Průběžná zpráva

Název projektu:

Výzkum, vývoj a validace univerzální technologie pro potřeby moderních ultrazvukových kontrol svarových spojů komplexních potrubních systémů jaderných elektráren

(Číslo projektu: TA01020457)

Dílčí cíle:

SW pro řízení pohybu robota

Název zprávy:

Algoritmy řízení pro vybranou variantu manipulátoru pro NDT - návrh realizace

Zapsáno (místo, datum):

KKY, 11. prosince 2013

Autor:

Martin Švejda

Tento dokument je součástí projektu TAČR Alfa TA02020414 „Nová robotická dálkově ovládaná technologie pro diagnostiku a opravu ponořených zařízení“ realizovaného za finanční spoluúčasti Technologické agentury České republiky.

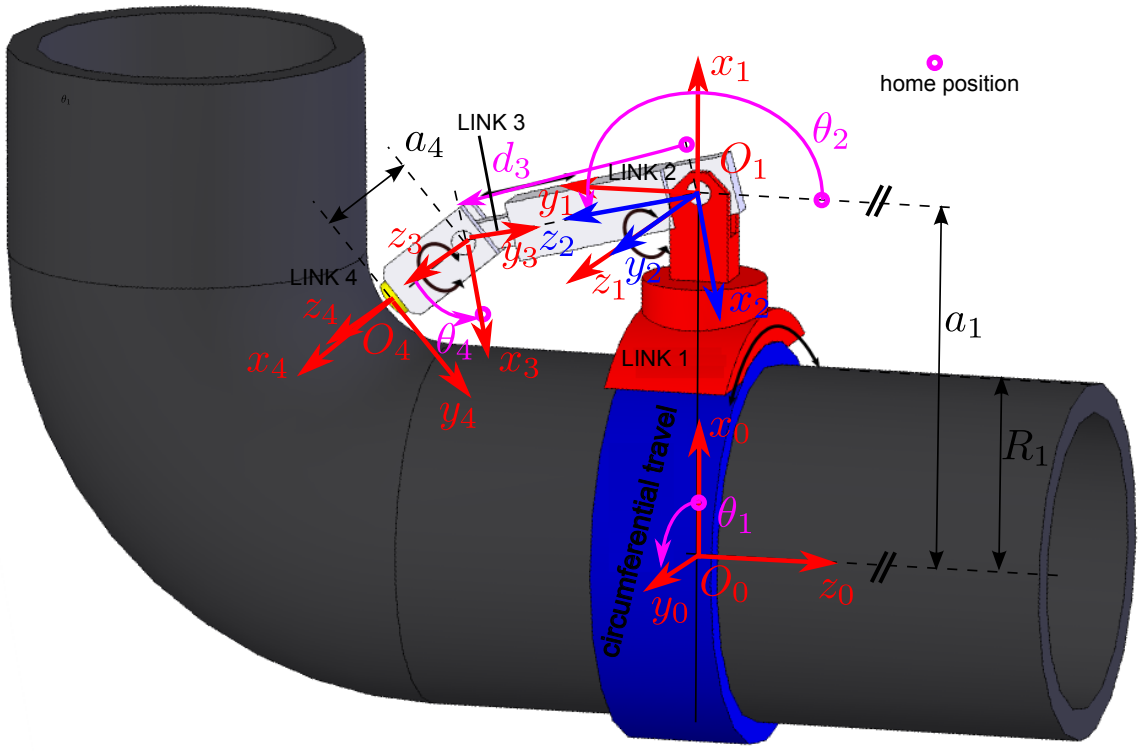
1 Úvod

Předložená zpráva se zabývá návrhem řídícího softwaru (SW) 4DoF manipulátoru určeného pro polohování ultrazvukové sondy v aplikacích nedestruktivního zkoušení (NDT) svarů komplexních potrubních systémů v JE.

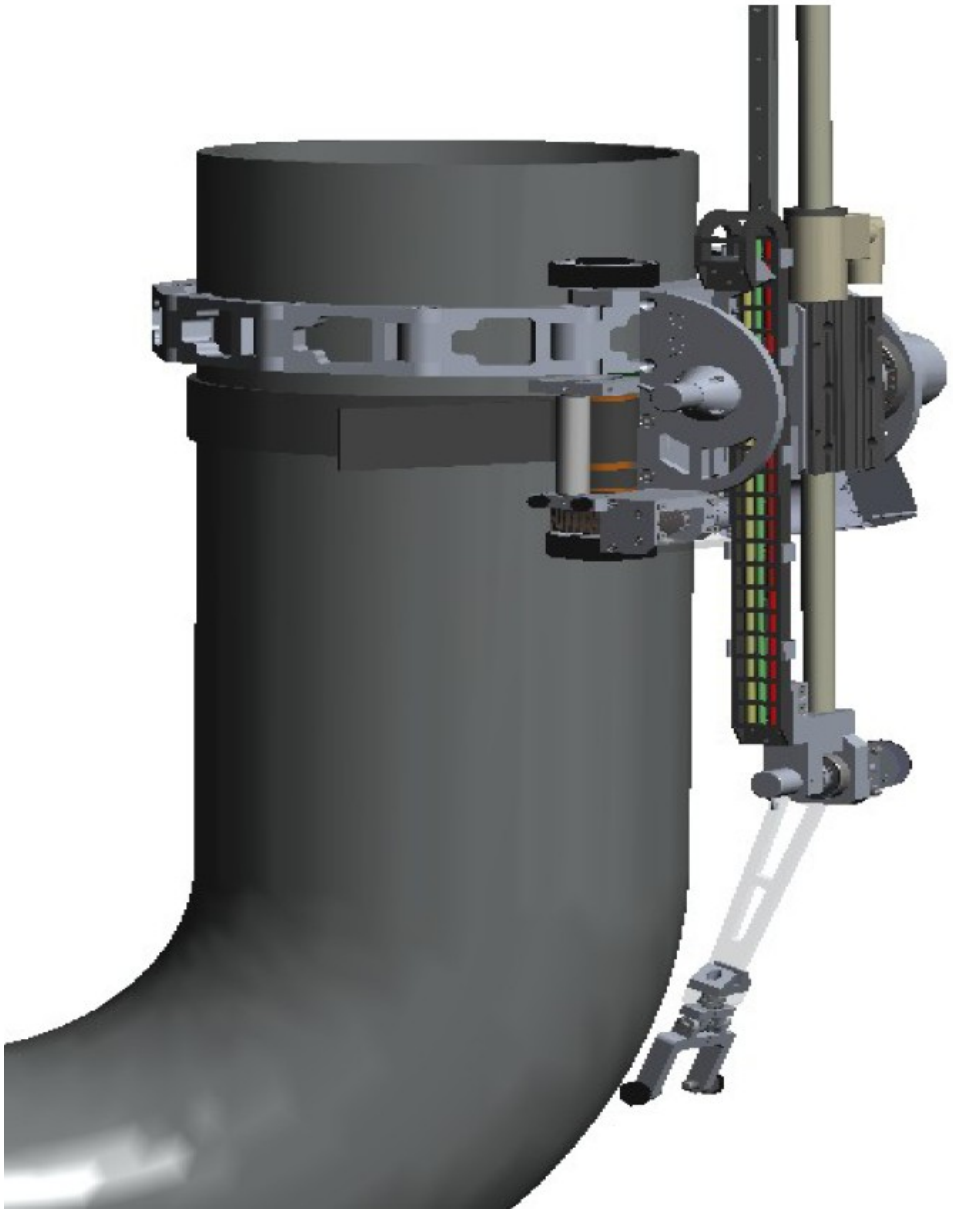
Manipulátor je tvořen třemi rotačními aktuátory a jedním aktuátorem prizmatickým (manipulátor typu **RRPR**). První kloub manipulátoru reprezentuje pojezd po potrubí kruhového průřezu. Koncový efektor manipulátoru umožňuje posunutí UZ sondy libovolně v prostoru (3 translační DoF) a její orientaci vzhledem k svislé ose pojezdu (1 rotační DoF). Obrázek 1 znázorňuje schématické uspořádání manipulátoru včetně zavedených souřadných systémů, viz [5], [10], dle D-H úmluvy, viz Tabulka. 1. CAD model prototypu manipulátoru je znázorněn na Obrázku 2.

LINK i	d_i	θ_i	a_i	α_i
1	0	$\theta_\mathbf{1}$	a_1	$-\frac{\pi}{2}$
2	0	$\theta_\mathbf{2}$	0	$\frac{\pi}{2}$
3	$\mathbf{d_3}$	0	0	$-\frac{\pi}{2}$
4	0	$\theta_\mathbf{4}$	a_4	0

Tabulka 1: D-H parametry souřadných systémů příslušných ramen, tučně vyznačeny aktivní kloubové souřadnice (aktuátory)



Obrázek 1: Sériový manipulátor - Varianta 2 (BEZ kompenzace polohy konc. efektoru)



Obrázek 2: CAD model prototypu manipulátoru

Kompenzace polohy koncového efektoru:

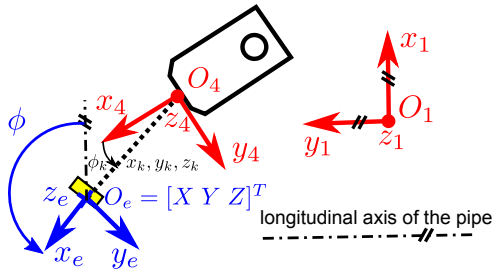
Vzhledem k tomu, že se předpokládá umístění UZ sondy v nějaké definované poloze, která nemusí být identická s umístěním posledního s.s. manipulátoru F_4, zavádíme tzv. kompenzaci polohy koncového efektoru T_e^4.[1] Jedná se o transformaci z polohy s.s. F_4 (posledního ramena manipulátoru) na s.s. F_e (s.s. koncového efektoru = s.s. UZ sondy), viz Obrázek 3.

