STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 9: Strojírenství, hutnictví a doprava

Čtyřnohý robot

Jakub Jon
Liberecký kraj

Vrchlabí 22.3.2021

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 9: Strojírenství, hutnictví a doprava

Čtyřnohý robot

Quadruped robot

Autor: Jakub Jon
Škola: Střední průmyslová škola strojní a elektrotechnická a Vyšší odborná škola, Liberec 1, Masarykova 3, příspěvková organizace, 460 84 Liberec 1
Kraj: Liberecký kraj
Konzultant: Ing. Jiří Haňáček, Ing. Radek Pavlíček

Vrchlabí 22.3.2021

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

Ve Vrchlabí dne 22.3.2021 ……

Jakub Jon

Poděkování

Rád bych tímto poděkoval vedoucímu práce Ing. Jiřímu Haňáčkovi za ochotu, čas a rady při zpracovávání této odborné práce.

Dále bych chtěl poděkovat panu Ing. Radku Pavlíčkovi za pomoc, rady a připomínky k textové části.

V neposlední řadě bych chtěl poděkovat panu Ing. Miroslavu Krejčíkovi z firmy MSV SYSTEMS s.r.o. za zakoupení servomotorů pro tuto obornou práci.

Závěrem chci ještě poděkoval svým rodičům a prarodičům za podporu a rady během stavby robota.

Anotace

Práce se zabývá návrhem a konstrukcí čtyřnohého robota poháněného hobby servomotory CLS6336HV. Součástí je návrh matematického modelu robota, výpočet dopředné a inverzní kinematiky, popis použitých komponentů a matematických metod nutných pro tvorbu takého systému. V závěru je popsán i ovládací software, díky kterému je robot rozpohybován. Výsledný systém je otestován v simulaci a následně i na reálném robotu.

Klíčová slova

robot; programování; inverzní kinematika; dopředná kinematika;

Annotation

This work deals with the design and construction of a four-legged robot actuated by hobby servomotors CLS6336HV. It includes the design of a mathematical model for the robot, forward and inverse kinematics, description of used components and mathematical methods needed to build such a system. In the end of this work, the control software, thanks to which we can make the robot move, is briefly explained. The final system is tested both in a simulation and on the real robot.

Keywords

robot; programming; inverse kinematics; forward kinematics

Obsah

Úvod

K této práci jsem se dostal díky videím na YouTube od americké firmy Boston Dynamics, která se zabývá vývojem dynamických robotů a systémů. Nejvíce mě zaujal robot Spot, což je čtyřnohý robotický pes, který se dokáže autonomně pohybovat v prostoru, chodit po schodech, a dokonce i skákat. Hlavní motivací při zpracovávání této odborné práce bylo naučit se, jakým způsobem lze spočítat dopřednou a inverzní kinematiku robota a jak lze podobného robota postavit a následně i ovládat takovým způsobem, aby se natáčel a pohyboval podle povelů z PS4 ovladače.

1 Úvod do problematiky čtyřnohých robotů

Čtyřnozí roboti spadají do kategorie mobilních robotů. Mobilní robotika je odvětví robotiky zabývající se vývojem robotů, kteří se mohou volně pohybovat v prostoru. Na rozdíl od tradičních průmyslových robotů nejsou mobilní roboti připevněni k podložce. Pro pohyb nejčastěji používají kola, pásy, vrtule nebo dokonce i nohy. Každá z těchto možností má své výhody a nevýhody, které se musí při vývoji mobilního robotického systému zohlednit.

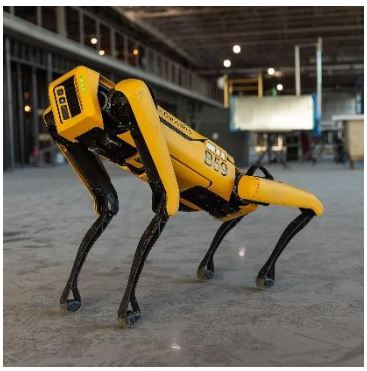
Při tvorbě chodících robotů se inspirujeme v přírodě, kde se často setkáváme s živočichy, kteří se pohybují pomocí nohou. Výhodou noh je fakt, že se s nimi lze přemisťovat i ve velmi nerovném a lidmi neupravovaném terénu. Na druhou stranu je v současné době často velmi náročné vytvořit takový algoritmus chůze, který by zaručil, že systém bude stabilní a zároveň že generovaný pohyb bude energeticky efektivní. To je jeden z problému, kterým se na světě zabývá mnoho univerzit.

