeným systémem.

DOPORUČENÁ LITERATURA:

1. Real-Time extension for Windows - RTX, www.intervalzero.com/rtx.htm, SW product page.
2. Karger Adolf, Kargerová Marie: Základy robotiky a prostorové kinematiky, ČVUT Praha, ISBN 80-01-02183-1.
3. Novák Petr: Mobilní roboty - pohony, senzory, řízení, BEN - technická literatura, 80-7300-141-1.
4. Šolc František, Žalud Luděk: Robotika, VUT Brno 2006.

Termín zadání:	9.2.2009	***Termín odevzdání:***	25.5.2009

Vedoucí práce: Ing. Pavel Kučera, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Vysoké učení technické v Brně
Fakulta elektrotechniky a komunikačních technologií
Ústav automatizace a měřící techniky

Řízení 6-ti osého robota v RTOS

Diplomová práce

Obor: Kybernetika, automatizace a měření
Student: Bc. Miroslav Václavek
Vedoucí práce: Ing. Pavel Kučera, Ph.D.

Abstrakt :

Tato práce se zabývá analýzou, návrhem software a rozborem 3-D kinematiky fyzikálního modelu robotického manipulátoru ROB 2-6. Pro tento manipulátor vytváří řidící software obsahující uživatelské rozhraní na bázi MFC aplikace. Práce se také zabývá řešením nejrůznějších problémů jako například nelinearita servomotorů nebo omezení dosažitelnosti manipulátoru. Práce je dokumentací samostatného projektu a otvírá nové možnosti pro další vývoj v řízení tohoto manipulátoru.

Klíčová slova:

Robotika, Kinematika robotů, ROB 2-6, Servomotor HS-422, Řízení robota

Brno University of Technology
Faculty of Electrical Engineering and Communication
Department of Control, Measurement and Instrumentation

6 AXIS ROBOT CONTROL IN RTOS

Master's Thesis

Specialization of study: Cybernetics, automation and measurement
Student: Bc. Miroslav Václavek
Supervisor: Ing. Pavel Kučera, Ph.D.

Abstract :

The thesis deals with the analysis, software design and the study of 3D kinematic model of robotic manipulator ROB 2-6. It creates the control software for this manipulator, containing user interface based on MFC applications. Moreover, the thesis is concentrated on the solution of various problems, e.g. non-linearity of servomotor or the limitation of manipulator´s attainability.

Finally, it poses a documentation of the single project and offers new possibilities of further development in the control of this manipulator.

Keywords

Robotics, Kinematic of robot, ROB 2-6, Servomotor HS-422, Robot control

Bibliografická citace této práce

VÁCLAVEK, M. *Řízení 6-ti osého robota v RTOS*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 55 s. Vedoucí diplomové práce Ing. Pavel Kučera, Ph.D.

P r o h l á š e n í

„Prohlašuji, že svou diplomovou práci na téma " *Řízení 6-ti osého robota v RTOS* " jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne : Podpis:

P o d ě k o v á n í

Děkuji Ing. Pavlovi Kučerovi, Ph.D., vedoucímu mé diplomové práce, za cenné rady a informace, které mi poskytl. Dále také děkuji svým rodičům za podporu během studia.

V Brně dne : Podpis:

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1. ÚVOD

Tato diplomová práce je pokračováním semestrálního projektu 1 a 2, které se zabývají kinematikou fyzikálního modelu 6-osého robotického manipulátoru ROB 2-6 a objektovou analýzou softwaru, jehož realizace je součástí této práce. Tyto projekty upravuje a dále na nich staví. Hlavním cílem této práce je vyvinout real-time řídící aplikaci pro výše zmíněný manipulátor. Dalším cílem této práce je software vyšší úrovně, umožňující nepřímé – offline programování (plánování dráhy robotu) pomocí uživatelského rozhraní. Jako řídící systém je zvolen osobní počítač PC. Real-time řízení je zajištěno naprogramováním softwaru, který pracuje jako RTOS (real time operation system) aplikace. Pro ovládání hardwaru je použita PCI karta firmy Adventech PCI-1710. Sekce uživatelského rozhraní pro obsluhu robotu je naprogramována jako MFC aplikace v prostředí Microsoft Visual C++ s využitím objektově orientovaného jazyka C++.

Aby mohl být splněn hlavní cíl, vytvoření řídící aplikace, je nutné nejdříve provést analýzu celého systému, návrh budoucího software a provést rozbor kinematiky předloženého fyzikálního zařízení. Práce není tedy čistě softwarově zaměřena, ale zahrnuje také kompletní matematické řešení kinematiky robotu a řešení problémů spojených s aplikací rovnic na reálný systém, jako je například odstranění nelinearity fyzikálního modelu.

Výstupem této práce je plně funkční real-time řídící aplikace, sloužící pro ovládání servomotorů a aplikace pro editaci a spouštění požadované trajektorie v 3D prostoru a to zadávané v kartézských souřadnicích.

1.1 VIZE PROJEKTU

Vizí tohoto celkového projektu, počínaje semestrálním projektem 1, konče diplomovou prací, je vytvořit software v jazyce C++ pro real-time řízení laboratorního robotu ROB2-6. Výsledkem bude tedy robotické rameno, které bude schopno dostat se svým koncovým bodem (svěrákem) do libovolného bodu v 3D prostoru (v dosažitelném prostoru) a pohybovat se v tomto prostoru po předem definovaných dráhách a za předem stanovených podmínek.

1.2 ZÁKLADNÍ POJMY

Model – je to nástroj, který nám pomáhá vytvořit smysluplnou abstrakci skutečného světa. Jeho výhodou je jednoduchost a zároveň přesnost, se kterou odráží skutečnost [1].

Modelovací jazyk – Jedná se o konvenci toho, jakým způsobem bude model zakreslen na papír [1].

Analýza – je to fáze, v níž je vize převáděna do konkrétní podoby. Cílem je jasně stanovit a zachytit požadavky [1].

Návrh – je to převedení požadavků na model, který lze implementovat v software [1].

Implementace – je fáze psaní programu ve zvoleném programovacím jazyce.

Testování – je proces ladění a hledání chyb.

Manipulátor – je tvořen efektory , receptory a ramenem s několika klouby [2].

Kloub – je osa servo motoru pro translační nebo rotační pohyb [2].

Kloubová souřadnice – je označována symbolem q a vyjadřuje úhel natočení nebo posunutí v daném kloubu [2].

Pracovní prostor – prostor kam robot dosáhne, kde ještě může manipulovat.

2. OBJEKTOVÁ ANALÝZA SYSTÉMU

Tato fáze tvorby projektu bývá velmi krátká, často jde jen o zapsání a podrobnější rozvedení vize do psané formy. Pro analýzu ale nepostačuje pouze znát vizi. Je také nutné pochopit, jak systém funguje, jak bude používán a co musí umět dělat. Výsledkem je pak souhrn požadavků systému [1]. Analýza robotického systému se opírá o tři základní případy použití. Vychází se přitom z domény (oblast, ve které robot pracuje) a z fyzického modelu robotu, viz *Obrázek 1*.

zdroj: www.hobbyrobot.cz

Obrázek 1: 6-ti osý robotický systém.

2.1 PŘÍPADY POUŽITÍ

Případem použití rozumíme formulaci jisté situace, do které se systém dostává, v jaké bude používán. Případy použití také úzce souvisí s návrhem systému. Pomáhají totiž nalézt správné třídy. Správné určení případů použití bývá tím nejdůležitějším krokem celé analýzy. Bude-li osoba nebo cizí systém, pracující s navrhovaným systémem, označen za „herce“, pak lze říci, že případ použití je interakce mezi navrhovaným systémem a hercem. Pro účel analýzy je vnitřek systému nezajímavý.

Případy použití pro robotický systém:

- Robot najede do žádané pozice (bod v operativním prostoru [x , y , z]).
- Robot urazí operátorem předem stanovenou trajektorii (při svém pohybu projde předem definovanou množinou bodů).
- Robot urazí danou trajektorii danou rychlostí, přičemž každý úsek trajektorie může projet jinou rychlostí.

2.2 SCÉNÁŘE PŘÍPADŮ POUŽITÍ

Po vytvoření souboru případů, jimiž popisujeme vztahy mezi prvky v doméně, je třeba jednotlivým případům použití přiřadit podrobné chování – napsat jim „scénář“. *Scénář je popis určité sady okolností, které odlišují různé eventuální prvky daného případu použití.* [1]

A. Obsluha žádá přemístění robotu do dané pozice

- Obsluha požaduje pozici R = [x , y , z], která se nachází v manipulačním prostoru. Robot se do této pozice přemístí po libovolné dráze, za libovolný čas. Nezáleží na trajektorii.
- Obsluha požaduje pozici R = [x , y , z], která se nenachází v manipulačním prostoru. Robot setrvává ve své pozici a obsluha dostává informaci o špatném zadání.

B. Obsluha žádá pohyb robotu po dané trajektorii.

- Obsluha žádá pohyb po trajektorii, která se nachází v manipulačním prostoru. Robot vykoná sekvenci pohybů, přičemž prochází předem definovanými body, které tvoří žádanou trajektorii.
- Obsluha žádá pohyb po trajektorii, která se nachází v manipulačním prostoru pouze z části. Část trajektorie, která se nenachází

v manipulačním prostoru, nesmí být povoleno uložit. Robot provede pohyb po dosažitelné části.

- Obsluha žádá pohyb po trajektorii ležící zcela mimo manipulační prostor. Robot setrvává ve své pozici, obsluha je informována o nedosažitelnosti trajektorie.

C. Obsluha žádá pohyb robotu po dané trajektorii, danou rychlostí.

- Obsluha zadá k jednotlivým bodům trajektorie rychlost, se kterou robot do této pozice dorazí.

3. NÁVRH

Návrh vytváří model systému, který lze pak již snadněji implementovat do programovacího jazyka. Návrh zahrnuje vytvoření tříd a objektů zastupujících reálné prvky domény. Skutečné objekty jsou nahrazovány abstraktními objekty a třídami, které věrně popisují jejich vlastnosti.

Objekty domény:

skutečné fyzikální objekty nacházející se v prostředí, se kterým se pracuje:

- Obsluha
- Robot
- Koncový nástroj robotu (svěrák)
- Servomotor
- Manipulační prostor
- Bod v prostoru

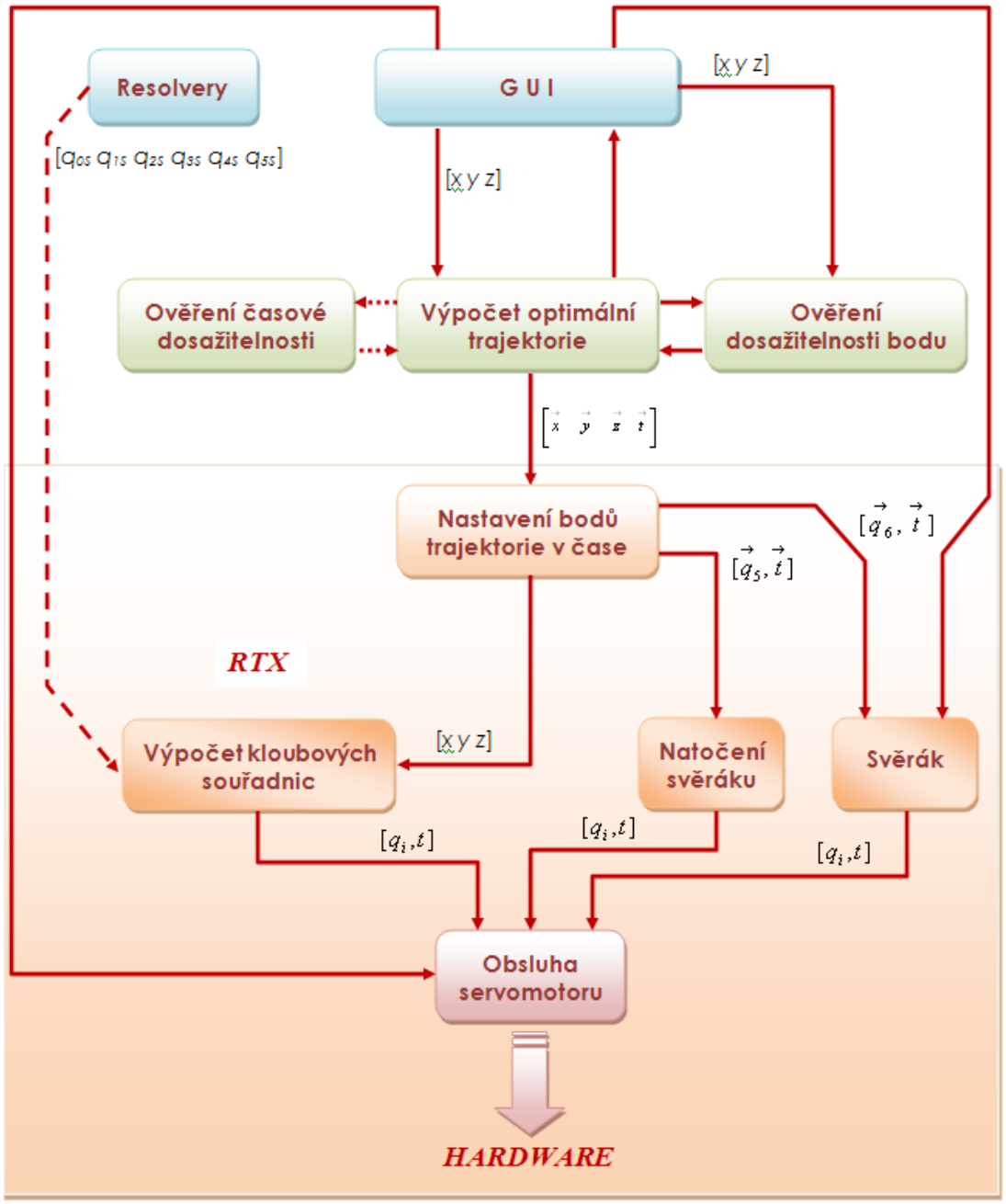
Objekty – třídy systému:

abstraktní objekty – třídy, náhradníci fyzických objektů:

- Servomotor
- Trajektorie v prostoru
- Bod v prostoru
- Manipulační prostor
- GUI - uživatelské rozhraní

Jednotlivou návaznost objektů a tříd na sebe a jejich spolupráci mezi sebou lze vidět z diagramu návrhu software. Ten je uveden na *Obrázku 2*. Nadřazeným objektem celého systému je uživatelské rozhraní GUI. Toto rozhraní spolupracuje s uživatelem, vyhodnocuje jeho požadavky a úkoluje objekty v podřazených

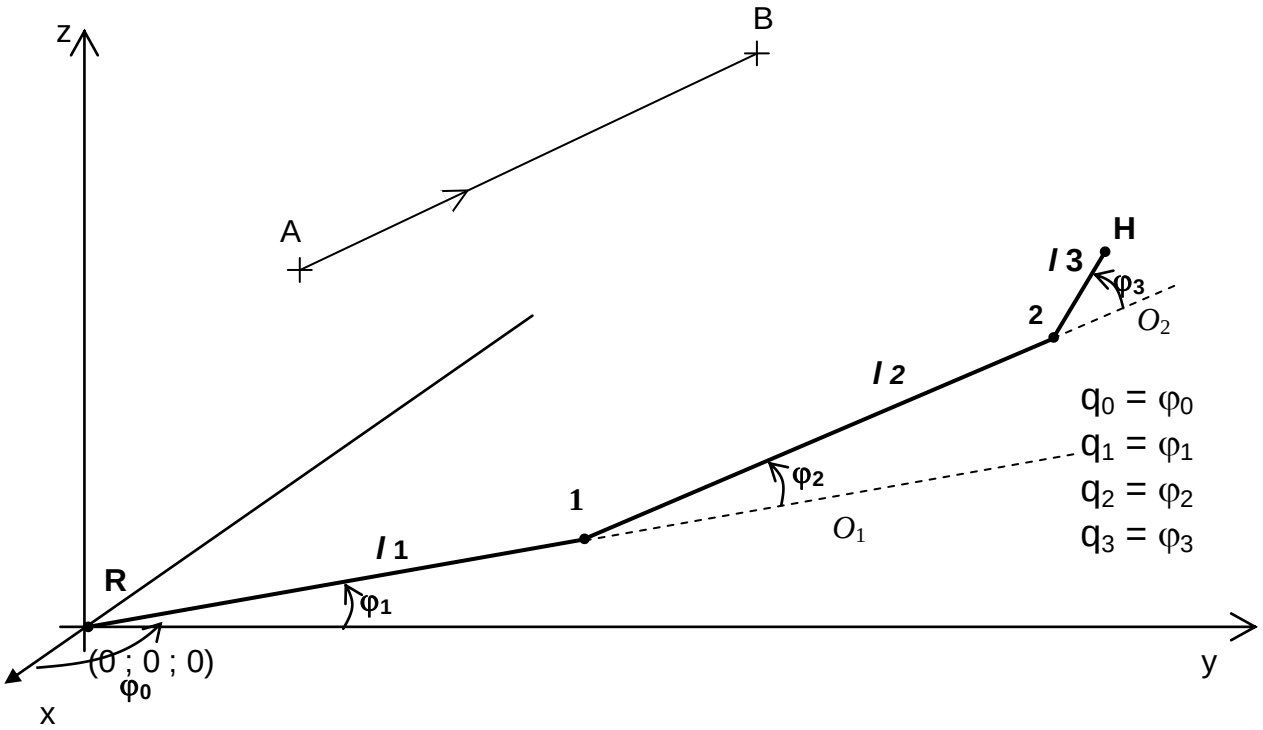
vrstvách. Tento objekt bude seskupovat množství menších objektů, které budou tvořit grafické rozhraní pro zadávání a čtení parametrů, a které budou vytvářet jednoduché výpočetní úkony. Na stejné úrovni se také může nacházet skupina snímačů polohy (resolvery), které zatím nejsou součástí domény (modelu). Případné dodání těchto snímačů již však návrh zohledňuje. Druhou úrovní je úroveň objektů, které mají za úkol výpočetní úkony s body a souřadnicemi. Jedná se o výpočet optimální trajektorie při přesunu do zadaného bodu, o výpočet dosažitelnosti žádaného bodu (zda se bod nachází v prostoru, kde je robot schopen dosáhnout) a o výpočet časové dosažitelnosti, tedy je-li robot schopen do zadaného bodu dojet v požadovaném čase. Třetí úrovní je nastavování bodů trajektorie v čase. Tato úroveň již musí pracovat v reálném čase. Předchozí úroveň vypočítá několik bodů v žádané trajektorii a definuje jakou rychlostí je má robot projet. Stávající úroveň bude v závislosti na čase požadovat vždy v daném okamžiku posunutí robotu do daného bodu a tím zajistí projetí trajektorie v daném čase a tím i danou rychlostí. Další úrovní je výpočet kloubových souřadnic na základě žádaného bodu. Vektor kloubových souřadnic směřuje do páté nejnižší vrstvy, ve které se nachází obsluha servomotoru (vrstva pracující s hardware). Tato poslední vrstva je také například napojena na vrstvu první a to z bezpečnostních důvodů, pro okamžité zastavení všech servomotorů. Vůči semestrálnímu projektu 1 došlo ke změně v třetí a čtvrté vrstvě, kde synchronizaci s časem převzala třetí vrstva a to z důvodů snadnější implementace. Z tohoto důvodu byla posunuta hranice RTX implementace mezi druhou a třetí vrstvu.



Obrázek 2: Diagram návrhu software

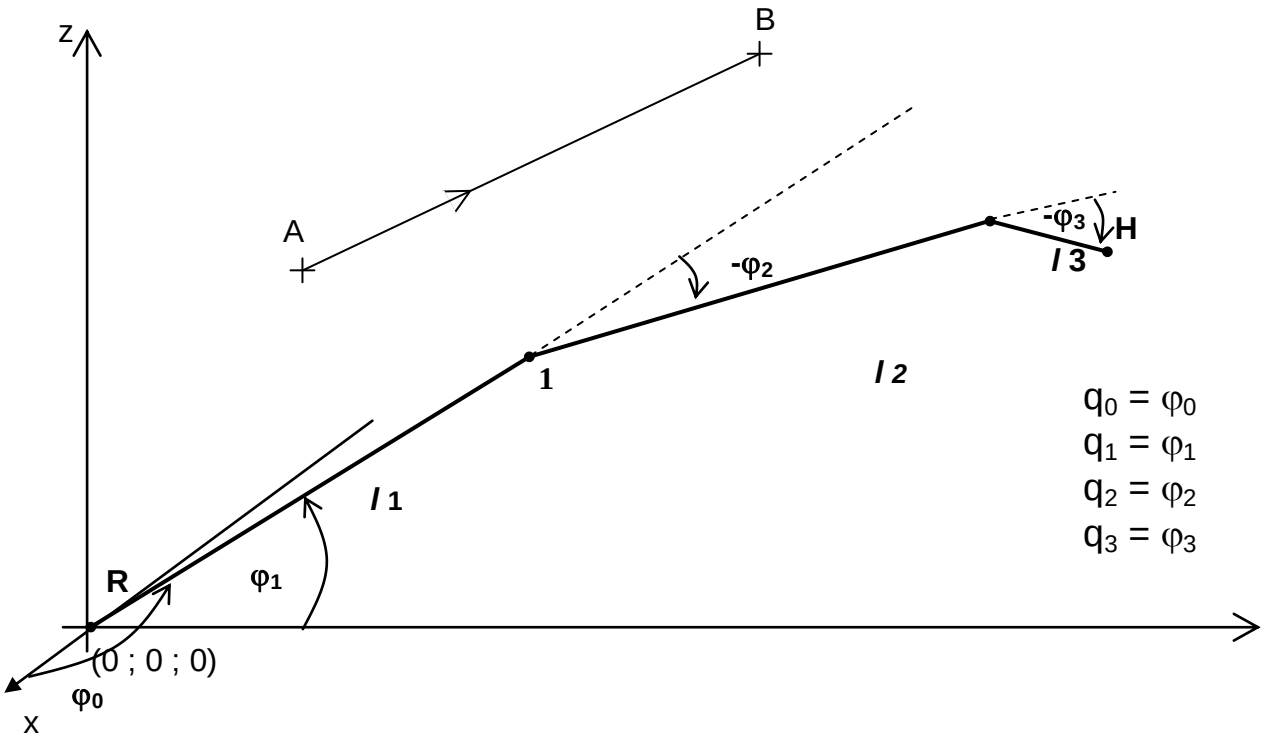
4. KINEMATIKA ROBOTU

Aby bylo možno s robotem dosáhnout určitého bodu v prostoru, je nutné znát jeho kinematiku. Chce-li se tedy koncový bod robotu dostat do nějakého bodu, je zapotřebí znát hodnoty všech kloubových souřadnic. Tento problém řeší nepřímá úloha kinematiky. Bude-li naopak robot znát své kloubové souřadnice a bude chtít vědět v jakém se nachází bodě, pak toto řeší přímá úloha kinematiky. Výsledkem úloh kinematiky jsou rovnice, které na základě známých parametrů vypočítají neznámé žádané parametry. Robot ROB 2-6 se skládá celkem ze tří ramen a otočného koncového efektoru, který má funkci svěráku. Rameno je pak posazeno na otočné lavici, viz. *Obrázek 1*. Z hlediska kinematiky postačí zabývat se pouze čtyřmi klouby. Poslední dva klouby, tedy natočení svěráku (rotace kolem osy O_2) a otevření svěráku, nemají vliv na polohování celého ramene. Vznikne tak tří-ramenný systém se čtyřmi klouby, přičemž třetí rameno tvoří rameno a svěrák dohromady.



Obrázek 3: Kinematický model robotického systému, řešení a)

Obecný model takového systému ukazuje *Obrázek 3* a *Obrázek 4*. Existují právě dvě řešení, jak dosáhnout robotem bodu H pro stejné natočení svěráku vůči koncovému bodu.



Obrázek 4: Kinematický model robotického systému, řešení b)

4.1 PŘÍMÁ ÚLOHA KINEMATIKY

Přímá kinematická úloha se zabývá nalezením rovnic závislostí pozice koncového bodu robotu (nejčastěji nějaké chapadlo) na kloubových souřadnicích *q* robotu (úhly, posunutí v kloubech). Řešení této úlohy je použitím matic homogenní transformace - MHT, které představují funkce kloubových souřadnic. Existují dvě základní matice - pro posun a pro natočení. Je výhodné použít Denavit-Hartenbergovu konvenci, protože převod souřadných systémů s různým posunutím a natočením, se děje násobením matic HT [2]. Obecně tyto matice vypadají následovně .