Transformační matice $\boldsymbol{T}_e^4$:

$$\boldsymbol{T}_e^4 = \left[\begin{array}{ccc:c} & \boldsymbol{R}_e^4 & & \boldsymbol{O}_e^4 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right], \text{ kde } \boldsymbol{R}_e^4 = \begin{bmatrix} \cos(\phi_k) & -\sin(\phi_k) & 0 \\ \sin(\phi_k) & \cos(\phi_k) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{O}_e^4 = \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} \quad (1)$$

[1] $\boldsymbol{X}_a^b$ označuje bod či transformační matici $\boldsymbol{X}_a$ vyjádřená vzhledem k s.s. F_b

kde x_k, y_k, z_k definují posunutí s.s. konc. efektoru F_e vzhledem k s.s. posledního ramene manipulátoru F_4 a ϕ je úhel natočení s.s. F_e vzhledem k s.s. F_4 okolo osy $\boldsymbol{z}_4$.



Obrázek 3: Kompenzace polohy konc. efektoru

Polohy kloubů manipulátoru jsou určeny kloubovými souřadnicemi $\boldsymbol{Q}$ jako:

$$\boldsymbol{Q} = \left[\ \theta_1 \quad \theta_2 \quad d_3 \quad \theta_4\ \right]^T \tag{2}$$

Návrhové parametry manipulátoru $\boldsymbol{\xi}$ představují délky jednotlivých ramen a parametry kompenzace polohy koncového efektoru:

$$\boldsymbol{\xi} = \left[\ a_1 \quad a_4 \quad x_k \quad y_k \quad z_k \quad \phi_k\ \right]^T \tag{3}$$

kde a_1, a_4 a x_k, y_k, z_k a ϕ_k jsou parametry kompenzace polohy konc. efektoru (umístění NDT sondy), viz dále.

Poloha koncového efektoru lze popsat zobecněnými souřadnicemi $\boldsymbol{X}$:

$$\boldsymbol{X} = \left[\ X \quad Y \quad Z \quad \phi\ \right]^T \tag{4}$$

kde X, Y, Z jsou souřadnice bodu $\boldsymbol{O}_e$ vzhledem k s.s. F_0 a ϕ je vzájemné natočení s.s. F_e a F_1 okolo osy $\boldsymbol{z}_1$, tedy matice rotace $\boldsymbol{R}_e^1 = \mathrm{Rot}(\boldsymbol{z}, \phi)$.

Podrobné informace o kinematickém modelu manipulátoru lze nalézt v [8].

Definice významných souřadných systémů

- **Bázový s.s. (světový)** F_0; $\boldsymbol{O}_0 - \boldsymbol{x}_0\boldsymbol{y}_0\boldsymbol{z}_0$
  S.s., který je pevně spojen s potrubím, na kterém je manipulátor připoután. S.s. je určen v okamžiku nasazení manipulátoru na potrubí a to tak, že střed s.s. leží v průniku osy potrubí a roviny ležící kolmo k ose potrubí a procházející počátkem $\boldsymbol{O}_1$ s.s. F_1. Osa $\boldsymbol{z}_0$ leží ve směru osy potrubí s kladnou orientací od měřeného svaru a osa $\boldsymbol{x}_0$ leží s kladnou orientací ve směru pojezdu manipulátoru, tzn. prochází bodem $\boldsymbol{O}_1$.

- **S.s.koncového efektoru** F_e; $\boldsymbol{O}_e - \boldsymbol{x}_e\boldsymbol{y}_e\boldsymbol{z}_e$
  S.s. definující pozici a orientaci UZ sondy. Jeho poloha je ovlivněna nastavením kloubových souřadnic $\boldsymbol{Q}$ a parametry kompenzace polohy koncového efektoru x_k, y_k, z_k, ϕ_k. Polohové souřadnice X, Y, Z koncového efektoru jsou apriori vyjádřeny a uvažovány vzhledem k bázovému s.s. F_0. Orientace koncového efektoru ϕ je definována vzhledem k s.s. F_1.

Pro návrh realizace SW řídícího systému uvažovaného manipulátoru byly využity dosavadní zkušenosti s podobnými zařízeními. Řídící systém manipulátoru sestává z několika základních bloků. Grafické znázornění struktury SW řídícího systému lze graficky interpretovat Obrázkem 4. Ve schématu je znázorněna část hardwaru manipulátoru (HW manipulátoru), která není náplní

předkládané technické zprávy, nicméně hraje podstatnou roli pro pochopení řídícího systému jako celku. Více podrobností o hardwarovém vybavení prototypu manipulátoru lze nalézt v [7].

Software manipulátoru lze dále rozdělit na bloky realizující vlastní řízení a na bloky zprostředkovávající interakci řídícího systému s obsluhou, tzv. *uživatelské rozhraní - vizualizace*. Vlastní uživatelské rozhraní v tomto případě opět není součástí předkládané zprávy. Předpokládá se, že tento vizualizační systém (obecně HMI - Human Machine Interface) bude vyvíjen částečně odděleně od řídícího systému s respektováním definováním komunikačního rozhraní (standardní komunikační protokoly), viz [6]. Vizualizační systém může být implementován ve standardních nástrojích, např. Genesis32 od firmy ICONICS či ve formě „tenkých“ klientů v standardních prohlížečích (Mozilla Firefox, Google Chrome, atd.). Vzhledem k podstatě aplikace je pravděpodobné, že vizualizační systém bude umístěn na hardwarové platformě (PC, notebook), která bude oddělena od hardwaru řídícího počítače realizující vlastní řídící algoritmy.

2 Realizace softwaru řídícího systému

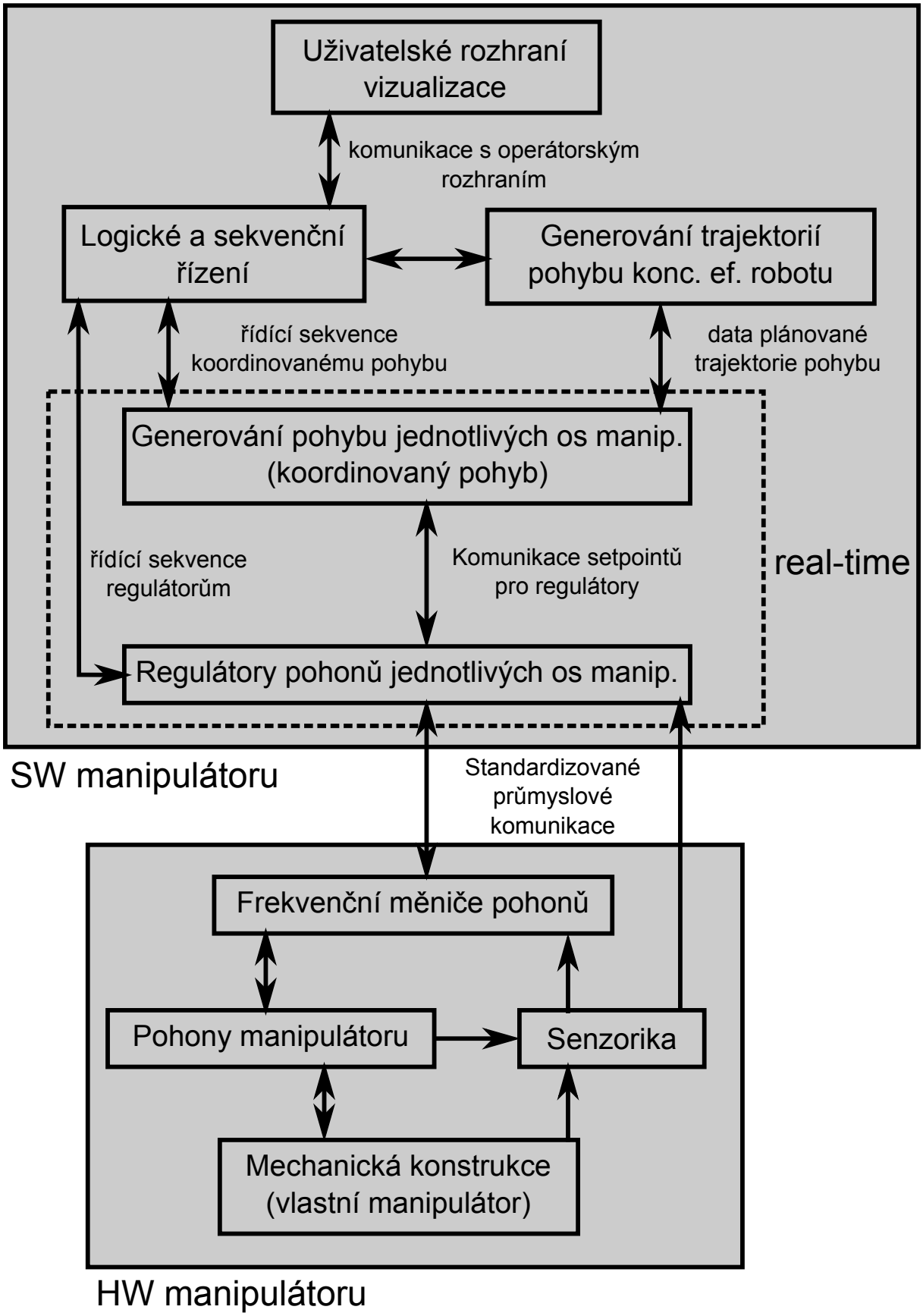
Celý řídící systém bude vyvíjen v řídícím systému reálného času REX, viz [4]. REX je vyspělý nástroj pro návrh a realizaci řídících algoritmů, který obsahuje celou řadu základních i velmi pokročilých funkcí pro realizaci algoritmů automatického řízení. Celý program řídícího algoritmu je realizován prostřednictvím rozsáhlé knihovny funkčních bloků, které mohou být ve vestavěném grafickém editoru přidávány do schématu řízení. Řídící systém REX podporuje celou řadu cílových platforem (windows XP/vista/7, Windows CE, GNU/Linux, atd.) a hardwarů (Alix PC Engines, Moxa, WinPAC, atd.). Samozřejmostí je podpora běžných průmyslových komunikačních protokolů (Ethernet POWERLINK, EtherCAT, Modbus TCP/IP, Modbus RTU, CAN, OPC Data Access 2.0 a 3.0, RS-232). V současnosti REX nabízí možnost realizace logického i sekvenčního řízení, řízení procesů prostřednictvím PID regulátorů (včetně automatických tunerů) i pokročilých regulátorů a filtrů. Systém REX je vybaven kvalitním nástrojem RexView pro servisní vizualizaci a správu algoritmu řízení s možností monitoringu a aktivního zásahu uživatele do procesu řízení. V nedávné době byl systém REX rozšířen o další knihovnu pokročilých bloků za účelem realizace algoritmů řízení pohybu, a to v jedné ose i více osách (analogie k systému vačkových mechanismů) a koordinovaného pohybu ve více osách (vhodné pro aplikace v robotice). S ohledem na univerzálnost a kompatibilitu jsou bloky knihovny pro řízení pohybu plně ve shodě s normou PLCopen Motion Control, viz [3]. Právě podpora bloků Motion Control dělá z řídícího systému REX silný nástroj, který lze použít pro realizaci řídících algoritmů koordinovaného pohybu uvažovaného manipulátoru. Bloky realizující vlastní řídící algoritmy jsou popsány v následujících kapitolách.