1.1 Profesionální čtyřnozí roboti

Vývojem čtyřnohých chodících robotů se zabývají firmy a univerzity po celém světě (1) (2) (3). V této kapitole jsou představeni 3 nejznámější čtyřnozí chodící roboti současnosti.

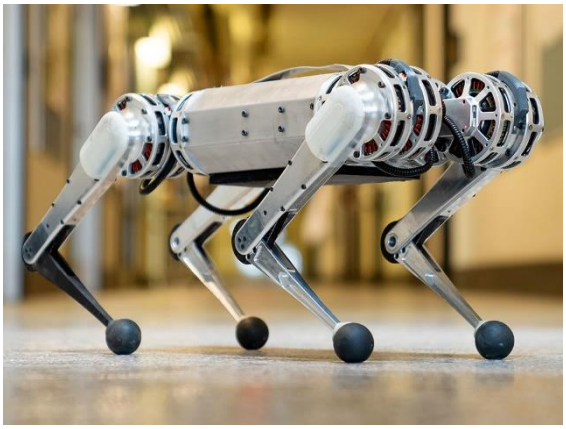
1.1.1 Spot

Spot je čtyřnohý robot vyvíjený americkou firmou Boston Dynamics. Jedná se o jednoho z nejvyspělejších čtyřnohých robotů současnosti. Do roku 2019 byl tento robot pouze výzkumným projektem, na kterém firma Boston Dynamics vyvíjela a zdokonalovala své metody a algoritmy pro ovládání čtyřnohých robotů. Od srpna minulého roku si lze Spota zakoupit a využít jeho potenciálu pro řešení nejrůznějších problémů. Kvůli své vysoké ceně si ho zatím pořizují pouze velké korporace a univerzity.



Obrázek 1 – Spot (20)

1.1.2 Mini Cheetah



Obrázek 2 – Mini Cheetah (19)

Mini Cheetah je další z mnoha čtyřnohých robotů. Byl vyvinut na americké univerzitě MIT. Cílem bylo vytvořit čtyřnohého robota vhodného pro testování nově vznikajících algoritmů pro ovládání pohybu (4). Právě cena robotů byla doposud vždy překážkou při testování nových metod, a proto v 80 % případů probíhalo testování v simulacích. Test softwaru v simulaci je perfektní možností ladit detaily a získat obrázek o tom, jak by výsledný algoritmus mohl reagovat na neočekávané situace. Oproti tomu je zcela nemožné simulaci vyladit tak, aby ve všech ohledech odpovídala realitě. Díky své relativně nízké ceně mohou vědci a studenti používat robota Mini Cheetah pro testování svých algoritmů bez nebezpečí vzniku velkých škod na hardwaru tohoto robotického psa.

1.1.3 A1

Obrázek 3 – A1 (21)

A1 je čtyřnohý robot vyvinutý čínskou firmou Unitree Robotics. Vzhledem i konstrukcí se podobá robotovi Mini Cheetah. Tento robot je velmi dynamický a levný. Cena robota A1 se pohybuje kolem 10.000 $. Díky své poměrně nízké ceně je A1 častou volbou vědců ve výzkumných centrech a na univerzitách.

1.2 Hobby čtyřnozí roboti

Nejen v profesionálním, ale i v hobby světě dochází k vývoji chodících robotů. Na rozdíl od profesionálních robotů se pro pohon nejčastěji používají hobby servomotory, které lze jednoduše zakomponovat do konstrukce těla.