Translační matice:

$$Trans(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

Rotační matice (rotace kolem osy x):

$$Rot_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.2)$$

Rotační matice (rotace kolem osy y):

$$Rot_y(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.3)$$

Rotační matice (rotace kolem osy z):

$$Rot_z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.4)$$

Kde α je obecně úhel natočení kolem příslušné osy, za který se dosadí vždy příslušný úhel φ. V robotice je však pro kloubovou souřadnici, a tedy i pro úhel natočení φ zaveden symbol q, proto dále bude používáno již jen symbolu q. Pro

zjednodušení výpočtu matice homogenní transformace pro planární robot z *Obrázku 3 a 4* lze robot rozdělit na 4 části. Výsledná matice homogenní transformace se pak bude rovnat:

$$H_{OH} = H_{R0}.H_{01}.H_{12}.H_{23}.H_{34}.H_{45}.H_{5H} \qquad (4.5)$$

Jednotlivé matice pak vypadají následovně:

$$H_{R0} = \begin{bmatrix} \cos(q_0) & -\sin(q_0) & 0 & 0 \\ \sin(q_0) & \cos(q_0) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.6)$$

$$H_{01} = \begin{bmatrix} \cos(-q_1) & 0 & \sin(-q_1) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-q_1) & 0 & \cos(-q_1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.7)$$

$$H_{12} = \begin{bmatrix} 1 & 0 & 0 & l_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.8)$$

$$H_{23} = \begin{bmatrix} \cos(-q_2) & 0 & \sin(-q_2) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-q_2) & 0 & \cos(-q_2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.9)$$

$$H_{34} = \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.10)$$

$$H_{45} = \begin{bmatrix} \cos(-q_3) & 0 & \sin(-q_3) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-q_3) & 0 & \cos(-q_3) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.11)$$

$$H_{5H} = \begin{bmatrix} 1 & 0 & 0 & l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.12)$$

Výsledná matice homogenní transformace je dána vztahem (4.5)

$$H_{0H} = \begin{bmatrix} 0{,}5\cos(q_0 - q_1 - q_2 - q_3) + 0{,}5\cos(q_0 + q_1 + q_2 + q_3) & -\sin(q_0) & 0{,}5\sin(q_0 - q_1 - q_2 - q_3) - 0{,}5\sin(q_0 + q_1 + q_2 + q_3) & x \\ 0{,}5\sin(q_0 - q_1 - q_2 - q_3) + 0{,}5\sin(q_0 + q_1 + q_2 + q_3) & \cos(q_0) & 0{,}5\cos(q_0 + q_1 + q_2 + q_3) - 0{,}5\cos(q_0 - q_1 - q_2 - q_3) & y \\ \sin(q_1 + q_2 + q_3) & 0 & \cos(q_1 + q_2 + q_3) & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Vynásobením výsledné matice sloupcovým vektorem vzniká matice se souřadnicemi koncového bodu v kartézském systému.

$$A_{konc} = H_{0H} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (4.13)$$

kde:

$$\begin{aligned} x \;=\; & 0{,}5l_3[\cos(q_0 - q_1 - q_2 - q_3) + \cos(q_0 + q_1 + q_2 + q_3)] + \\ & + 0{,}5l_2[\cos(q_0 - q_1 - q_2) + \cos(q_0 + q_1 + q_2)] + \\ & + 0{,}5l_1[\cos(q_0 - q_1) + \cos(q_0 + q_1)] \end{aligned} \qquad (4.14)$$

$$\begin{aligned} y \;=\; & 0{,}5l_3[\sin(q_0 - q_1 - q_2 - q_3) + \sin(q_0 + q_1 + q_2 + q_3)] + \\ & + 0{,}5l_2[\sin(q_0 - q_1 - q_2) + \sin(q_0 + q_1 + q_2)] + \\ & + 0{,}5l_1[\sin(q_0 - q_1) + \sin(q_0 + q_1)] \end{aligned} \qquad (4.15)$$

$$z \;=\; l_3 \sin(q_1 + q_2 + q_3) + l_2 \sin(q_1 + q_2) + l_1 \sin(q_1) \qquad (4.16)$$

Ověření vypočítaných rovnic:

Př.: a) budou-li všechny souřadnice q_0 až q_3 nulové pak dle *Obrázku 3* lze odhadnout, že výsledný dosažený bod H by měl, při délce ramen l_1 = 9.5, l_1 = 9.5, l_1 = 11, být H = *[30; 0; 0]*. Dosazením do rovnic 4.14 až 4.16 pak vychází následující výsledek:

$$x = 0{,}5 \cdot 9{,}5[\cos(0) + \cos(0)] + 0{,}5 \cdot 9{,}5[\cos(0) + \cos(0)] + 0{,}5 \cdot 11[\cos(0) + \cos(0)] = 30$$
$$y = 0{,}5 \cdot 9{,}5[\sin(0) + \sin(0)] + 0{,}5 \cdot 9{,}5[\sin(0) + \sin(0)] + 0{,}5 \cdot 11[\sin(0) + \sin(0)] = 0$$
$$z = 11\sin(0) + 9{,}5\sin(0) + 9{,}5\sin(0) = 0$$

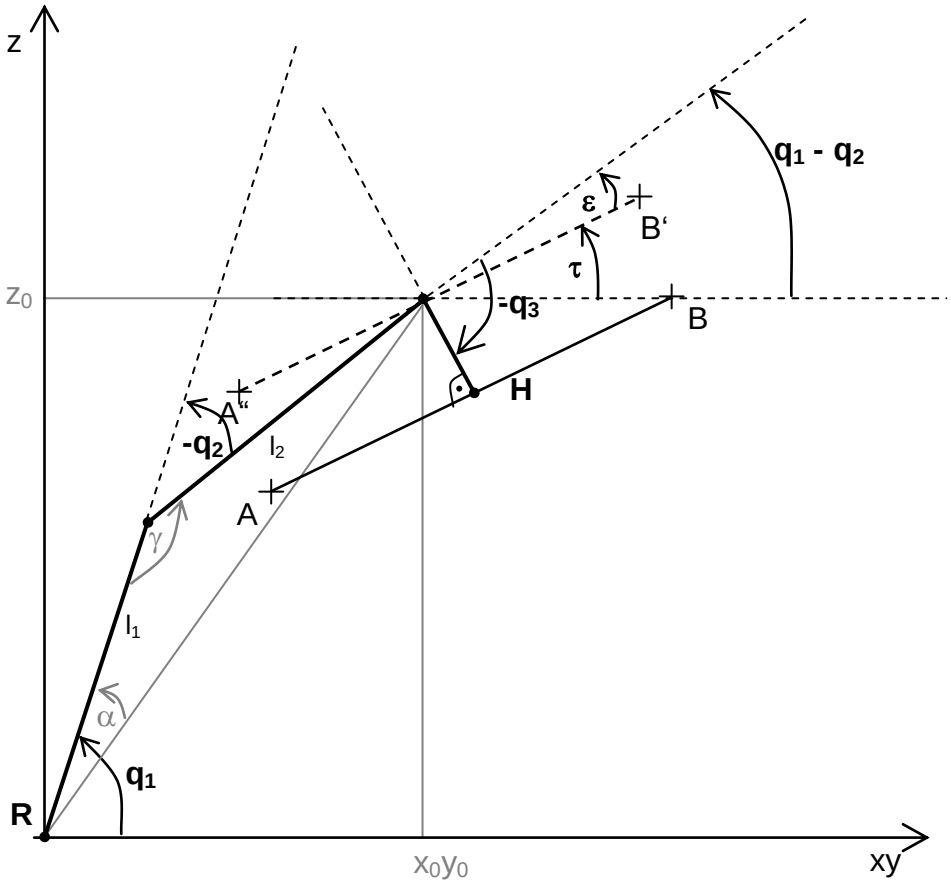
b) budou-li souřadnice q_0 až q_2 natočeny o úhel +90° a souřadnice q_3 bude mít nulové natočení, pak leze opět dle *Obrázku 3* odhadnout pozici, ve kterém se bude nacházet bod H, ta by měla být H = *[0;-20.5; 9.5]*. Dosazením do rovnic 4.14 až 4.16 pak vychází následující výsledek:

$$x = 0{,}5 \cdot 9{,}5[\cos(-90) + \cos(270)] + 0{,}5 \cdot 9{,}5[\cos(-90) + \cos(270)] + 0{,}5 \cdot 11[\cos(0) + \cos(180)] = 0$$
$$y = 0{,}5 \cdot 9{,}5[\sin(-90) + \sin(270)] + 0{,}5 \cdot 9{,}5[\sin(-90) + \sin(270)] + 0{,}5 \cdot 11[\sin(0) + \sin(180)] = -20{,}5$$
$$z = 11\sin(180) + 9{,}5\sin(180) + 9{,}5\sin(90) = 9{,}5$$

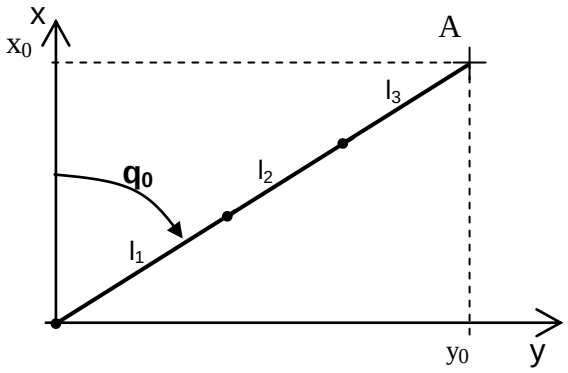
4.2 INVERZNÍ ÚLOHA KINEMATIKY

Je-li známá poloha koncového bodu robotu, pak lze pomocí inverzní kinematické úlohy určit kloubové souřadnice robotu. Inverzní úloha kinematiky může mít více řešení, může mít také nekonečně mnoho řešení. Je také podstatně složitější na výpočet. Nalezení řešení totiž nebývá vždy triviální. Z tohoto důvodu bylo zavedeno při výpočtu inverzní úlohy jisté zjednodušení. Je-li totiž pracovní doménou systému troj-rozměrný prostor, pak minimální počet kloubů pro dosažení kteréhokoli bodu v tomto prostoru je tři. Z toho vyplývá, že bude stačit, když fyzikální model, kterým se práce zabývá, bude řešen pouze pro první tři kloubové souřadnice. Bude-li totiž žádán například pohyb po přímkové dráze z bodu A do

bodu B, pak bude stačit vytvořit pomyslnou rovnoběžnou dráhu, posunutou o délku posledního ramene. Podmínkou pak bude, že poslední rameno se bude pohybovat kolmo k trajektorii. Názorněji si lze situaci představit podle *Obrázku 5* a *Obrázku 6*.



Obrázek 5: Inverzní úloha – kinematický model v rovině (XY)Z



Obrázek 6: Inverzní úloha – kinematický model v rovině XY

Z *Obrázku 5.* je zřejmé, že inverzní úloha může mít řešení pouze tehdy, bude-li platit

$$l = \sqrt{x_0^{\,2} + y_0^{\,2} + z_0^2} \le l_1 + l_2 \qquad (4.17)$$

Platí-li $\sqrt{x_0^{\,2} + y_0^{\,2} + z_0^2} < l_1 + l_2$, pak existuje dvojí řešení, jedno pro $q_1 < 0$ a druhé pro $q_1 > 0$

Z kosinové věty vychází tato rovnice:

$$\cos(\gamma) = \frac{l_1^{\,2} + l_2^{\,2} - l^2}{2 \cdot l_1 \cdot l_2} = a \qquad (4.18)$$

nalezením inverzní funkce lze získat úhel γ

$$\gamma = \arccos(a) \qquad (4.19)$$

Použitím funkce cosinus vzniká pouze jedno ze dvou možných řešení. Druhé řešení vzniká přepočtením goniometrické funkce cosinus na funkci tangens

$$\begin{aligned}
&\cos(\gamma) = a \\
&\cos^2(\gamma) = a^2 \\
&\frac{1}{\cos^2(\gamma)} = \frac{1}{a^2} \\
&1 + tg^2(\gamma) = \frac{1}{a^2} \\
&tg(\gamma) = \pm\sqrt{\frac{1}{a^2} - 1} \\
&tg(\gamma) = \pm\sqrt{\frac{1 - a^2}{a^2}} \\
&tg(\gamma) = \frac{\pm\sqrt{1 - a^2}}{a}
\end{aligned} \qquad (4.20)$$

nalezením inverzní funkce opět vzniká úhel γ, nyní už však má 2 řešení

$$\gamma = \arctan\left(\frac{\pm\sqrt{1-a^2}}{a}\right) = \arctan\left(\pm\sqrt{1-a^2}, a\right) \qquad (4.21)$$

nyní lze podle *Obrázku 5* vypočítat kloubovou souřadnici q_2

$$q_2 = \pi - \gamma \qquad (4.22)$$

Ze sinové věty vychází tato rovnice:

$$\frac{l_2}{\sin(\alpha)} = \frac{l}{\sin(\gamma)}$$
$$\sin(\alpha) = \frac{l_2}{l}\cdot\sin(\gamma) = b \qquad (4.23)$$

nalezením inverzní lze získat úhel α

$$\alpha = \arcsin(b) \qquad (4.24)$$

Použitím funkce sinus opět vzniká pouze jedno z dvou možných řešení. Obě řešení vznikají přepočtením goniometrické funkce sinus na funkci tangens.