2.1 Logické a sekvenční řízení

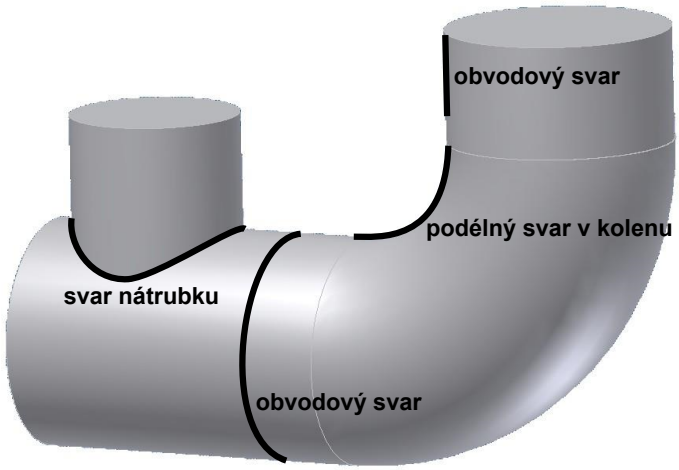
Realizuje základní řídící algoritmus logického a sekvenčního řízení. V tomto bloku jsou zpracovány povely z uživatelského rozhraní, které komunikují s dalšími bloky a zajišťují koordinovaný chod celého řízení. V současnosti jsou komponenty logického a sekvenčního řízení ve stádiu vývoje. Finální verze bude moci být plně dokončena až v okamžiku realizace prototypu a možnosti provádět zkušební experimenty. Blok logického a sekvenčního řízení bude realizován funkčními bloky systému REX.

2.2 Generování trajektorií pohybu koncového efektoru manipulátoru

Pro NDT testování svarů komplexních potrubních systémů v JE jsou uvažovány následující typy svarů: obvodový svar, podélný svar, podélný svar v kolenu a svar nátrubku, viz Obrázek 5.

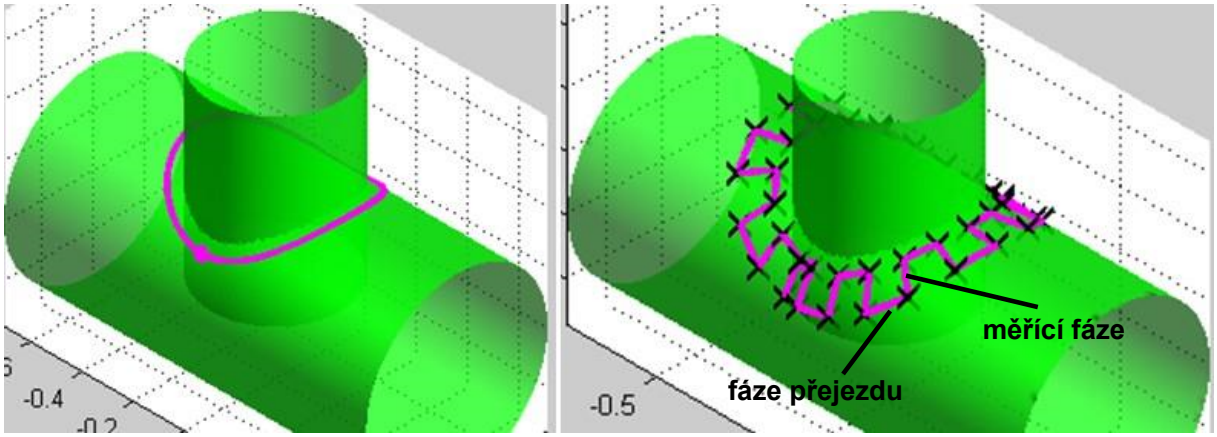


Obrázek 4: Schéma řídícího systému robotu



Obrázek 5: Uvažované typy svarů

S ohledem na předpokládané metody NDT je požadováno, aby manipulátor generoval dva základní pohyby podél testovaného svaru. *Jednoduchý pohyb*, který zajišťuje pouze posun UZ sondy podél svaru a samotné testování je realizováno UZ sondou vybavenou technologií Phased Array. Sonda Phased Array umožňuje realizovat pouze jednoduchý pohyb díky svému vnitřnímu uspořádání, kde je svazek UZ paprsků elektronicky rozmítán a tím je dosaženo požadovaného proskenování celé roviny svaru. V případě, že je použita pouze jednoduchá sonda, případně sonda s několika konstantními UZ paprsky, je nutné sondu polohovat tzv. *menadrovitým pohybem*, který se skládá z fáze přejezdu podél uvažovaného svaru a z fáze měření odpovídající kolmému pohybu na rovinu uvažovaného svaru. Jednoduchý i meandrovitý pohyb je znázorněn na Obrázku 6.



Obrázek 6: Uvažované typy pohybů podél svaru, jednoduchý pohyb (bez rozmítání) a meandrovitý pohyb (s rozmítáním) pro svar nátrubku

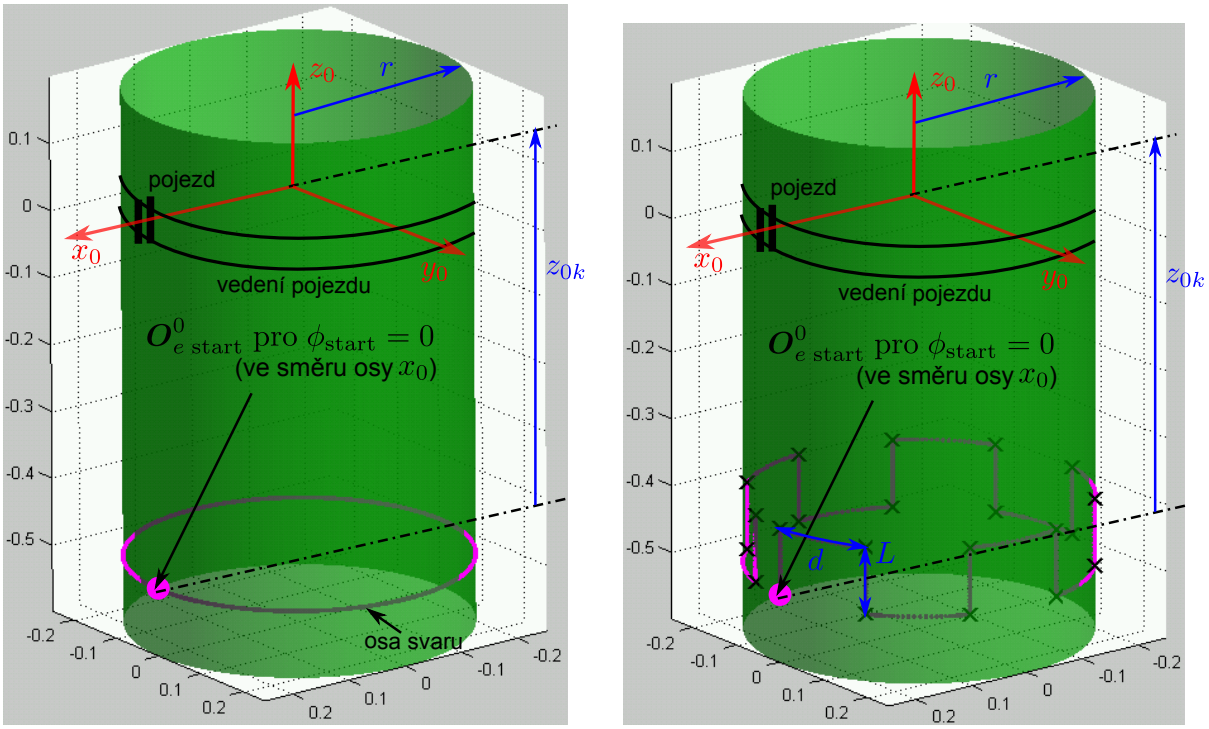
Algoritmy pro generátory trajektorií pro jednotlivé typy svarů byly implementovány v programovém prostředí Matlab [1] ve formě funkcí. Předpokládá se, že prostřednictvím těchto funkcí budou vygenerovány požadované polohy koncového efektoru manipulátoru pro všechny uvažované svary (typy a rozměry). Tato data budou uložena ve formě datových souborů a následně volána v řídícím algoritmu manipulátoru. Funkce pro generování příslušných trajektorií svarů jsou uvedeny dále. Podrobný popis jejich implementace lze nalézt v [9].