Obrázek 6 – Robot Champ (22)

Obrázek 5 – Robot Pupper (23)

Obrázek 4 – Robot SpotMicroAI (24)

2 Návrh matematického modelu robota

2.1 Souřadné systémy

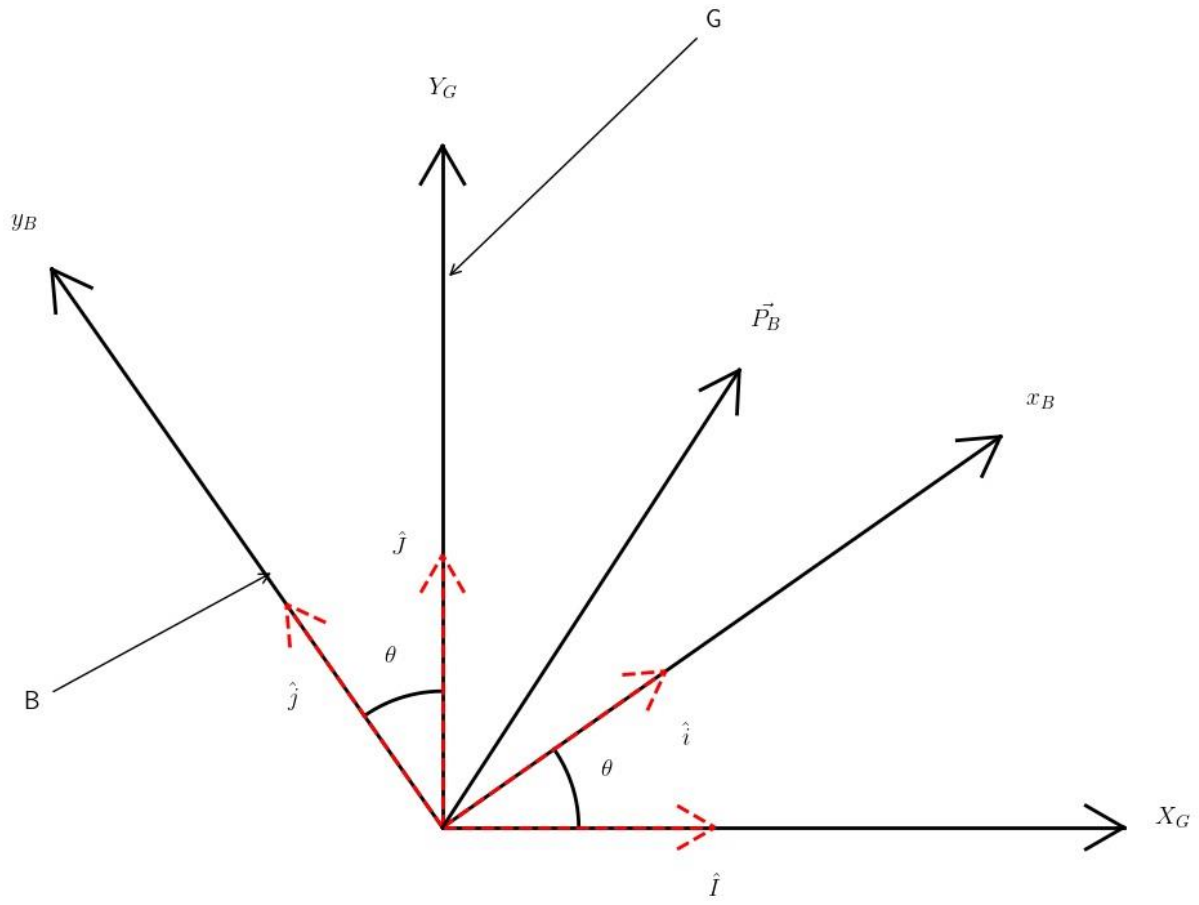
V robotice se obvykle pro popis matematického modelu používají souřadné systémy. Využívá se při tom vlastností matic, díky kterým lze vztah mezi dvěma souřadnými systémy popsat pomocí jedné 3x3 nebo 4x4 matice. V ojedinělých případech se lze setkat i s maticemi 2x2. (5)

2.1.1 Rotace v rovině

V rovině lze popsat vektor $\vec{p}$ jako součet součinů jednotkových vektorů s hodnotami x_p a y_p daného vektoru $\vec{p}$. Jednotkové vektory v rovině označujeme $\hat{\imath}$ a $\hat{\jmath}$.

$$\overrightarrow{p} = \begin{pmatrix} x_p \\ y_p \end{pmatrix} = x_p * \begin{pmatrix} 1 \\ 0 \end{pmatrix} + y_p * \begin{pmatrix} 0 \\ 1 \end{pmatrix} = x_p * \hat{\imath} + y_p * \hat{\jmath} \tag{2.1}$$

Pokud dojde k rotaci souřadného systému B, dojde k vychýlení jednotkových vektorů tohoto souřadného systému ($\hat{\imath}$ a $\hat{\jmath}$) od jednotkových vektorů souřadného systému G ($\hat{I}$ a $\hat{J}$) o úhel Θ.