$$
\begin{aligned}
&\sin(\alpha) = b \\
&\sin^2(\alpha) = b^2 \\
&1 - \cos^2(\alpha) = b^2 \\
&\cos^2(\alpha) = 1 - b^2 \\
&\frac{1}{\cos^2(\alpha)} = \frac{1}{1-b^2} \\
&1 + tg^2(\alpha) = \frac{1}{1-b^2} \\
&tg(\alpha) = \pm\sqrt{\frac{1}{1-b^2} - 1} \\
&tg(\alpha) = \pm\sqrt{\frac{b^2}{1-b^2}} \\
&tg(\alpha) = \frac{b}{\pm\sqrt{1-b^2}}
\end{aligned}
\qquad (4.25)
$$

nalezením inverzní funkce opět vzniká úhel α, nyní už však má 2 řešení

$$\alpha = \arctan\left(\frac{b}{\pm\sqrt{1-b^2}}\right) = \arctan\left(b, \pm\sqrt{1-b^2}\right) \qquad (4.26)$$

s použitím souřadnic trajektorie x_0, y_0 a z_0 lze podle *Obrázku 5*. vypočítat kloubovou souřadnici q_1

$$q_1 = \arctan\left(\frac{z_0}{\sqrt{x_0^2 + y_0^2}}\right) - \alpha \qquad (4.27)$$

pro výpočet kloubové souřadnice q_3 je nutné vypočítat úhly τ a ε

$$\tau = \arctan\left(\frac{B_z - A_z}{\sqrt{B_x^2 + B_y^2} - \sqrt{A_x^2 + A_y^2}}\right) \qquad (4.28)$$

Dle *Obrázku 5* lze určit úhel ε jako:

$$\varepsilon = q_1 - q_2 - \tau \qquad (4.29)$$

Na základě znalosti ε pak lze vypočíst kloubovou souřadnici q_3 jako:

$$q_3 = \frac{\pi}{2} + \varepsilon \qquad (4.30)$$

Bude-li však požadavek na pohyb posledního ramene rovnoběžně s podložkou, na které je robot umístěn, pak souřadnice q_3 bude vypočtena následovně:

$$q_3 = q_1 - q_2$$

Pro výpočet kloubu q_0 je nutno vyjít z *Obrázku 6*. Z něj lze poměrně snadno určit:

$$q_0 = \arctan\left(\frac{y_0}{x_0}\right) \qquad (4.31)$$

5. HARDWARE

Robotické rameno (viz *Obrázek 1*) se skládá z kovových dílů tvořící jednotlivá ramena. Tyto díly jsou spolu spojeny spojovacími šrouby a do takto poskládaných dílů jsou vmontovány polohovací modelářské servomotory HS-422. Robotické rameno je zakončeno svěrákem pro uchopování předmětů. Servomotory jsou přímo napojeny na PCI-1710 I/O kartu v počítači. To znamená že celý hardware robotu je tvořen pouze těmito servomotory.

5.1 SERVOMOTOR HS-422

Tyto servomotory (také jen serva) jsou spolehlivým řešením pro ovládání RC modelů. Zde je také jejich hlavní využití. Toto využití však není jediným možným. Jsou totiž také vhodné pro jednodušší robotické experimenty. Servo je opatřeno stejnosměrným motorem, převodovkou, snímačem polohy a řídící elektronikou.

zdroj: www. hobbyrobot.cz

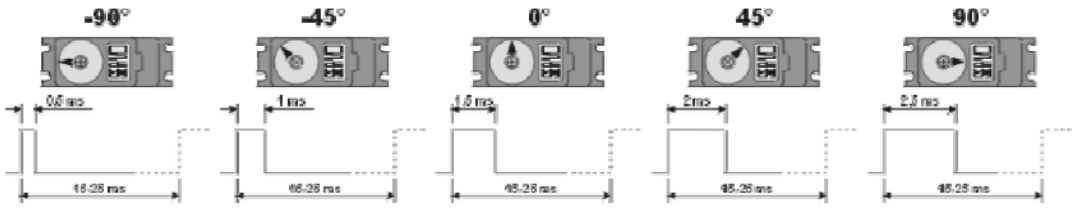
Obrázek 7: Servomotor HS-422

Parametry:

Elektrické parametry:		Mechanické parametry:			Barvy přívodního kabelu:	
Napájecí napětí:	+4,5 až 6 V	Při napájecím napětí:	4,8 V	6 V	černá	0 V
Klidový proud:	< 10 mA	Kroutící moment:	0,33 Nm	0,37 Nm	bílá	řídící impulz
Maximální proud:	cca 800 mA	Čas natočení o 60°:	0,19 s	0,15 s	rudá	+4,5 až 6 V

zdroj: www. hobbyrobot.cz

Požadovaná pozice je do serva přenášena pulzně šířkovou modulací řídicího signálu. Frekvence modulačního signálu je doporučena na frekvenci 50 Hz. Tato hodnota nemusí být přesně dodržena, ale závisí na ní dosažitelný kroutící moment a klidový přídržný moment serva. Poloha natočení hřídele je úměrná šířce řídicího impulsu. Šířka pulzu se pohybuje od 0,5 ms do 2,5ms, přičemž hodnota 1.5 ms je rovna nulovému natočení hřídele. Hřídel se tedy natáčí pro hodnoty -90° až +90°.



Obrázek 8: Polohování serva [4]

Připojení serva je řešeno třemi vodiči. Dva pro napájení a jeden řídící, napojen na I/O PCI kartu. Při volbě vhodného zdroje je nutno zohlednit maximální odběrový proud servomotorů, který činí 800 mA a ten vynásobit počtem servomotorů.

5.2 PCI-1710 KARTA

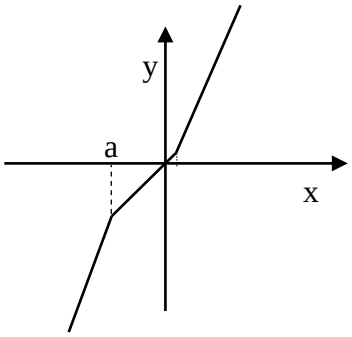
Tato karta mimo jiné disponuje 16 bity digitálních výstupů. Tyto výstupy jsou použity pro napojení servomotorů. Aby mohla být karta používána, je nutno jí adresovat bázovou adresou. Tuto adresu přiděluje zařízením systém, proto je nutné adresu nalézt. O to se stará následující funkce:

```
void  InitPCI_card ( void )
```

Funkce hledá parametry na základě vendor ID a device ID. Tyto hodnoty jsou dány konstantami **PCI1710_VENDOR_ID** = `0x13FE` a **PCI1710_DEVICE_ID** = `0x1710`, které udává výrobce.

6. ODSTRANĚNÍ NELINEARITY SERVA

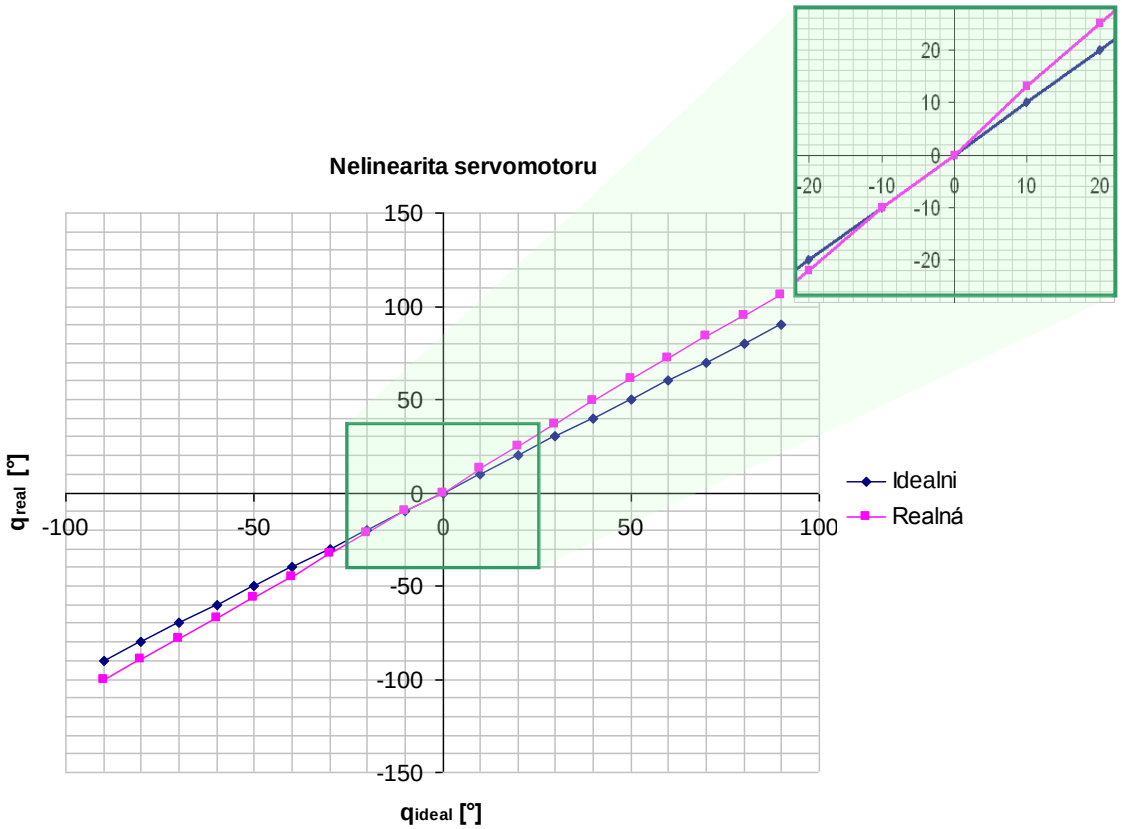
Po aplikaci ideálních výpočtů a následném experimentování bylo zjištěno, že servomotor s kloubovou souřadnicí značenou dle *Obrázku 5*. jako q_3 vykazuje značnou nelinearitu. Ta se projevovala výraznou odchylkou žádaného natočení osy pro určité intervaly požadovaných úhlů. Z tohoto důvodu bylo provedeno měření charakteristiky vyjadřující závislost úhlu skutečného natočení vůči požadovanému natočení. Z důvodu provozních podmínek, které omezovaly měření na použití jednoduchých měřidel (úhloměr), bylo provedeno pouze experimentální jednoduché měření, jehož výsledek lze nalézt v *Tabulce 1*. a následně prezentován grafem na *Obrázku 10*. I když se jednalo pouze o jednoduché měřidlo a ne příliš přesné měření, výsledky byly výborné. Celkový pohyb robotu byl vizuálně daleko přesnější než předtím. Na tento projekt nebyly kladeny větší nároky než je přesnost odhadnutelná pouhým okem (z hlediska praxe lze říci, že se může jednat o použití v aplikaci, kde nebude nutná vysoká přesnost, pokud by to však aplikace vyžadovala, bylo by nutné zpřesnit měření použitím jiných měřidel), proto byl takto dosažený výsledek považován za úspěšný. Z měření vyplývá, že nelinearita je typu obecná nelinearita a její obecný tvar ukazuje *Obrázek 9*.



Obrázek 9: Typ nelinearity pro servo q_3

Aby bylo možno nelinearitu eliminovat, je nutno na intervaly $(-\infty;a)$ a $(b; \infty)$ aplikovat inverzní funkce, které přímky v těchto intervalech upraví tak, aby tvořily

jednu přímku společně se středovou částí. Jak je vidět v detailu na *Obrázku 10*., bod zlomu *a* je roven -10 a bod zlomu *b* je roven 0.



Obrázek 10: Nelinearita na servu q_3 v porovnání s ideálním průběhem

Z naměřených hodnot v *Tabulce 1*. lze stanovit rovnici reálné přímky v intervalu (-∞;a) a k ní funkci inverzní a to stejné lze provést i pro interval (b; ∞).