2.2.1 Obvodový svar

Parametry obvodového svaru, viz 7, a trajektorie (rozmítání) jsou dány jako: (pokud $L = 0$ předpokládá se generování trajektorie bez rozmítání)

$$\boldsymbol{\xi_{os}} = \{\ R \quad z_{0k} \quad d \quad L \quad N \quad \phi_{\text{start}} \text{ nebo } \boldsymbol{O}^0_{e\,\text{start}} \quad v_{max_{prejezd}} \quad a_{max_{prejezd}}, \quad v_{max_{mereni}} \quad a_{max_{mereni}}\ \}$$

R ..... poloměr válce, na kterém je nasazen manipulátor
z_{0k} ..... vzdálenost pojezdu manipulátoru od svaru
d ..... vzdálenost mezi úseky rozmítání podél trajektorie svaru
L ..... délka úseku rozmítání
N ..... časová diference mezi generovanými body (rozlišení generování trajektorie)
ϕ_{start}, $\boldsymbol{O}^0_{e\,\text{start}}$ ... parametr určující počáteční bod generování trajektorie
$v_{max_{prejezd}}$ ..... max. rychlost přejezdu mezi úseky měření
$a_{max_{prejezd}}$ ..... max. tečné zrychlení přejezdu mezi úseky měření
$v_{max_{mereni}}$ ..... max. rychlost měření
$a_{max_{mereni}}$ ..... max. tečné zrychlení měření



(a) Bez rozmítání (b) S rozmítáním

Obrázek 7: Generovaná trajektorie obvodového svaru

Implementovaná funkce v Matlabu: Vstupem funkce jsou výše uvedené parametry a výstupem funkce je poloha, rychlost a zrychlení pozice (translační) koncového efektoru manipulátoru $\boldsymbol{O}_e^0$. Orientace ϕ je v tomto případě volena konstantní. V případě rozmítání je navíc vracen vektor indexů reprezentující pořadí bodu polohy konc. efektoru $\boldsymbol{O}_e^0$, ve které dochází k rozmítání.

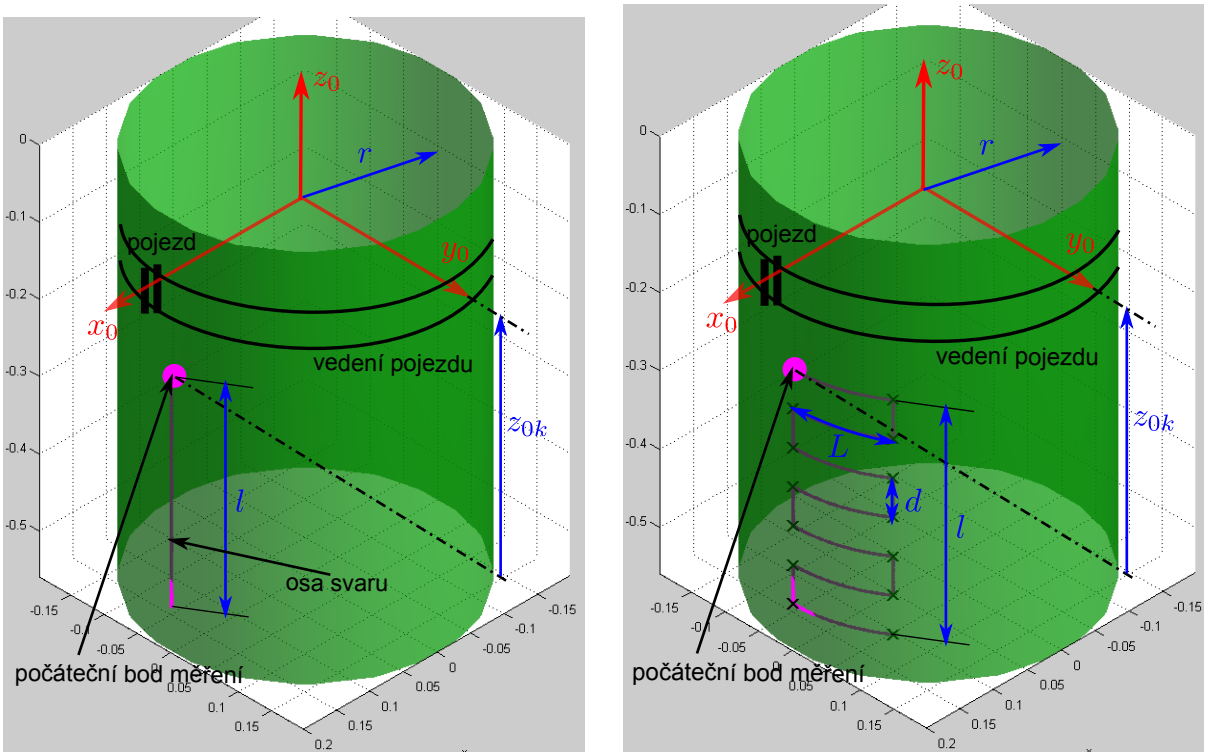
```
function varargout = trajectoryGenerator_obvodovySvar(par,Oe0_start,constraints)
% generator trajektorie obvodoveho svaru
%
% parametry obvodoveho svaru - par
% R ... polomer potrubi
% z0 ... vzdalenost pojezdu manipulatoru od trajektorie svaru
%
% parametry trajektorie
% d ... vzdalenost mezi rozmitanim na trajektorii natrubku
% L ... delka primky rozmitani, pro L = 0 ... bez rozmitani
% N ... casova diference mezi interpolovanymi body (rozliseni algoritmu)
%
% Oe0_start ... pocatecni bod mereni vzhhledem k s.s. F0 (nemusi lezet presne na obvodovem sv
% pokud length(Oe0_start) = 1 ... Oe0_start ... uhel posunu pocatecniho
% bodu mereni kolem z0
%
% varargout = {[Oe0;dOe0;ddOe0],time,Oe0_index_prejezd}
% Oe0 ... vektor bodu (polohy) koncoveho efektoru manipulatoru vzhledem k
% s.s. F0
% time ... vektor casovych okamziku v generovanych bodech, vzhledem na
% pozadovana omezeni constraints
% Oe0_index_prejezd ... indexy bodu na zmenach mezi rozmitanim a prejezdy
% (pouze ve variante s rozmitanim)
%
% constraints = [v_max_prejezd;a_max_prejezd;v_max_mereni;a_max_mereni] ...
% omezeni na generovani trajektorie
```

2.2.2 Podélný svar

Parametry podélného svaru, viz 8, a trajektorie (rozmítání) jsou dány jako: (pokud $L = 0$ předpokládá se generování trajektorie bez rozmítání)

$$\boldsymbol{\xi_{ps}} = \{\ r \quad z_{0k} \quad l \quad d \quad L \quad N \quad v_{max_{prejezd}} \quad a_{max_{prejezd}}, \quad v_{max_{mereni}} \quad a_{max_{mereni}}\ \}$$

r ..... poloměr potrubí
z_{0k} ..... vzdálenost pojezdu manipulátoru od počátku svaru
l ..... délka svaru
d ..... vzdálenost mezi úseky rozmítání podél trajektorie svaru
L ..... délka úseku rozmítání
N ..... časová diference mezi generovanými body (rozlišení generování trajektorie)
$v_{max_{prejezd}}$ ..... max. rychlost přejezdu mezi úseky měření
$a_{max_{prejezd}}$ ..... max. tečné zrychlení přejezdu mezi úseky měření
$v_{max_{mereni}}$ ..... max. rychlost měření
$a_{max_{mereni}}$ ..... max. tečné zrychlení měření



(a) Bez rozmítání (b) S rozmítáním

Obrázek 8: Generovaná trajektorie podélného svaru

Implementovaná funkce v Matlabu: Vstupem funkce jsou výše uvedené parametry a výstupem funkce je poloha, rychlost a zrychlení pozice (translační) koncového efektoru manipulátoru $\boldsymbol{O}_e^0$. Orientace ϕ je v tomto případě volena konstantní. V případě rozmítání je navíc vracen vektor indexů reprezentující pořadí bodu polohy konc. efektoru $\boldsymbol{O}_e^0$, ve které dochází k rozmítání.