Obrázek 7 – Rotace souřadného systému v rovině

Jednotkové vektory mají ve vlastním souřadném systému souřadnice $\binom{1}{0}$ a $\binom{0}{1}$. Chceme-li zjistit, jak vypadají jednotkové vektory souřadného systému B v souřadném systému G, lze k tomu využít goniometrických funkcí a fakt, že délka každého jednotkového vektoru je vždy 1.

$$\hat{\imath}_G = \begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix} \tag{2.2}$$

$$\hat{\jmath}_G = \begin{pmatrix}-\sin\theta \\ \cos\theta\end{pmatrix} \tag{2.3}$$

Rovnice (2.2) a (2.3) vyjadřují jednotkové vektory souřadného systému B v G. Chceme-li tedy vyjádřit souřadnice vektoru $\overrightarrow{P_B} = \begin{pmatrix}x_B \\ y_B\end{pmatrix}$ v souřadném vektoru G, lze k tomu využít jednotkové vektory souřadného systému B vyjádřené v systému G.

$$\overrightarrow{P_G} = x_B * \hat{\imath}_G + y_B * \hat{\jmath}_G \tag{2.4}$$

$$\overrightarrow{P_G} = x_b * \begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix} + y_b * \begin{pmatrix}-\sin\theta \\ \cos\theta\end{pmatrix} \tag{2.5}$$

Rovnici (2.5) lze zapsat jako součin matice a vektoru.

$$\overrightarrow{P_G} = \begin{pmatrix}\cos\theta & -\sin\theta \\ \sin\theta & \cos\theta\end{pmatrix} * \begin{pmatrix}x_B \\ y_B\end{pmatrix} \tag{2.6}$$

Matici v rovnici (2.6) často označujeme jako rotační matici, která mapuje souřadnice z B do G. Značíme ji ${}^{G}R_B$, kde levý horní index označuje, do kterého souřadného systému mapujeme a pravý dolní index označuje, ze kterého souřadného systému souřadnice převádíme.

$$\overrightarrow{P_G} = {}^{G}R_B * \overrightarrow{P_B} \tag{2.7}$$

Rovnici (2.7) lze upravit i tak, abychom převáděli souřadnice z B do G. Obě strany rovnice lze vynásobit maticí ${}^{G}R_B^{-1}$, tedy inverzní maticí původní rotační matice.

$$\overrightarrow{P_B} = {}^{G}R_B^{-1} * \overrightarrow{P_G} \tag{2.8}$$

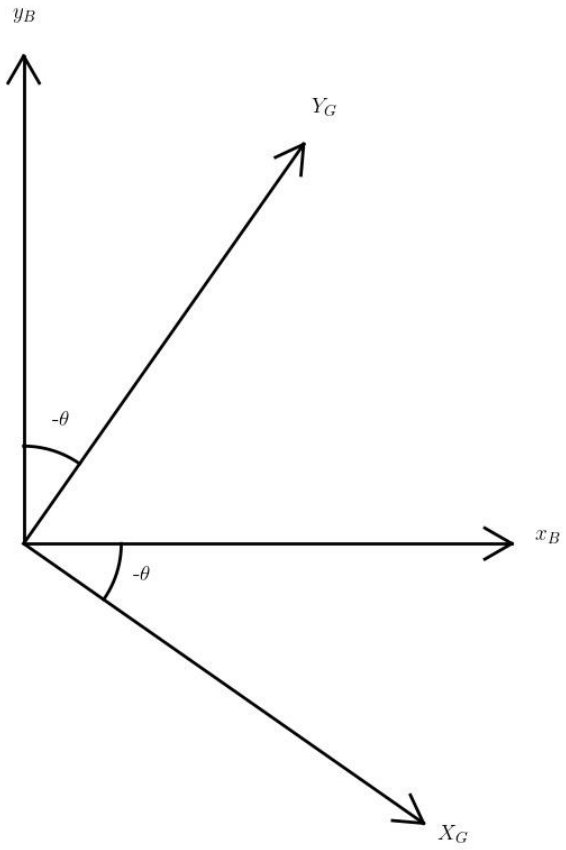
V této rovnici dochází k převodu souřadnic z G do B. K takovému převodu lze tedy použít i matici ${}^{B}R_G$.