Interval (-∞;a):

$$y_{real} = 1.114x + b$$
$$y_{inv} = 0{,}897x + b$$

Interval (b; ∞):

$$y_{real} = 1.18x$$
$$y_{inv} = 0{,}85x$$

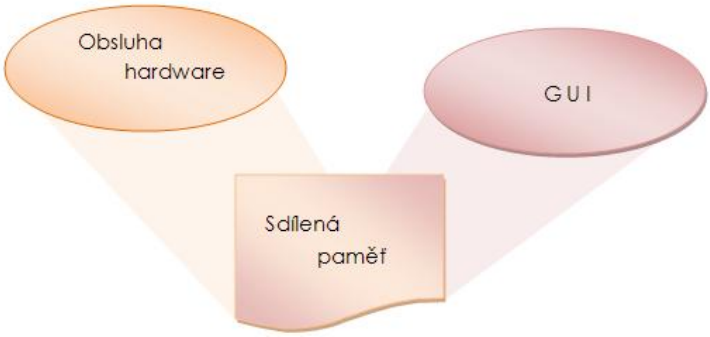
Pronásobí-li se na daném intervalu každá hodnota dle příslušné rovnice inverzní funkce pro daný interval, pak na výstupu bude plynulá přímka bez zlomů a úhel natočení osy serva bude skutečně takový, jaký je požadován.

$\varphi_{\text{žádané}}$ [°]	$\varphi_{\text{reálné}}$ [°]
-90	-100
-80	-89
-70	-78
-60	-67
-50	-56
-40	-45
-30	-33
-20	-22
-10	-10
0	0
10	13
20	25
30	37
40	49
50	61
60	72
70	84
80	95
90	106

Tabulka 1: Naměřené hodnoty úhlu natočení pro servo q_3

7. SOFTWARE

Softwarová koncepce projektu je založena na dvou procesech. Jeden pro bezprostřední řízení hardware, běžící v reálném čase a druhý pro uživatelské rozhraní mezi uživatelem a hardwarem. Komunikace mezi sdílenými procesy je realizována sdílenou pamětí, přes kterou si procesy vyměňují data, viz *Obrázek 11*. Zdrojové kódy obou procesů jsou psány objektově v jazyce C++.

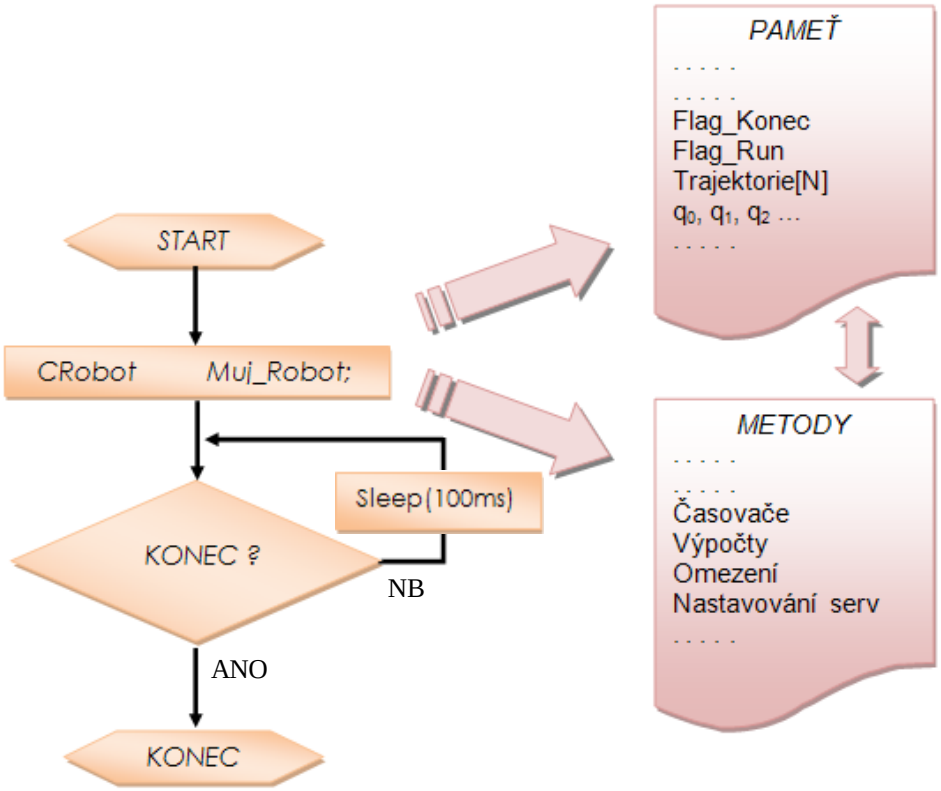


Obrázek 11: Procesy a jejich propojení

7.1 PROCES OBSLUHY HARDWARU

Tento proces je vytvořen jako RTX aplikace. To znamená, že běží v systému v ringu 0, což mu zajišťuje pravidelné přidělování procesorového času, takže je zajištěna real-time odezva. Díky tomu můžeme použít k řízení i takto náročného úkolu (náročná synchronizace jednotlivých serv) běžný stolní počítač. Proces pracuje přímo s vstupně výstupní kartou PCI-1710, na niž ovládá digitální výstupy, na které jsou přímo zapojeny řídící vstupy jednotlivých servomotorů. Činnost procesu je znázorněna v diagramu na *Obrázku 12*. Proces v globálním pohledu provádí velice jednoduchou činnost. Vytvoří instanci třídy *CRobot* a následně přejde do stavu čekání na příznak konce procesu. Kontrola příznaku je prováděna s periodou 100 ms,

přičemž pokud se netestuje, je proces uspán, což je výhodné, protože není zbytečně přidělován procesorový čas. To, že i v době uspání je instance *Můj_robot* aktivní a robot se tudíž může pohybovat, je zajištěno tím, že instance obsahuje sama v sobě své časovače, které se chovají jako samostatný proces, tudíž přístup na hardware je zajištěn pravidelně tak, jak je požadováno. Podrobněji o těchto časovačích níže v kapitole 7.1.1.2.

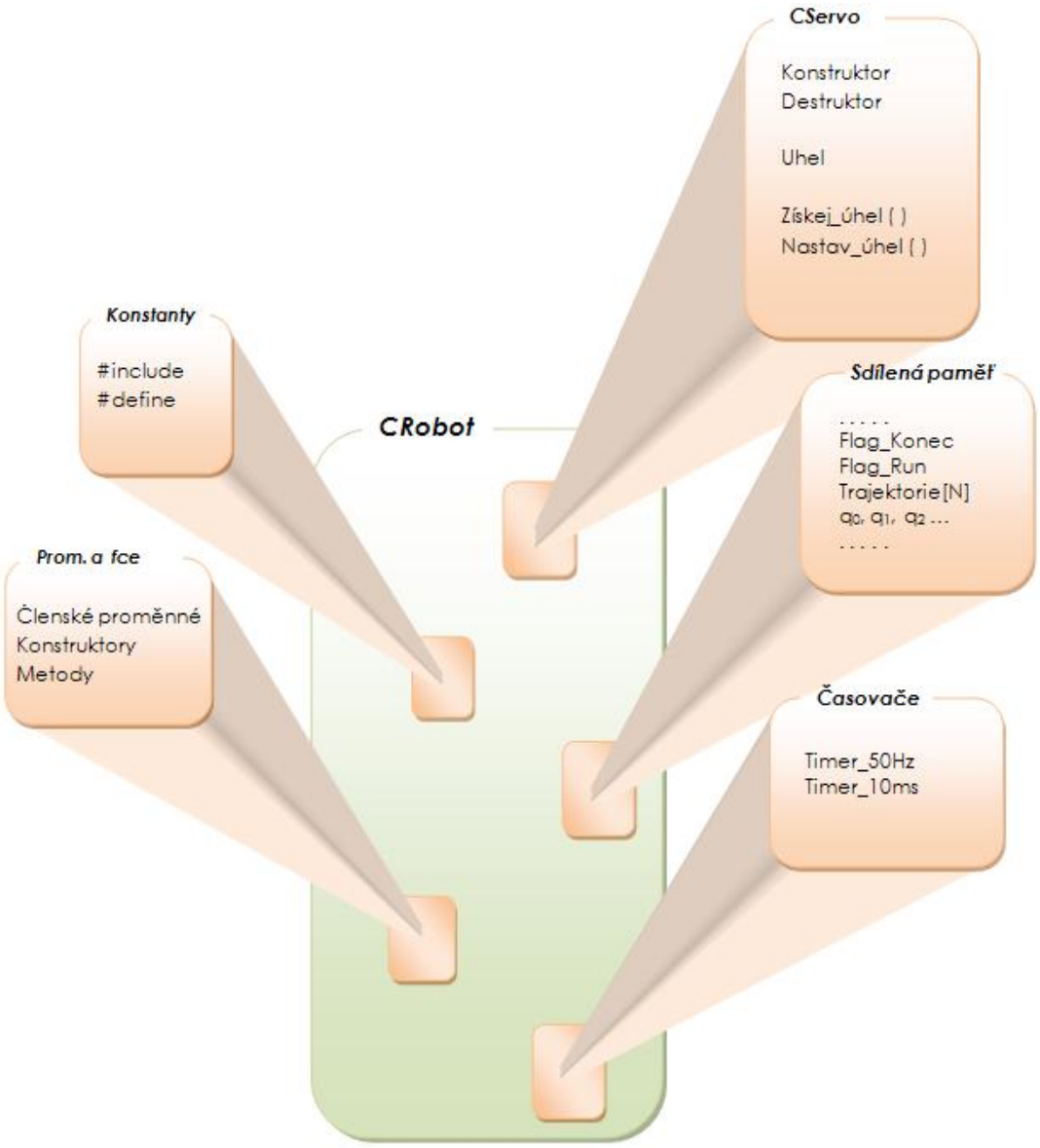


Obrázek 12: Činnost procesu pro obsluhu hardwaru

Hlavní náplň procesu je tedy obsažena v instanci třídy *CRobot*, která již sama pracuje s hardwarem. Podrobněji o třídě *CRobot* v následující kapitole.

7.1.1 Třída *CRobot*

Tato třída vytváří instance (objekty) typu *CRobot*, kterými modeluje chování skutečného robotu. Třída v sobě zapouzdřuje několik členských proměnných a metod, které zpracovávají příchozí data. Zapouzdřuje také dva časovače, sdílenou paměť a mutex pro synchronizaci přístupu do sdílené paměti. V poslední řadě také obsahuje podtřídu *CServo*, která definuje vlastnosti servomotoru. Složení třídy *CRobot* je názorně vidět na *Obrázku 13*. Jednotlivé vyjmenované části budou následně podrobněji rozebrány.



Obrázek 13: Třída *CRobot*

Třída *CRobot* je navržena pro šesti osý robot se třemi rameny a svěrákem, viz. *Obrázek 1*. Počet serv je však možno měnit, stejně jako je možno měnit délku ramen. V tomto případě je však nutné provést změny v konstantách, viz níže. Primárně je třída vázána na kartu PCI-1710 od firmy Adventech. I toto lze však změnit nastavením vhodných konstant.

7.1.1.1 Sdílená paměť

Aby mohl objekt komunikovat s nadřazeným procesem uživatelského rozhraní, je nutné mít v paměti společné úložiště pro oba procesy. Jeden z procesů oblast sdílené paměti vytvoří, druhý pak získá ukazatel. Sdílená paměť se definuje za pomocí funkce:

```
HANDLE RtCreateSharedMemory
(
    DWORD       flProtect,
    DWORD       MaximumSizeHigh,
    DWORD       MaximumSizeLow,
    LPCTSTR     lpName
    VOID        **location
 );
```

zdroj: [3]

Data ve sdílené paměti jsou sdružena ve struktuře *STGSTR*. Struktura obsahuje následující členy:

```
typedef struct
{
      int         Run_traj;
      int         Flag_je_pocatek;
      int         Flag_Kolmo_k_traj;
      int         Konec;
      int         Chyba;
      int         Pause;
      float       q0,q1,q2,q3,q4,q5;

      POZICE      Trajektorie[MAX_POCET_B_TRAJ];
      PPOZICE     pTrajektorie;
      Bod         Aktual_poloha;
} STGSTR, *PSTGSTR;
```

Proměnná ***Run_traj*** slouží pro předání pokynu z uživatelského rozhraní, že trajektorie je zadána a že robot se má pohybovat. Kontrola jejího stavu je prováděna v časovači `Timer_10ms`. Proměnná nabývá hodnot 0 a 1, přičemž 1 znamená „spusť pohyb“. Proměnná je aktivní (tedy 1) po celou dobu pohybu robotu. Po ukončení pohybu je shozena na 0, což signalizuje uživatelskému rozhraní, že pohyb byl vykonán nebo přerušen.

Proměnná ***Flag_je_pocatek*** je příznak indikující, že pohyb bude začínat. Na základě něho pozná proces, že má nastavit ukazatel na první bod trajektorie. Rozdíl oproti proměnné *Run_traj* je v tom, že *Run_traj* je aktivní během celého pohybu a tudíž, bylo-li by přiřazování ukazatele na první prvek trajektorie od *Run_traj* závislé, byla by adresa na první prvek přiřazována neustále a pohyb by neskončil. *Flag_je_pocatek* nabývá hodnot 0 a 1, přičemž 1 signalizuje aktivní stav. Proměnná je nulována po nastavení adresy.