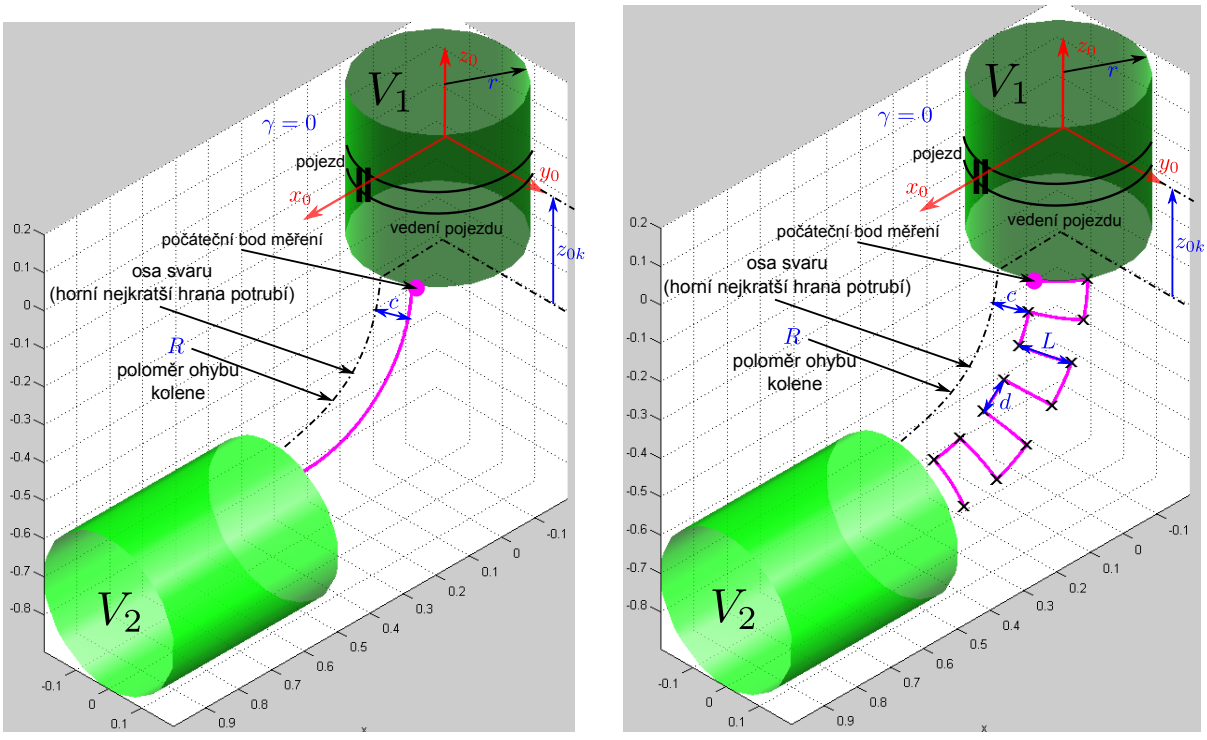
```
function varargout = trajectoryGenerator_podelnySvar(par,constraints)
% generator trajektorie podelneho svaru
%
% parametry podelneho svaru v kolenu - par
% r ... polomer potrubi
% z0 ... vzdalenost pojezdu manipulatoru od pocatku trajektorie svaru
% l ... delka podelneho svaru
%
% parametry trajektorie
% d ... vzdalenost mezi rozmitanim
% L ... delka rozmitani, pro L = 0 ... bez rozmitani
% N ... casova diference mezi generovanymi body (perioda vzorkovani)
%
% varargout = {[Oe0;dOe0;ddOe0],time,Oe0_index_prejezd}
% Oe0 ... vektor bodu (polohy) koncoveho efektoru manipulatoru vzhledem k
% s.s. F0
% time ... vektor casovych okamziku v generovanych bodech, vzhledem na
% pozadovana omezeni constraints
% Oe0_index_prejezd ... indexy bodu na zmenach mezi rozmitanim a prejezdy
% (pouze ve variante s rozmitanim)
%
% constraints = [v_max_prejezd;a_max_prejezd;v_max_mereni;a_max_mereni] ...
% omezeni na generovani trajektorie
```

2.2.3 Podélný svar v kolenu

Parametry podélného svaru v kolenu, viz 9, a trajektorie (rozmítání) jsou dány jako: (pokud $L = 0$ předpokládá se generování trajektorie bez rozmítání)

$$\boldsymbol{\xi_{psk}} = \{\ R \quad r \quad z_{0k} \quad \gamma \quad d \quad L \quad c \quad N \quad v_{max_{prejezd}} \quad a_{max_{prejezd}}, \quad v_{max_{mereni}} \quad a_{max_{mereni}}\ \}$$

R ..... vnitřní poloměr ohybu kolene
r ..... poloměr potrubí
z_{0k} ..... vzdálenost pojezdu manipulátoru od počátku svaru
γ ..... natočení osy potrubí V_1 okolo osy $\boldsymbol{z}_0$
d ..... vzdálenost mezi úseky rozmítání podél trajektorie svaru
L ..... délka úseku rozmítání
c ..... vzdálenost počátku oblouku rozmítání od svaru
N ..... časová diference mezi generovanými body (rozlišení generování trajektorie)
$v_{max_{prejezd}}$ ..... max. rychlost přejezdu mezi úseky měření
$a_{max_{prejezd}}$ ..... max. tečné zrychlení přejezdu mezi úseky měření
$v_{max_{mereni}}$ ..... max. rychlost měření
$a_{max_{mereni}}$ ..... max. tečné zrychlení měření



(a) Bez rozmítání (b) S rozmítáním

Obrázek 9: Generovaná trajektorie podélného svaru v kolenu

Implementovaná funkce v Matlabu: Vstupem funkce jsou výše uvedené parametry a výstupem funkce je poloha, rychlost a zrychlení pozice (translační) koncového efektoru manipulátoru $\boldsymbol{O}_e^0$ a jeho orientace ϕ. V případě rozmítání je navíc vracen vektor indexů reprezentující pořadí bodu polohy konc. efektoru $\boldsymbol{O}_e^0$, ve které dochází k rozmítání.

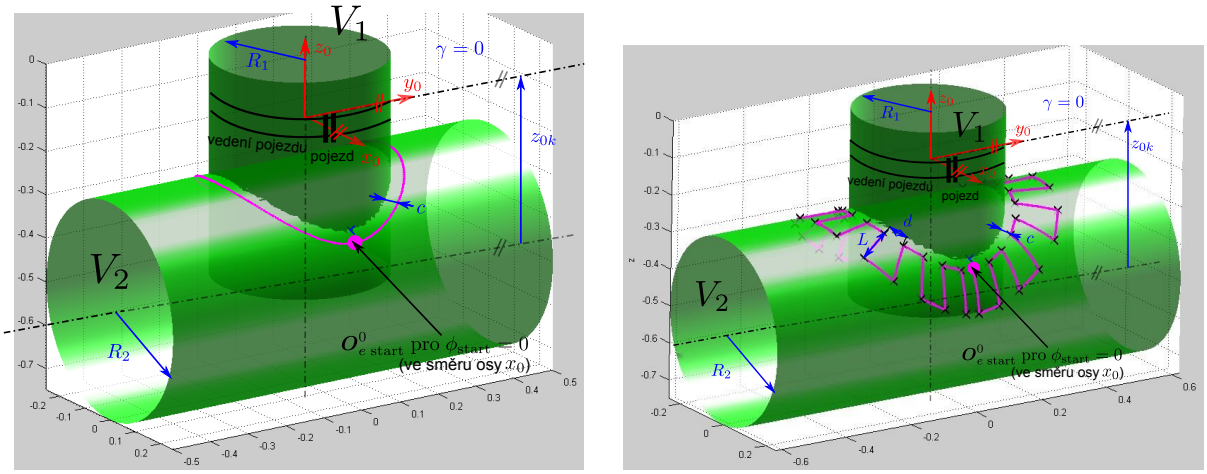
```
function varargout = trajectoryGenerator_podelnySvarVKolenu(par,constraints)
% generator trajektorie podelneho svaru v kolenu
%
% parametry podelneho svaru v kolenu - par
% R ... radius ohybu kolene (vnitrni)
% r ... polomer potrubi
% z0 ... vzdalenost pojezdu manipulatoru od pocatku trajektorie svaru
% gamma ... natoceni kolene, dane pripevnenim manipulatoru k potrubi
%
% parametry trajektorie
% d ... vzdalenost mezi rozmitanim na trajektorii podelneho svaru v kolenu
% L ... delka primky rozmitani, pro L = 0 ... bez rozmitani
% c ... vzdalenost pocatku oblouku rozmitani od trajektorie podelneho svaru
% v kolenu
% N ... casova diference mezi generovanymi body (perioda vzorkovani)
%
% varargout = {[Oe0;dOe0;ddOe0],time,Oe0_index_prejezd}
% Oe0 = [Oe0x;Oe0y;Oe0z;phi] ... vektor bodu (polohy + orientace)
% koncoveho efektoru manipulatoru vzhledem k s.s. F0
% phi ... uhel odklonu od osy x0
% time ... vektor casovych okamziku v generovanych bodech, vzhledem na
% pozadovana omezeni constraints
% Oe0_index_prejezd ... indexy bodu na zmenach mezi rozmitanim a prejezdy
% (pouze ve variante s rozmitanim)
%
% constraints = [v_max_prejezd;a_max_prejezd;v_max_mereni;a_max_mereni] ...
% omezeni na generovani trajektorie
```