$$\overrightarrow{P_B} = {}^{B}R_G * \overrightarrow{P_G} \tag{2.9}$$

Z rovnic (2.8) a (2.9) lze snadnou usoudit, že obě matice ${}^{G}R_B^{-1}$ a ${}^{B}R_G$ si jsou rovny.

$${}^{G}R_B^{-1} = {}^{B}R_G \tag{2.10}$$

V případě, že převádíme souřadnice z B do G, lze si představit, že při rotaci souřadného systému B o úhel Θ zůstal souřadný systém B stát a systém G rotoval relativně k B o úhel -Θ.



Obrázek 8 – Relativní rotace souřadného systému G vzhledem k B

Podle rovnice (2.6) lze sestavit novou rotační matici ${}^{B}R_{G}$.

$$ {}^{B}R_{G} = \begin{pmatrix} \cos -\theta & -\sin -\theta \\ \sin -\theta & \cos -\theta \end{pmatrix} \tag{2.11} $$

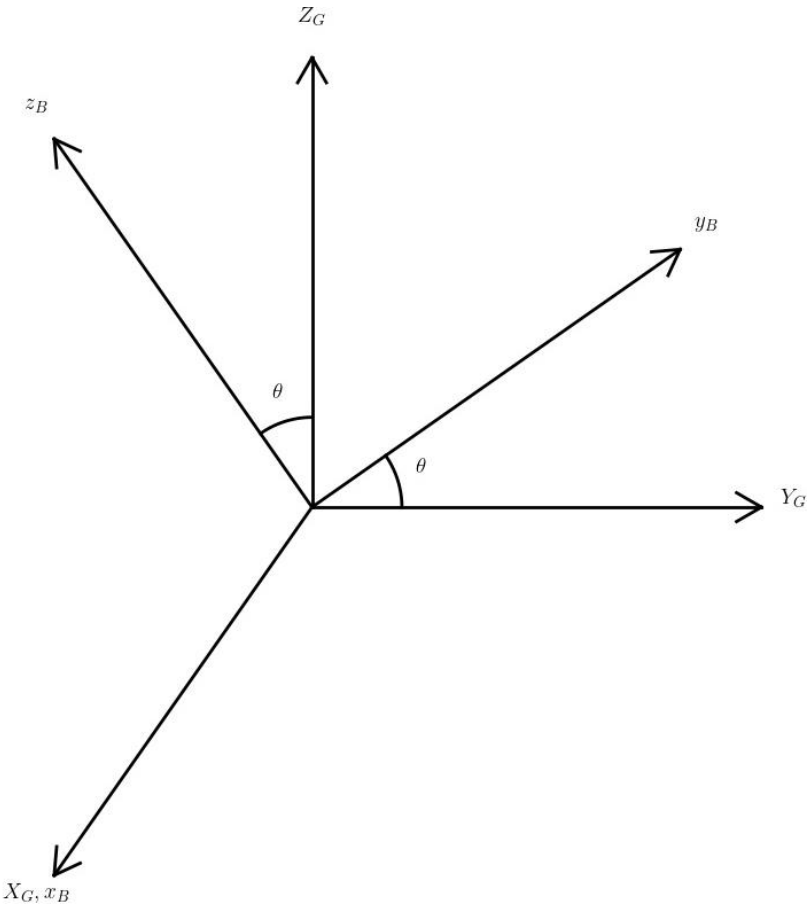
$$ {}^{B}R_{G} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{2.12} $$

Rovnici (2.11) dále upravíme tak, abychom získali rovnici (2.12). Lze si povšimnout, že oproti rotační matici ${}^{G}R_{B}$ z rovnice (2.6) došlo pouze k záměně řádků a sloupců matice, tzn. došlo k transpozici matice. Z tohoto vyplývá velmi důležitá vlastnost rotačních matic, a to taková, že inverzní rotační matice se rovná rotační matici transponované.