Proměnná ***Flag_Kolmo_k_traj*** je příznak, který indikuje, jak se má poslední rameno robotu pohybovat. Třída *CRobot* umožňuje dva typy pohybu koncového ramene robotu. V případě, že proměnná bude 1, se rameno bude pohybovat kolmo k zadané trajektorii, viz například *Obrázek 5*. Tento pohyb může být například žádoucí při činnostech jako je například svařování nebo řezání. Druhou možností, tedy bude-li proměnná v 0, je pohyb posledního ramene rovnoběžně s podložkou. Tento pohyb je například žádoucí při přenášení předmětů (třeba nádoba s kapalinou). Proměnná může opět tedy nabývat hodnot 0 a 1 a je nastavována uživatelským rozhraním.

Proměnná ***Konec*** předává zprávu od uživatelského rozhraní, že aplikace má být ukončena. Toto je vyhodnocováno v hlavní smyčce procesu, viz *Obrázek 12*. a při hodnotě 1 je zavolán destruktor třídy *CRobot*, který uvolní paměť, proměnné, časovače a mutexy a ukončí se proces. Proměnná nabývá hodnot 0 a 1 a je opět nastavována uživatelským rozhraním

Proměnná ***Chyba*** indikuje chyby, ke kterým došlo v procesu ovládání hardwaru a poskytuje tak informaci uživatelskému rozhraní o chybách. Popis chyb ukazuje *Tabulka 2*. Stávající verze uživatelského programu Robot kontrol 1.0

s chybami sice nepracuje, ale další verze již tohoto prvku v případě potřeby mohou plně využít.

Druh chyby	Kód chyby
Vše je v pořádku	0
Servo 0 je mimo rozsah	1
Servo 1 je mimo rozsah	2
Servo 2 je mimo rozsah	3
Servo 3 je mimo rozsah	4
Servo 4 je mimo rozsah	5
Servo 5 je mimo rozsah	6

Tabulka 2: Číselné kódy chyb v procesu přistupujícímu k hardwaru

Proměnná ***Pause*** indikuje, že proces má přejít do stavu „PAUZA“, tedy pohyb se má zastavit v poloze, ve které se právě nachází, přičemž při opětovné aktivaci proměnné *Run* bude pohyb plynule pokračovat z místa, ve kterém se zastavil. Při přechodu do režimu „PAUZA“ se automaticky nuluje proměnná *Run*. Proměnná je nastavována uživatelským rozhraním a může nabývat hodnot 0 a 1.

Proměnné ***q0,q1,q2,q3,q4,q5*** jsou zpětnou vazbou pro uživatelské rozhraní o tom, jaké úhly byly nastaveny pro jednotlivé osy servomotorů. Ve stávajícím stádiu projektu neexistuje zpětná vazba o úhlu natočení přímo ze servomotorů, tudíž do proměnných jsou ukládány vypočtené a tedy žádané hodnoty úhlů natočení. V případě přidání snímačů natočení tedy revolverů, bude do těchto proměnných ukládána skutečná hodnota přímo ze servomotorů. Proměnné mohou nabývat reálných hodnot určujících úhel natočení ve stupních.

Proměnná ***Trajektorie[MAX_POCET_B_TRAJ]*** obsahuje jednotlivé body trajektorie celého pohybu. Každý bod určuje nastavení servomotorů, natočení svěráku a sevření svěráku v daném bodě, pro daný deseti milisekundový interval. Proměnná je typu *POZICE*, což je struktura následujícího charakteru:

```
typedef struct
{
      CBod        Pozice;
      float       Natoc_Sveraku;
      float       Sverak;
}POZICE, *PPOZICE;
```

kde *Pozice* je objekt třídy *CBod* a určuje žádanou pozici v kartézských souřadnicích, tedy [x,y,z]. Třída *CBod* je popsána níže. *Natoc_Sveraku* udává úhel natočení svěráku a *Sverak* určuje rozevření svěráku v milimetrech.

Proměnná ***pTrajektorie*** je ukazatel na danou pozici, se kterou se bude právě pracovat. A proměnná ***Aktual_poloha*** slouží jako zpětná vazba pro uživatelské rozhraní o aktuální poloze, ve které se robot nachází. Opět je dána pouze vypočtenou hodnotou, jelikož nejsou dostupné žádné snímače polohy koncového bodu robotu. Proměnná je typu *CBod* a udává polohu v kartézských souřadnicích, tedy [x,y,z].

7.1.1.2 Časovače

Třída *CRobot* vytváří pro každou novou instanci dva časovače. Jsou jimi *Timer_50Hz* a *Timer_10ms*. Časovače běží jako samostatné procesy s pravidelným přidělováním procesorového času v době exspirace, dle nastavené priority. Časovače jsou tvořeny v konstruktoru třídy a je k tomu využita následující funkce:

```
HANDLE RtCreateTimer
(
   PSECURITY_ATTRIBUTES   pThreadAttributeslProtect,
   ULONG                  MStackSize,
   VOID          (RTFCNDCL *Routine) (PVOID context,
   PVOID                  Context,
   ULONG                  Priority
   CLOCK                  Clock
);
```

zdroj: [3]

Podrobný význam jednotlivých parametrů je uveden v [3]. Jelikož obslužné rutiny časovačů jsou statickými funkcemi a tudíž jsou pro všechny objekty typu *CRobot*

společné, je nutné předávat obslužným rutinám také ukazatel na objekt, jehož timer právě vyexpiroval. To se děje přes parametr *Context*.

Timer_50Hz:

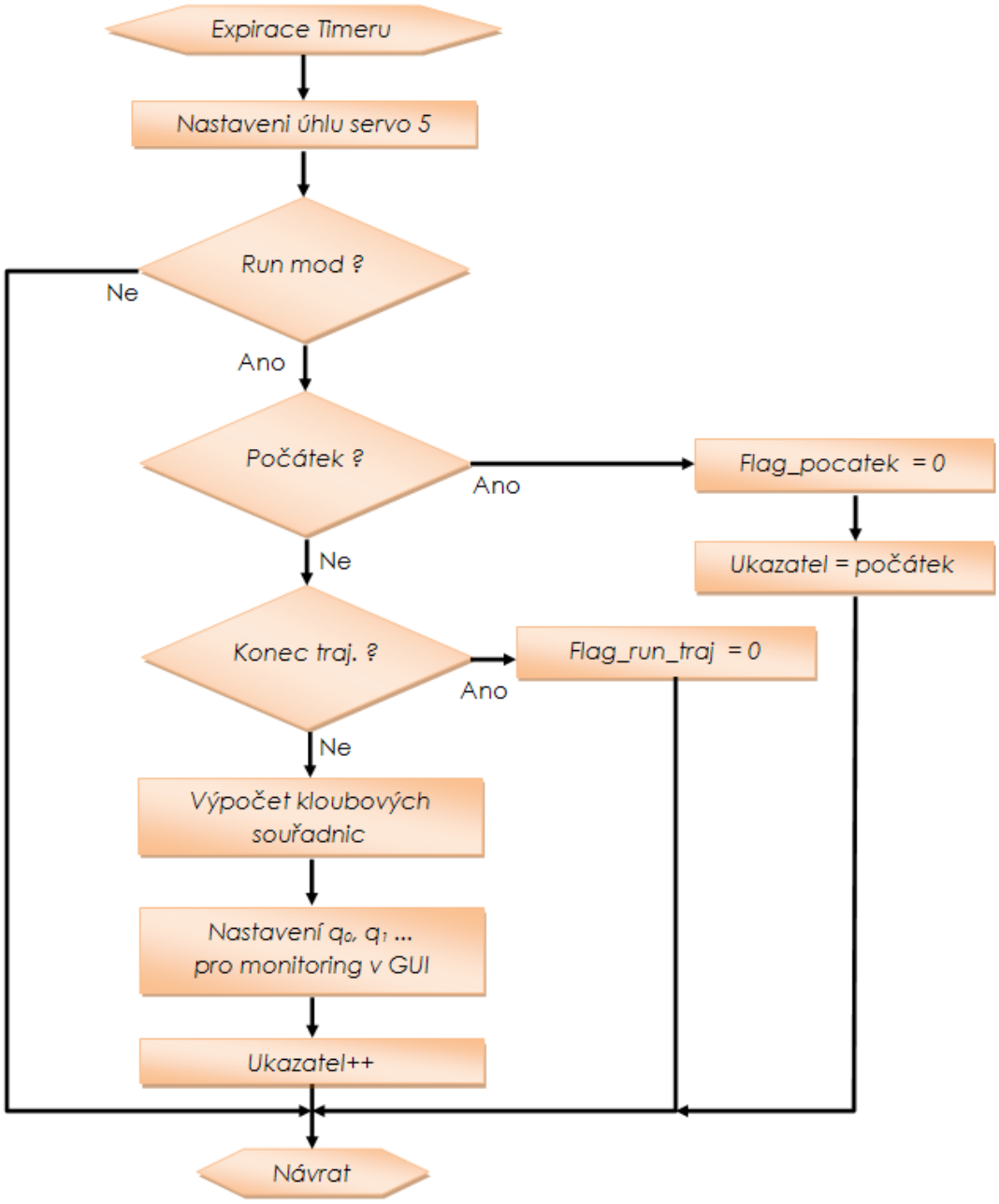
Tento časovač zajišťuje PWM modulaci servomotorů. Expiruje s frekvencí 50Hz a upravuje střídu jednotlivých PWM signálů. Algoritmus časovače pracuje tak, že nejprve nastaví všechny výstupy do hodnoty 1 a následně, dle požadované doby pulzu, která je úměrná úhlu natočení, postupně nuluje jednotlivé signály na 0. Jakmile jsou všechny signály 0, pak je obslužná rutina ukončena a je vrácen procesor dalším procesům.

Timer_10ms:

Tento časovač zajišťuje přesun do dalšího bodu trajektorie. Je-li aktivní režim „Run“, pak každých 10ms je nastavená nová poloha z množiny bodů požadované trajektorie. Pohyb je ukončen, narazí-li se při kontrole na bod typu NULL, tedy bod se souřadnicemi [NULL, NULL, NULL]. Algoritmus obslužné rutiny ukazuje *Obrázek 14*.

7.1.1.3 Konstanty

Konstanty slouží pro nastavení třídy. Jsou zde konstanty udávající minimální a maximální možné natočení jednotlivých servomotorů, konstanty chyb (uvedeno výše) a matematické konstanty pro výpočty. Jednou z důležitých konstant je **POCET_SERV**, která udává, kolik servomotorů obsahuje fyzikální model. Další důležitou konstantou je **ADR_D0_D7**, což je adresa digitálních výstupů na PCI-1710 kartě. Hodnota této konstanty je 16 pro dolních osm bitů. Kdyby však byl robot připojen na datové bity D8 až D15 bylo by nutné nastavit konstantu na 17. Hodnoty určuje dokumentace k PCI kartě.



Obrázek 14: Obslužná rutina časovače *Timer_10ms*

7.1.1.4 Třída CServo

Tato třída vytváří objekty typu *CServo*. Tyto objekty jsou obrazem skutečných servomotorů HS-422. Třída obsahuje privátní proměnné úhel natočení a číslo serva. K těmto proměnným lze přistupovat pouze přes přístupové metody.