2.2.4 Svar nátrubku

Parametry svaru nátrubku, viz 10, a trajektorie (rozmítání) jsou dány jako: (pokud $L = 0$ předpokládá se generování trajektorie bez rozmítání)

$$\boldsymbol{\xi_{nat}} = \{\ R_1 \quad R_2 \quad z_{0k} \quad \gamma \quad d \quad L \quad c \quad N \quad \phi_{\text{start}} \text{ nebo } \boldsymbol{O}^0_{e\ \text{start}}\ \}$$

R_1 ..... poloměr válce, na kterém je nasazen manipulátor
R_2 ..... poloměr průběžného válce
γ ..... natočení válcové plochy průběžného potrubí okolo osy $\boldsymbol{z}_0$
z_{0k} ..... vzdálenost pojezdu manipulátoru od osy průběžného válce
d ..... vzdálenost mezi úseky rozmítání podél trajektorie průsečíku válcových ploch
L ..... délka úseku rozmítání
c ..... vzdálenost trajektorie od průsečíku válcových ploch
N ..... vzdálenost mezi generovanými body (rozlišení generování trajektorie)
ϕ_{start}, $\boldsymbol{O}^0_{e\ \text{start}}$ ... parametr určující počáteční bod generování trajektorie



(a) Bez rozmítání

(b) S rozmítáním

Obrázek 10: Generovaná trajektorie svaru nátrubku

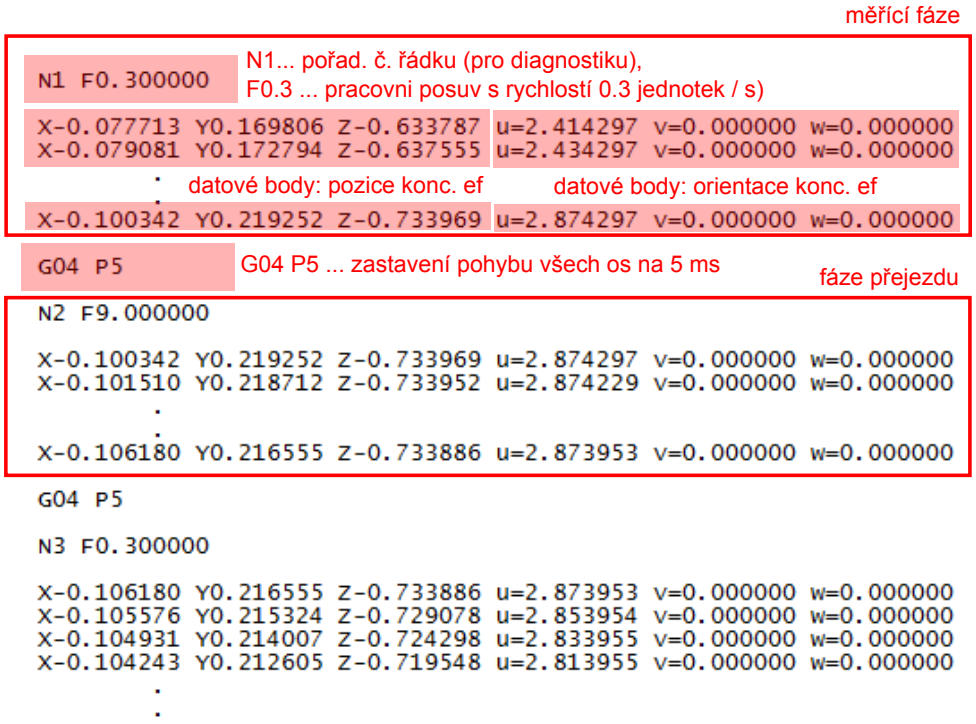
Implementovaná funkce v Matlabu: Vstupem funkce jsou výše uvedené parametry a výstupem funkce je poloha (translační) koncového efektoru manipulátoru $\boldsymbol{O}_e^0$ a jeho orientace ϕ. V případě rozmítání je navíc vracen vektor indexů reprezentující pořadí bodu polohy konc. efektoru $\boldsymbol{O}_e^0$, ve které dochází k rozmítání. Funkce generující pohyb po trajektorii nátrubku vrací, na rozdíl od výše uvedených případů, pouze polohové souřadnice konc. efektoru manipulátoru.

```
function varargout = trajectoryGenerator_natrubek(par,Oe0_start)
% generator trajektorie svaru natrubku - par
%
% parametry natrubku
% R1 ... polomer potrubi, na kterem je pripoutan robot
% R2 ... polomer prubezneho potrubi, R1 < R2
% z0 ... vzdalenost pojezdu manipulatoru od osy prubezneho potrubi (dane
% pripevnenim manipulatoru)
% gamma ... natoceni podelneho potrubi v ose z0 (dane pripevnenim manipulatoru)
%
% parametry trajektorie
% d ... vzdalenost mezi rozmitanim na trajektorii natrubku (priblizne)
% L ... delka oblouku rozmitani (priblizne), pro L = 0 ... bez rozmitani
% c ... vzdalenost pocatku oblouku rozmitani od trajektorie natrubku (priblizne)
% N ... priblizna vzdalenost mezi interpolacnimi body (rozliseni generovani
% trajektorie)
% Oe0_start ... pocatecni bod mereni vzhhledem k s.s. F0 (nemusi lezet presne na natrubku)
% pokud length(Oe0_start) = 1 ... Oe0_start ... uhel posunu pocatecniho
% bodu mereni kolem z0
%
% varargout = [Oe0,Oe0_index_prejezd]
% Oe0 ... vektor bodu (polohy) koncoveho efektoru manipulatoru vzhhledem k
% s.s. F0
% Oe0_index_prejezd ... indexy bodu na zmenach mezi rozmitanim a prejezdy
```

2.2.5 Implementace generátoru trajektorie v řídícím systému

Generátory trajektorie obvodového svaru, podélného svaru a podélného svaru v kolenu poskytují kompletní informaci o průběhu pohybu koncového efektoru ve smyslu požadované polohy, rychlosti a zrychlení podél trajektorie svaru s omezením na max. rychlost a zrychlení. Z důvodů uvedených ve zprávě [8] nelze jednoduše analyticky generovat průběhy rychlostí a zrychlení koncového efektoru manipulátoru v případě svaru nátrubku (nepřirozená parametrizace křivek). Vzhledem k podstatě úlohy NDT lze předpokládat, že rychlosti a zrychlení konc. efektoru manipulátoru budou relativně nízké a celá úloha se velmi podobá problematice víceosého obrábění. Z toho vyplývá, že generování požadované rychlosti a zrychlení jako veličin pro dopřednou složku ve standardním kaskádním uspořádání regulátorů jednotlivých aktuátorů manipulátoru (regulátor polohy - rychlosti - zrychlení/proudu) nebude třeba využívat (příslušné regulátory v kaskádním zapojení budou poskytovat dostatečně přesnou regulaci). Předpokládejme proto, že dále využijeme pouze datové body trajektorie příslušného svaru reprezentující požadovanou polohu $\boldsymbol{O}_e^0$ a orientaci ϕ koncového efektoru. V takovém případě je tedy podobnost k víceosému obrábění velmi blízká a nabízí se možnost využít standardního postupu plánování pohybu koncového efektoru manipulátoru ze zadaných datových bodů prostřednictvím tzv. *G-kódu*, viz [2]. G-kód obsahuje řídící příkazy (rychlost posuvu, prodlevu mezi posuvy, atd.) a datové body, které mají být projety konc. efektorem manipulátoru. Funkční blok, který generuje pohyb ze zápisu v G-kódu je implementován v řídícím systému REX pod názvem *RM_ Gcode*. Blok interně provádí interpolaci mezi zadanými datovými body s ohledem na předepsanou rychlost posuvu a maximální povolené zrychlení.

Zápis programu v G-kódu z datových bodů získaných z generátorů trajektorií vypadá následovně:



Obrázek 11: Příklad zápisu G-kódu v bloku RM_Gcode pro meandrovitý pohyb konc. ef. manipulátoru ($\boldsymbol{O}_e^0 = [X, Y, Z]$, $\phi = u$)

Výsledný blok generování trajektorií pohybu konc. ef. robotu bude tedy realizován offline a bude poskytovat soubor datových podkladů pro pohyb koncového efektoru v zobecněných souřadnicích (pozice a orientace konc. efektoru ve formátu G-kódu) pro jednotlivé typy a parametry svarů.

Tato data budou interpretována v real-time části řídícího systému pomocí bloku *RM_Gcode* řídícího systému REX.