Každý objekt vytvořen touto třídou má tedy své číslo a úhel natočení. Přístupové metody jsou:

- Nastav_uhel - tato metoda nastavuje privátní proměnnou úhel
- Získej_uhel - tato metoda získává pro uživatele hodnotu z privátní proměnné úhel

7.1.2 Třída *CBod*

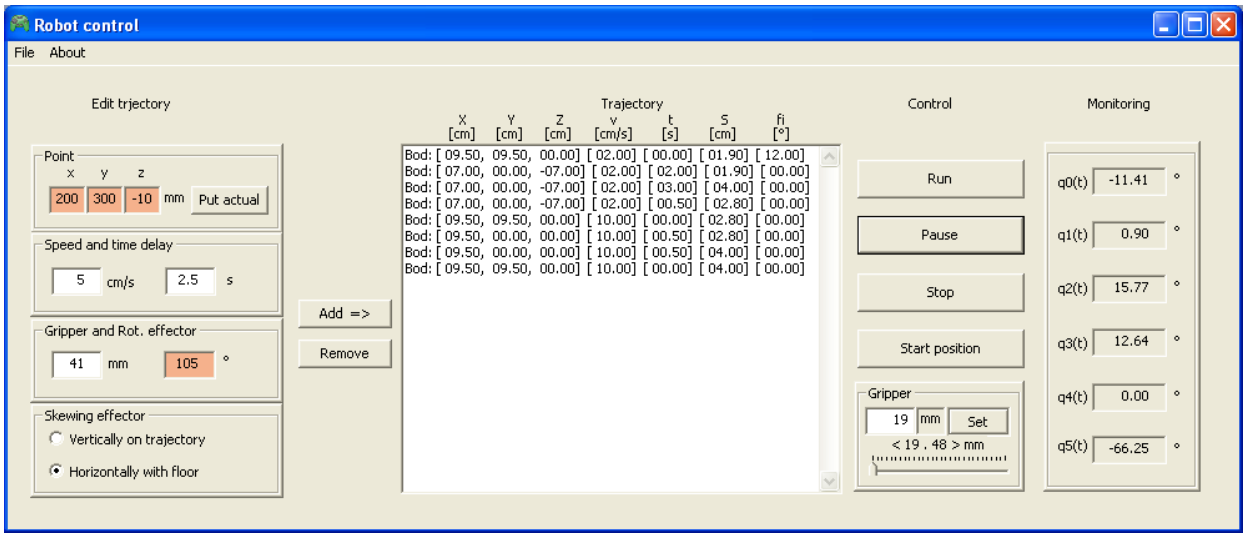
Tato třída vytváří objekty typu *CBod*. Tyto objekty jsou pak obrazem skutečných bodů v prostoru kolem robotu. Třída obsahuje vnitřní proměnné X, Y, Z, které obsahují hodnoty jednotlivých souřadnic bodu. Dále obsahuje konstruktor pro tvorbu objektů a přetížené operátory pro pohodlné operace s objekty typu *CBod* v programu. Přetíženy jsou následující operátory:

- operátor = pro přiřazení hodnot k danému objektu a to na základě rovnosti s jiným objektem, např.: A = B;
- operátor = pro přiřazení hodnot k danému objektu z řetězce typu float, např.: A = b; kde b má definici float b[3];
- operátor+ pro jednoduché sčítání dvou bodů např.: A = B + C; přičítají se k sobě jednotlivé složky obou bodů ($x_B + x_C$;….)
- operátor- pro jednoduché odečítání dvou bodů např.: A = B - C; odečítají se od sebe jednotlivé složky obou bodů ($x_B - x_C$;….)
- operátor* pro jednoduché násobení dvou bodů např.: A = B * C; násobí se skalárně, tedy ($x_B * x_C$;….)
- operátor/ pro jednoduché dělení dvou bodů např.: A = B / C; dělí se skalárně, tedy (x_B / x_C ;….)
- operátor== pro porovnávání dvou bodů, porovnávají se vždy jednotlivé souřadnice, jsou-li si všechny rovny, vrací se true
- operátor!= pro porovnávání dvou bodů, porovnávají se vždy jednotlivé souřadnice, je-li nerovnost alespoň v jedné dvojicí, vrací se false

7.2 APLIKACE UŽIVATELSKÉHO ROZHRANÍ GUI

Aby bylo možno robotické rameno komfortně ovládat a pracovat s ním, je nutné vytvořit grafické uživatelské rozhraní (GUI – graphic user interface)

Uživatelské rozhraní je tvořeno jako MFC aplikace v prostředí Microsoft Visual C++ a umožňuje uživateli komfortně ovládat robotické rameno ROB 2-6 a s ním kompatibilní robotická ramena. Uživatel může pomocí aplikace zadat robotu předem známou trajektorii, zvolit rychlost na jednotlivých úsecích a sledovat stav jednotlivých kloubových souřadnic. Komunikace s procesem ovládajícím hardware je zajištěna přes sdílenou paměť, viz *Obrázek 11*. Aplikace si získává ukazatel na tuto paměť, případně tuto paměť definuje, pokud neexistuje. Využívá k tomu funkci dle předpisu v kapitole 7.1.1.1. Ovládání aplikace je intuitivní a obdobné jako ve standardních aplikacích Windows. Ovládací panel ukazuje *Obrázek 15*.



Obrázek 15: Řídící panel uživatelského rozhraní

Řídící panel je rozdělen do čtyř celků. První částí je editační panel, následuje panel ukazující zvolenou trajektorii, dále pak panel pro řízení a panel monitorovací. Součástí aplikace je také menu. Popis jednotlivých prvků, jejich činnost a práce s nimi je vysvětlena níže.

Editační panel:

Tento panel slouží pro zadávání požadované trajektorie a na řídícím panelu je označen jako *Edit trajectory*. Ovládání tohoto panelu spočívá v nastavení jednotlivých editačních oken a po nastavení všech oken, přidáním bodu do seznamu. Nastavují se následující atributy:

- Pozice v kartézských souřadnicích [x, y, z].
- Rychlost, se kterou robot dorazí do této pozice.
- Doba, po kterou má robot setrvat v dané pozici.
- Rozevření svěráku v dané pozici.
- Natočení hlavy svěráku v dané pozici.

Pro zadávání pozice slouží první tři editační okna, v pod-sekci *Point*, označena písmeny *X, Y, Z*. Zadávaná hodnota je v milimetrech a rozsah hodnot, které lze zadávat určuje dosažitelný prostor, viz kapitola 8. Pro vložení pozice, ve které se právě robot nachází, slouží tlačítko *Put actual*. To načte hodnotu o aktuální pozici ze sdílené paměti. Toto tlačítko je vhodné zejména když je potřeba, aby výchozím bodem trajektorie byla právě momentální pozice. Bude-li zadaná pozice mimo dosažitelný prostor, nebude uživateli umožněno tento bod přidat do seznamu trajektorie a bude o tom informován zčervenáním editačních oken, viz *Obrázek 15*. Po opravě špatně zadaných souřadnic a přidání takto upraveného bodu, editační okna opět zbělají.

Pro zadávání rychlosti slouží editační okno v sekci *Speed*. Zadávaná hodnota vyjadřuje rychlost v centimetrech za sekundu, kterou robot pojede do dalšího bodu. Na servomotorech HS-422 ve skutečnosti rychlost regulovat nelze, servomotory se umí otáčet jen maximální rychlostí. Proto, aby manipulátor měl tuto důležitou funkci, bylo vytvořeno následující řešení. Je známa dráha mezi bodem zadávaným a předcházejícím, který už je zadán. Známe-li tedy dráhu a rychlost, kterou zadáváme, pak výpočtem získáme čas, který robot bude potřebovat na uražení této dráhy. Aby byl pohyb vykonán požadovanou rychlostí, je nutné mezi tyto dva body vložit

několik dalších bodů a v každém z nich bude robot chvíli čekat. Bude-li se počet bodů zvětšovat, pohyb se bude zdát stále více spojitějším. Experimentováním bylo zjištěno, že bude-li bodů tolik, aby doba čekání v každém bodě byla do $t_{MIN} = 10\ ms$, pak se bude pohyb jevit jako plně spojitý. Proto je nutné, aby vypočtitaný čas na uražení trajektorie mezi dvěma body byl rozdělen na N 10-ti ms intervalů, přičemž N udává počet bodů, které mají být přidány. Pokud tyto body budou rozmístěny pravidelně do přímky mezi původní dva body, bude výsledný pohyb spojitý. Podrobněji situaci popíše následující příklad:

Mějme počáteční bod A[10, 0, 0] cm, cílový bod B[20, 0, 0] cm a požadovanou rychlost v = 10 cm/s. Z toho vyplývá:

$$t = \frac{s}{v} = \frac{\sqrt{(A_X - B_X)^2 + (A_Y - B_Y)^2 + (A_Z - B_Z)^2}}{v} \frac{10cm}{10cm \cdot s^{-1}} = 1s \qquad (7.1)$$

$$N = \frac{t}{t_{MIN}} = \frac{1}{0.01} = 100 \quad [-] \qquad (7.2)$$

Dle výpočtu 7.2 bude nutné vložit 100 bodů rovnoměrně mezi body A a B. Aby rozmístění bylo rovnoměrné je zapotřebí vypočíst přírůstky P pro každou souřadnici:

$$P_X = \frac{|A_X - B_X|}{N} = \frac{|10 - 20|}{100} = 0.1\ cm \qquad (7.3)$$

$$P_Y = \frac{|A_Y - B_Y|}{N} = \frac{|0 - 0|}{100} = 0\ cm \qquad (7.4)$$

$$P_Z = \frac{|A_Z - B_Z|}{N} = \frac{|0 - 0|}{100} = 0\ cm \qquad (7.5)$$

Nyní, bude-li se bod A v každém 10-ti ms intervalu měnit o přírůstek *P*, bude bodu B dosaženo za 1s a tudíž rychlostí 10cm/s.

Pro zadávání doby setrvání v daném bodě slouží editační okno nacházející se v sekci *time delay*. Tento čas je parametr, který dává prostor robotu pro manipulaci v daném bodě, například uchopení nebo natočení předmětu. Čas je zadáván v sekundách.

Dalšími editačními prvky jsou rozevření svěráku a natočení svěráku, které se zadávají do editačních oken umístěných v sekci *Gripper and Rot. effector*. Rozevření se udává v milimetrech a natočení ve stupních. Opět zde platí stejný princip jako u zadávání pozice. Bude-li se uživatel snažit přidat bod s nastavením rozevření svěráku nebo natočením větším než je povoleno, aplikace mu to nepovolí a bude o tom informován změnou barvy editačního okna. Rozsah rozevření je <19; 48> mm a natočení je povoleno v mezích <-90; 90> °. Jsou-li všechna editační okna nastavena, pak stačí kliknout na tlačítko *Add* a je-li vše v pořádku, bod se přidá do seznamu. Pokud všechna okna nejsou vyplněna, pak nevyplněné okno je bráno jako nula.

Panel Trajektorie:

Tento panel slouží pro přehled o tom, jaké body jsou uloženy v trajektorii. Umožňuje pomocí tlačítka *Add* bod přidat a pomocí tlačítka *Remove* bod ubrat. Ubírá se vždy bod, který je myší vybrán. Odstraněný bod se přesune do editačních oken v editačním panelu. Význam jednotlivých prvků záhlaví je následující:

- *X, Y, Z* - jsou kartézské souřadnice pozice, uváděno v centimetrech
- *v* - je rychlost se kterou robot do bodu dorazí, uváděno v centimetrech za sekundu
- *t* - je čas, po který robot v dané pozici setrvá, uváděno v sekundách
- *S* - je rozevření svěráku, uváděno v milimetrech
- *fi* - je úhel natočení svěráku, uváděno ve stupních

Panel řízení – Control:

Panel *Control* slouží ke spouštění, zastavování a upravování pohybu. Nachází se zde čtyři ovládací tlačítka a panel ovládání svěráku. Prvním tlačítkem je tlačítko *Run*. Toto tlačítko spouští jednak nově započatý pohyb a pohyb přerušený tlačítkem *Pause*. Je-li robot v pohybu tlačítko není aktivní.

Tlačítko *Pause* slouží k přerušení pohybu robot. Potřebuje-li obsluha z nějakých důvodů, ať už manipulačních nebo bezpečnostních robot zastavit, pak použije právě toto tlačítko. Po stisknutí tlačítka se robot zastaví a zůstane v aktuální poloze. Z tohoto stavu jej lze dostat dvěma způsoby. Prvním je pokračování v pohybu a tedy stiskem tlačítka *Run*, jehož stisk se uživateli nabídne tím, že tlačítko se stane aktivním nebo druhým způsobem a to stiskem tlačítka *Stop*, kdy robot sice zůstává ve stávající pozici, ale již se nachází v módu stop, tedy pohyb se považuje za ukončený a v případě opětovného stisku tlačítka *Run* se pohyb začne vykonávat od počáteční pozice. Tlačítko *Stop* může uvést robot do režimu stop i během pohybu. Nutno však podotknout, že toto tlačítko neslouží jako nouzové tlačítko stop, které by mělo vypnout všechny servomotory. Toto je řešeno při zavření aplikace, kdy po zmáčknutí zavíracího křížku je přerušen PWM řídící signál do všech servomotorů. To sice nezajišťuje bezpečné ukončení činnosti robotu, jelikož se může nacházet v takové situaci, ve které odpojení servomotorů může vést například k destrukci předmětu, se kterým se manipuluje, ale zajišťuje to bezpečné odpojení v případě, že se některý ze servomotorů začne chovat nestandardním způsobem a ohrožuje bezpečnost.

Tlačítko *Start position* umožňuje uvést robot do startovní pozice požadované trajektorie. Tuto činnost sice vykonává i tlačítko *Run*, nenachází-li se robot ve startovní pozici. Rozdíl je však v tom, že po stisku tlačítka *Run* se do výchozí pozice najíždí rychlostí definovanou ve startovní pozici, zatímco po stisku tlačítka *Start positron* se do startovní pozice najíždí maximální rychlostí. Tlačítko je aktivní pouze není-li robot v pohybu. Pokud je tlačítko *Start position* stisknuto v režimu pause, pak dojde k ukončení pohybu a robot se přesune do počáteční pozice.