2.3 Real-time část řídícího systému (koordinovaný pohyb manipulátoru)

Koordinovaný pohyb manipulátoru spočívá v řízení jeho jednotlivých kloubů takovým způsobem, aby byl zajištěn pohyb koncového efektoru podél požadované trajektorie. Jinými slovy, k realizaci koordinovaného pohybu je nezbytná znalost tzv. inverzního geometrického modelu (IGM) - pro zobrazení z prostoru zobecněných souřadnic $\boldsymbol{X}$ do prostoru kloubových souřadnic $\boldsymbol{Q}$, případně inverzní okamžité kinematické úlohy (IOKÚ) - pro zobrazení z prostorů rychlostí $\dot{\boldsymbol{X}}$ resp. zrychlení $\ddot{\boldsymbol{X}}$ zobecněných souřadnic do prostoru rychlostí $\dot{\boldsymbol{Q}}$ resp. zrychlení $\ddot{\boldsymbol{Q}}$ souřadnic kloubových. Matematicky je možné tyto relace zapsat jako:

$$\boldsymbol{Q} = \mathbf{F}^{-1}(\boldsymbol{X}) \quad \text{IGM} \tag{5}$$

$$\dot{\boldsymbol{Q}} = \boldsymbol{J}^{-1}(\boldsymbol{Q}) \cdot \dot{\boldsymbol{X}} \quad \text{(IOKÚ - rychlosti)} \tag{6}$$

$$\ddot{\boldsymbol{Q}} = \dot{\boldsymbol{J}}^{-1}(\boldsymbol{Q}, \dot{\boldsymbol{Q}}) \cdot \dot{\boldsymbol{X}} + \boldsymbol{J}^{-1}(\boldsymbol{Q}) \cdot \ddot{\boldsymbol{X}} \quad \text{(IOKÚ - zrychlení)} \tag{7}$$

kde $\boldsymbol{J}(\boldsymbol{Q}) = \frac{\partial \mathbf{F}(\boldsymbol{Q})}{\partial \boldsymbol{Q}}$ resp. $\dot{\boldsymbol{J}} = \frac{\partial^2 \mathbf{F}(\boldsymbol{Q})}{\partial \boldsymbol{Q}^2}$ je kinematický jakobián resp. jeho časová derivace a funkce $\mathbf{F}(\boldsymbol{Q})$ je přímý geometrický model *DGM* manipulátoru. Více informací a úplné matematické odvození lze nalézt [9].

Koordinovaný pohyb je realizován v řídícím systému REX pomocí knihovny funkčních bloků Motion Control, viz [4]. Stručný popis funkcí jednotlivých bloků je následující:

- **RM_Axis**
  Základní blok pro nastavení jednotlivých os manipulátoru (aktivní klouby, pohony). Představuje sdílenou strukturu, ve které jsou uloženy všechny stavy a parametry osy. Realizuje základní funkce požadované normou PLCopen Motion Control, např. kontrola nastavených mezí polohy, rychlosti, zrychlení, jerku, ošetření havarijních stavů - odchylka skutečného a požadovaného stavu osy ⇒ zajištění správné funkce regulátoru (není součástí tohoto bloku) ⇒ havarijní zastavení, atd. Blok je použit pro každou osu nezávisle (neřeší koordinovaný pohyb)

- **RM_Track**
  Implementuje některé funkce pro nezávislé řízení jedné osy, např.: najetí na požadovanou polohu, rychlost, pojíždění s osou v ručním režimu pomocí tlačítek NAHORU/DOLŮ. To vše s respektováním zadaných limitů na max. rychlost, zrychlení, jerk.

- **MCP_MoveAbsolute**
  Pohyb osy do zadané absolutní polohy s respektováním omezení na rychlost, zrychlení, jerk. V bloku jsou implementovány pokročilé funkce pro správu řízení postupného pohybu os (převzetí řízení os, napojování pohybu os, atd.).

- **MC_Power**
  Blok musí být použit s každou osou a jako jediný převádí osu ze stavu *disabled* do stavu *StandStill*, t.j. osa je aktivní. V případě vypnutí osy v okamžiku, kdy je aktivní (vykonává nějaký pohyb) je nejprve vykonána zastavovací sekvence (dle parametrů nastavených v bloku RM_Axis).

- **MC_Home**
  Blok spolupracující s blokem RM_Axis definuje, jakým způsobem bude nastaven výchozí bod pohybu osy, případně zajišťuje sekvenci tzv. homování osy (např. dle koncových spínačů, mechanického dorazu, atd.) V našem případě slouží po zapnutí napájení k načtení polohy absolutního snímače polohy v jednotlivých pohonech.

- **RM_AxesGroup**
  Základní blok pro koordinovaný pohyb. Představuje sdílenou strukturu, ve které jsou uloženy všechny stavy a parametry skupiny os. Blok zajišťuje kontrolu nastavených mezí rychlosti, zrychlení, jerku, havarijní zastavení, předávání dat z a do bloku RM_Axis podřízených os. Definuje počet translačních a rotačních os kvůli interpolaci pohybu dle předepsaných profilů.

- **MC_AddAxisToGroup**
  Blok přidává vybranou osu do skupiny za účelem realizace koordinovaného pohybu skupiny přidaných os.

- **MC_SetKinTransform_NDT**
  Blok realizující kinematickou transformaci, respektive IGM a IOKÚ, viz rovnice (5 - 7). Parametry bloku jsou návrhové geometrické parametry manipulátoru.

- **MC_GroupEnable**
  Blok aktivuje skupinu os. Koordinovaný pohyb je potom realizován prostřednictvím vygenerovaného G-kódu s pomocí bloku RM_Gcode.

- **RM_GroupTrack**
  Blok vykonávající podobné funkce jako blok RM_Track, ovšem pro koordinovaný pohyb os. Jinými slovy, je možné manuálně přejíždět s osami manipulátoru koordinovaně dle zadané hodnoty polohy/rychlosti zobecněných souřadnic manipulátoru.

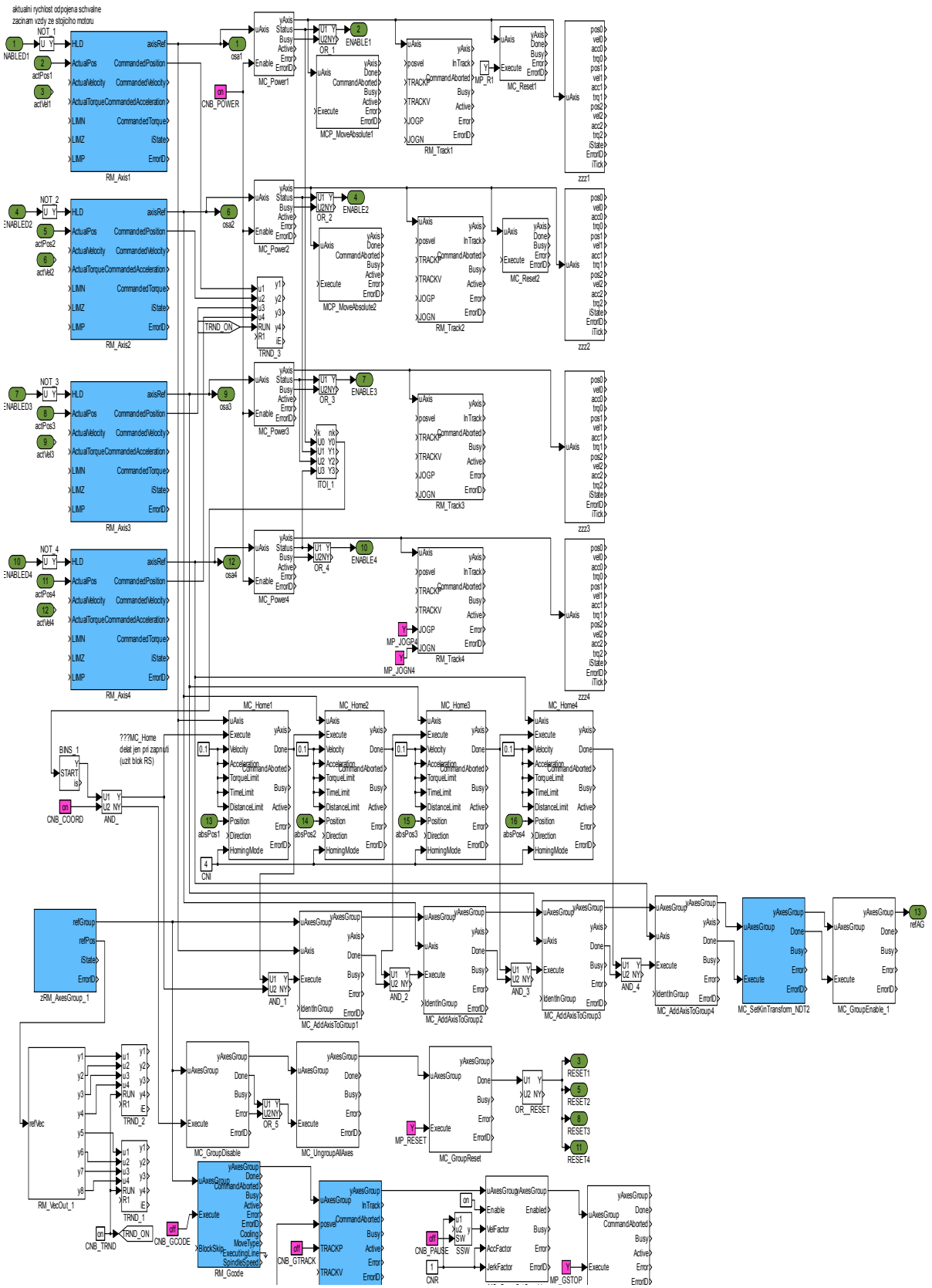
- **MC_GroupSetOverride**
  Blok slouží k zastavení pohybu manipulátoru prostřednictvím přenásobení aktuální požadované rychlosti/zrychleni/jerku zadaným faktorem. Např. pro faktor roven nule je pohyb zastaven a při opětovném nastavení nenulové hodnoty je pohyb opět obnoven z aktuální pozice, tzn. pohyb není přerušen a nezačíná tak od začátku sekvence.

- **MCP_GroupHalt**
  Blok sloužící ke koordinovanému zastavení všech os ve skupině. Dojde k zastavení pohybu dle zadaných omezení na tečnou rychlost, zrychlení a jerk ve směru původního pohybu. Pokud toto nelze realizovat, je aktivováno nezávislé zastavení v rámci jednotlivých os (nekoordinovaně).

- **RM_VecOut, RM_Vector**
  Bloky sloužící k zadávání či získávání vektoru dat (polohy, rychlosti, atd.) ze skupiny os.

- **RM_AxisOut**
  Blok, který zpřístupňuje důležité stavy bloku RM_Axis.

Výčet obsahuje pouze nejdůležitější použité bloky knihovny PLCopen Motion Control pro koordinovaný pohyb více os implementované v řídícím systému REX. Více informací a popisu ostatních bloků ve schématu řízení, viz Obrázek 12, lze nalézt v uživatelské příručce systému REX [4] či přímo ve specifikaci PLCopen [3].



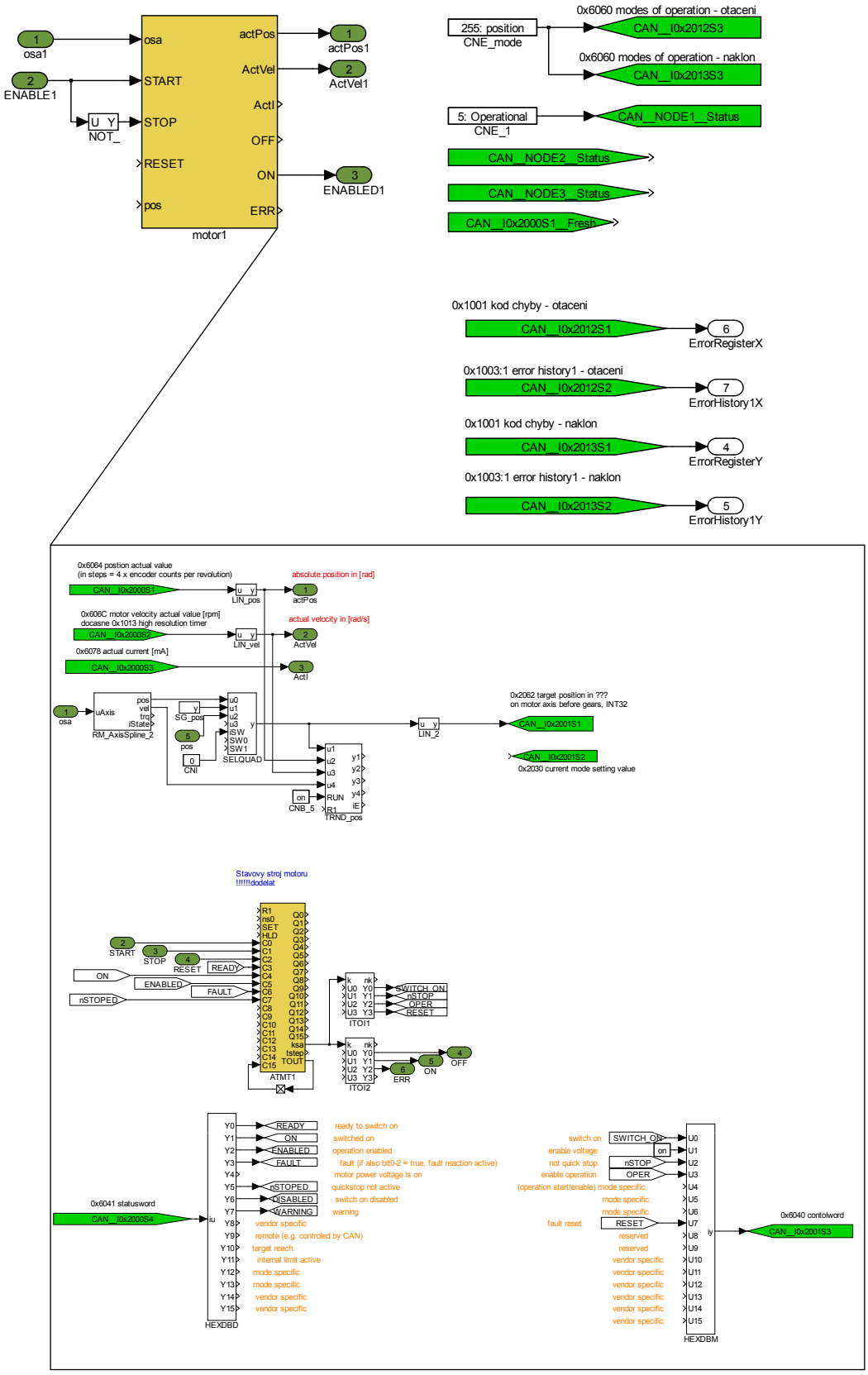
Obrázek 12: Blokové schéma hlavního úlohy (tasku, viz [4]) řízení manipulátoru v řídícím systému REX

Kromě hlavního tasku na Obrázku 12 musí konfigurace řídícího systému REX obsahovat ještě

task pro vlastní řízení pohonů manipulátoru, viz Obrázek 13 a ovladače pro komunikaci s reálnými pohony prostřednictvím sběrnice Ethernet POWERLINK, CAN či další komunikace pro sběr dat ze snímačů. Více podrobností ke konfiguraci ovladačů v systému REX lze nalézt v [4] a v příslušných manuálech hardwarového vybavení manipulátoru [7]. Regulátory pohonů jednotlivých os manipulátoru, viz Obrázek 4, můžou být implementovány jako součást řídícího systému REX, nebo je možné využít standardní PID kaskádní regulace (regulátor polohy, rychlosti, proudu) implementované ve firmwaru řídících jednotek pohonů, viz HW manipulátoru [7]. V současné konfiguraci v řídícím systému REX jsou regulátory skutečně vynechány a je využito jejich implementace v řídících jednotkách. Řídící systém REX posílá prostřednictvím komunikačního protokolu pouze požadované polohy aktuátorů manipulátoru.

3 Závěr

V předložené zprávě je ve stručnosti shrnut základní přístup k realizaci řídícího softwaru vyvíjeného manipulátoru pro NDT svarů komplexních potrubních systémů JE. SW řídícího systému je rozdělen do několika bloků, které implementují jeho dílčí funkční celky. Pro celý řídící systém se předpokládá, že subsystém uživatelského rozhraní a vizualizace bude implementován na cílové platformě (PC, notebook) odlišné od platformy (Alix, Moxa, atd.) vlastního real-time řídícího SW. Vzájemná komunikace bude realizována na principu standardních komunikačních protokolů používaných v systémech HMI. Bloky generování trajektorií pohybu konc. ef. manipulátoru budou poskytovat datové soubory pro uvažovanou množinu testovaných svarů (typ svaru, velikost, lokalizace, atd.) ve formátu G-kódu. Tato data budou dále interpretovány real-time částí řízení. Celý řídící systém zahrnující logické a sekvenční řízení a real-time část (generování pohybu jednotlivých os manipulátoru) bude realizován řídícím systémem REX. Pro řízení pohybu (coordinovaný pohyb ve více osách) budou využity bloky knihovny Motion Control (MC_SINGLE, MC_COORD) jejichž implementace je ve shodě s normou PLCopen Motion Control.



Obrázek 13: Blokové schéma tasku řízení jednoho motoru manipulátoru v řídícím systému REX

Reference

[1] The MathWorks, Matlab. http://www.mathworks.com/.

[2] Oberg, E.; Jones, F.; Horton, H.; aj.: *Machinery's Handbook: A Reference Book for the Mechanical Engineer, Designer, Manufacturing Engineer, Draftsman, Toolmaker, and Machinist*. Machinery's Handbook: A Reference Book for the Mechanical Engineer, Designer, Manufacturing Engineer, Draftsman, Toolmaker, and Machinist, Industrial Press, 2004, ISBN 9780831127374.
URL `http://books.google.cz/books?id=31eWvgTgD7oC`

[3] PLCopen: Motion Control. 2013.
URL `http://www.plcopen.org/pages/tc2_motion_control/index.htm`

[4] REX Controls s.r.o.: Řídící systém REX. 2013.
URL `http://www.rexcontrols.cz`

[5] Sciavicco, L.; Siciliano, B.: *Modelling and Control of Robot Manipulators*. Springer, druhé vydání, 2000, ISBN 978-1-85233-221-1.

[6] Severa, O.: Nástroje pro ovládání a diagnostiku manipulátoru. Technická zpráva, Katedra kybernetiky, FAV, ZČU v Plzni, 2013.

[7] Uhlíř, J.; Vlček, P.: Hardware manipulátoru Sáva. Technická zpráva, ÚJV Řež, a.s., 2013.

[8] Švejda, M.: Kinematická a dynamická analýza robotických architektur pro potreby moderních ultrazvukových kontrol svarových spoju komplexních potrubních systému jaderných elektráren. 2011, Katedra kybernetiky, ZČU v Plzni.
URL `http://home.zcu.cz/~msvejda/_publications/2011/2_UJVKinematikaManipulatoru.pdf`

[9] Švejda, M.: Manipulátor pro NDT, varianta 2: RRPR manipulátor, implementacní poznámky. 2013, Katedra kybernetiky, ZČU v Plzni.
URL `http://home.zcu.cz/~msvejda/_publications/2013/6_generatoryUJVrobot.pdf`

[10] W. Khalil, E. D.: *Modeling, Identification and Control of Robots*. Butterworth-Heinemann, 2004, ISBN 190399666X.