Panel *Gripper* slouží uživateli k ovládání svěráku, není-li robot v pohybu, tedy je-li v režimu stop nebo pause. Smyslem tohoto ovládacího prvku je možnost operátora dolaďovat sevření nebo rozevření svěráku podle aktuálních potřeb. Například nebude-li zadání rozměrů uchopovaného předmětu zcela přesné, pak již za běhu aplikace může operátor zastavit vykonávaný pohyb a špatně uchopený předmět (např. málo sevřený) uchytit lépe pomocí tohoto panelu. Pro komfortní práci slouží dva editační prvky. Jedním je standardní editační okno, jehož nastavenou hodnotu je nutno potvrdit tlačítkem *Set* a druhým je editační táhlo, které dovoluje nastavovat rozevření plynulým tahem, přičemž nastavená hodnota se zobrazuje v editačním okně.

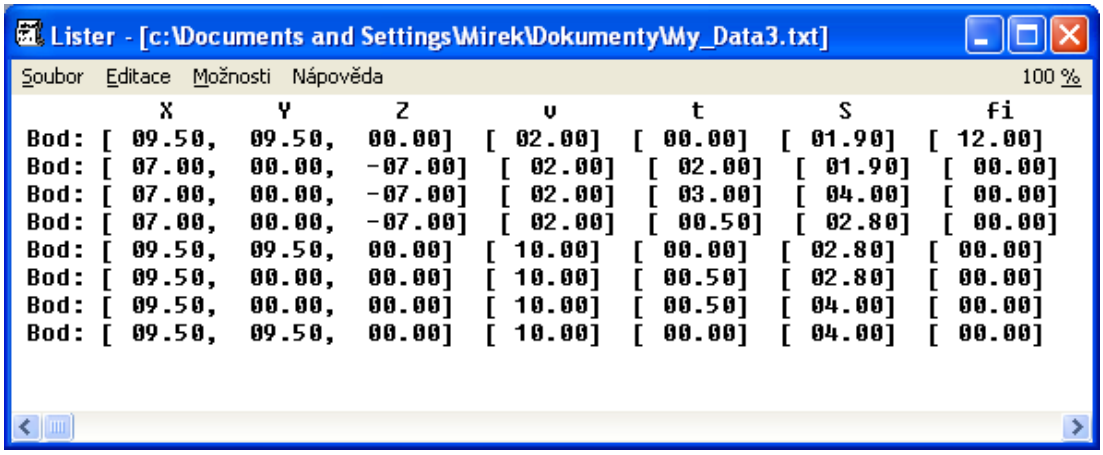
Panel Monitoring:

Tento panel slouží ke sledování aktuálních hodnot kloubových souřadnic. Jak popisuje kapitola 7.1.1.1 tyto hodnoty nejsou skutečnou zpětnou vazbou o úhlu natočení ze servomotorů, jsou to jen hodnoty vypočtené ze žádané hodnoty, které počítá výpočetní metoda třídy *CRobot*. I tak je ale výhodné tyto hodnoty sledovat, protože uživatel není schopen v reálném čase, v případě potřeby, si tyto hodnoty vypočíst a funkce monitoringu je tak připravena pro pozdější přidání zpětné vazby úhlů natočení od jednotlivých servomotorů. Hodnota je obnovována každých 100 ms.

Panel Menu:

Položka menu, nacházející se v záhlaví aplikace, vytvořena na bázi standardních aplikací Windows, umožňuje následující čtyři činnosti. První dvě z nich jsou uložení a načtení souboru dat. Tato funkce umožňuje vytvořit si v aplikaci požadovanou trajektorii dle výše uvedených postupů a tu následně uložit pro další použití, což výrazně zvýší komfort aplikace. Dalším využitím je situace, kdy bude například existovat software vyšší úrovně, který bude sloužit právě pro generování trajektorie při daných činnostech. Zde pak soubor může sloužit jako prostředek pro předávání dat mezi aplikacemi. Aplikace soubor ukládá v předem přesně definovaném tvaru a tento tvar je požadován i pro načtení dat. V případě nedodržení stanoveného formátu

dat, nebudou data načtena. Ukázka formátu ukládání souboru dat je znázorněna na *Obrázku 16*. Je požadován soubor s příponou txt a každý řádek musí být ukončen oddělovačem nového řádku, tedy znakem '\n'.



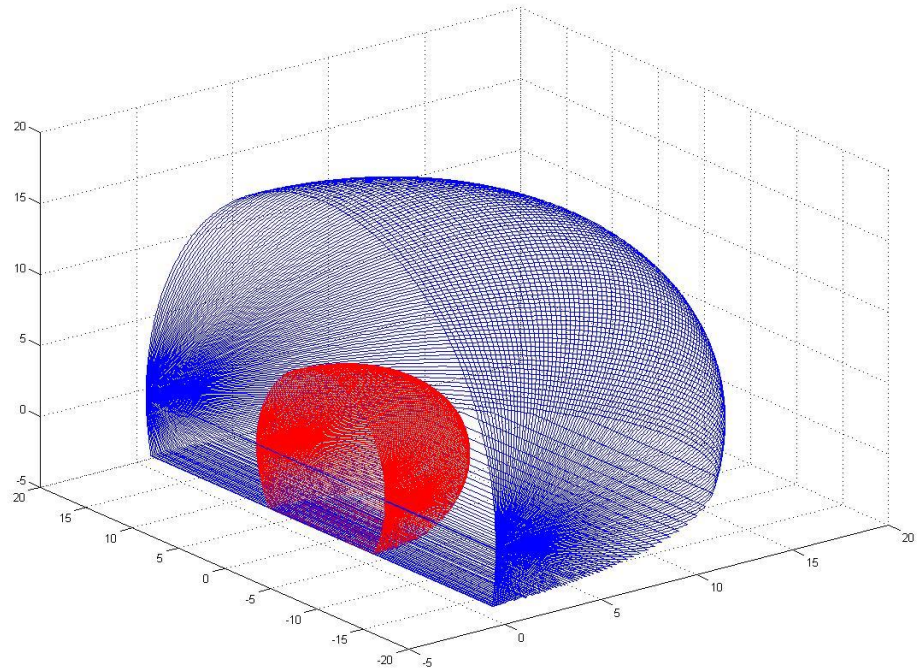
Obrázek 16: Formát ukládaných dat do souboru

Dalším prvkem menu je volba *Exit*, tedy možnost ukončit aplikaci. Tato volba má identické chování se zavíráním aplikace pomocí křížku. Posledním prvkem menu je volba *About*. Zde je velmi stručně popsána aplikace. Okno ukazující tyto údaje je vyobrazeno na *Obrázku 17*.

Obrázek 17: Okno informačního panelu *About*

8. MANIPULAČNÍ PROSTOR ROBOTU

Manipulační prostor robotu udává, jaké souřadnice, do kterých se má robot dostat, může uživatel požadovat. Tento prostor je dán konstrukčním provedením robotu. Aby prostor mohl být určen, je nejprve nutné stanovit nulovou souřadnici, viz *Obrázek 4* bod *R*. Tímto bodem je průsečík osy kloubové souřadnice q_0 a q_1, tedy servomotoru 1. Z *Obrázku 1*. je vidět, že tento bod [0, 0, 0] se nenachází přímo na podložce, na které je robot umístěn, ale je vyvýšen. Toto zvýšení činí 4.1 cm. Z konstrukčních vlastností a experimentů vyplývá, že existuje prostor kolem nulového bodu (tedy paty) robotu, kde se již robot dostat nemůže. Tento prostor lze popsat koulí o poloměru $r = 7\ cm$. Prostor, ve kterém lze manipulovat s robotem, je tedy zevnitř touto koulí ohraničen. Zvenku je ohraničen opět koulí.



Obrázek 18: Manipulační prostor robotu

Tentokráte se jedná o kouli s poloměr rovným součtu délek prvních dvou ramen, což je logické, protože maximální dosah má robot právě při natažení těchto dvou ramen (je však nutno se vymezovat pouze na body dle *Obrázku 5*. tedy body A', B' atd..

což jsou souřadnice osy q_4). Dosažitelný prostor velmi názorně ukazuje *Obrázek 18*. Obě koule mají střed v průsečíku osy q_0 a q_1. Je-li tedy žádaný bod dosažitelný zjišťuje následující funkce uživatelského rozhraní:

```
int Overeni_dosazitelnosti(CBod A)
```

Vstupem do této funkce je objekt třídy *CBod*, tedy bod o souřadnicích *X, Y, Z* a na základě následujících nerovnic je určeno, zda bod patří dostupnému prostoru, či nikoli. Bod musí vyhovovat oběma nerovnicím:

$$x^2 + y^2 + z^2 \leq (l_1 + l_2)^2 \tag{8.1}$$

$$x^2 + y^2 + z^2 > 7^2 \tag{8.2}$$

9. ZÁVĚR

Výstupem tohoto projektu je software pro řízení fyzikálního modelu robotického manipulátoru ROB 2-6 a s ním kompatibilní. Ovládání a možnosti tohoto software jsou popsány v kapitole 7. Systém je určen pro manipulaci s předměty. Analýza systému jako celku je rozebrána v kapitole 2. Zde je rozebráno prostředí robotu a jednotlivé třídy a objekty, které toto prostředí vystihují. Na toto plynule navazuje kapitola 3, která popisuje návrh software. Kinematika a matematický rozbor pohybu robotu je rozebrán v kapitole 4. Zde jsou k nalezení všechny hlavní rovnice pro výpočet všech kloubových souřadnic a další rovnice, zajišťující pohyb robotického ramene po žádané trajektorii. Kapitola 5 pak popisuje použitý hardware, způsob jeho ovládání a jeho omezení. Na to navazuje kapitola 6, která řeší nelineární chování servomotoru číslo 4, který vykazoval velkou nelinearitu. Pravděpodobný důvod této nelinearity je chyba ve vnitřním snímači polohy uvnitř servomotoru (pravděpodobně způsobeno opotřebením). Tento problém byl v danou chvíli neopravitelný a tudíž bylo nutné přistoupit k řešení nikoli lineárního, ale již nelineárního systému. Kapitola 7 pak popisuje softwarové řešení a to jak pro bezprostřední řízení hardware, tak pro aplikaci vyšší úrovně. V poslední 8. kapitole je pak popsán manipulační prostor pro robot.

Vývoj na tomto rozsáhlém projektu ještě stále není ukončen, zakončená je pouze určitá část. Je zde několik dalších možností, jak projekt rozšířit. První z nich je kontrola časové dosažitelnosti daného bodu. Tuto úlohu již zohledňuje návrh v kapitole 3. Stávající systém totiž prozatím neumí uživateli říci, zda-li jím požadovaná rychlost do následujícího bodu není příliš velká a tudíž nereálná. Dalším rozšířením může být více variant pohybu koncového ramene. Například pohyb ne pouze kolmo k trajektorii, ale i pod různými volitelnými úhly. Nejvýznamnějším a nejdůležitějším rozšířením je však přidání zpětné vazby. Ta zde výrazně chybí. Monitoring sice v aplikace existuje, jsou to však pouze vypočtené hodnoty, nikoli skutečné. Pro další vývoj je tedy doporučeno přidat minimálně zpětnou vazbu od úhlů natočení jednotlivých kloubů pomocí snímačů úhlu natočení a případně také zpětnou vazbu od výsledné polohy, například pomocí GPS, kamery, nebo jinak.

10. POUŽITÁ LITERATURA

[1] LIBRETY, J. *Naučte se C++ za 21 dní*. Praha: Computer Press 2002 .

[2] ŠOLC, F., ŽALUD, L. *Robotika*. Brno: VUT 2006.

[3] *RTX Support* [online] Dostupné z: ⟨http://www.ardence.com⟩.

[4] *Data sheets HS-422* [online]. Hobbyrobot, 2006. Dostupné z: ⟨http://www.hobbyrobot.cz/PDF/T_T101.pdf⟩.

[5] PROSISE, J. *Programování ve windows pomocí MFC* Praha: Computer Press 2002 .

11. OBSAH PŘILOŽENÉHO CD

V kořenovém adresáři na CD se nacházejí tyto podadresáře:

- Datasheets ... obsahuje používané katalogové listy.
- Dokument ... obsahuje dokumentaci k diplomové práci
- Program ... obsahuje veškerý vytvořený software.
- Video-dokumentace ... obsahuje video prezentující funkčnost.