$$ {}^{B}R_{G} = {}^{G}R_{B}^{T} \tag{2.13} $$

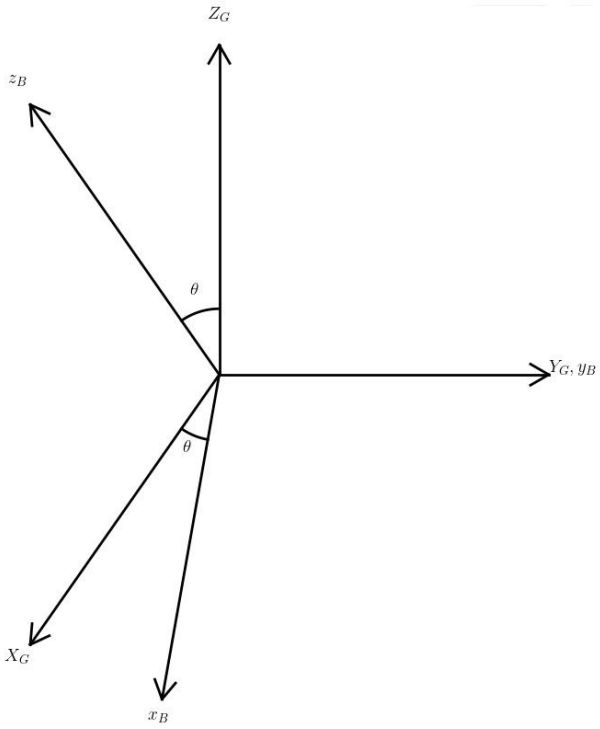
2.1.2 Rotace v prostoru

Rotace v rovině je speciálním případem rotace v prostoru, kdy k rotaci dochází pouze kolem jedné osy. I když je rotace v rovině často dostačující pro popis robotů pohybujících se ve 2D prostoru, pro tvorbu robotů operujících ve 3D prostoru jsou rotace v rovině nedostačující. Je tedy nutné doplnit 3D rotační matice popisující rotace kolem os X, Y a Z námi zvoleného souřadného systému.



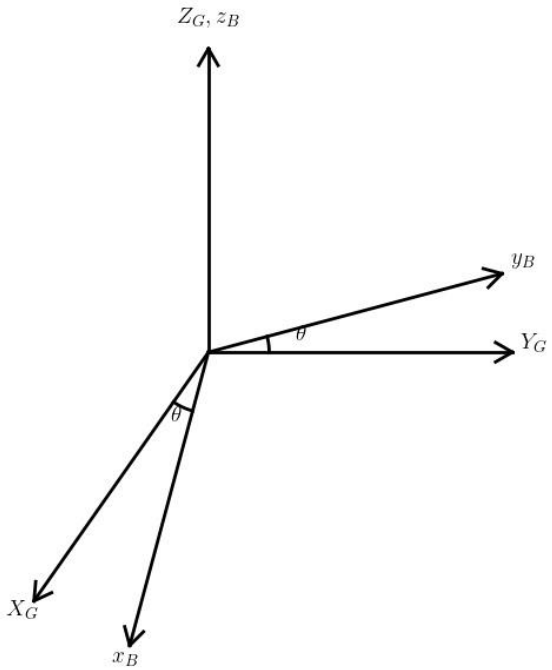
Obrázek 9 – Rotace kolem osy X v prostoru

$$R_{X,\Theta} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \tag{2.14}$$



Obrázek 10 – Rotace kolem osy Y v prostoru

$$R_{Y,\Theta} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{2.15}$$



Obrázek 11 – Rotace kolem osy Z v prostoru

$$R_{Z,\Theta} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.16}$$

Rotační matice v prostoru lze stejně jako rotační matice v rovině použít pro převod souřadnic v rotovaném souřadném systému do souřadného systému stacionárního. I zde lze použít rovnici (2.7) a (2.9). Rozdílem 3D rotačních matic oproti rotačním maticím v rovině je počet řádku a počet sloupců této matice. Pro rovnice (2.7) a (2.9) platí, že $\overrightarrow{P_G} \in \mathbb{R}^2$, $\overrightarrow{P_B} \in \mathbb{R}^2$, ${}^{G}R_B \in \mathbb{R}^{2x2}$, pokud se jedná o rotaci v rovině a $\overrightarrow{P_G} \in \mathbb{R}^3$, $\overrightarrow{P_B} \in \mathbb{R}^3$, ${}^{G}R_B \in \mathbb{R}^{3x3}$, uvažujeme-li rotaci v prostoru. Došlo tedy k přidání souřadnice Z.

V reálném světě nedochází k rotaci pouze kolem jedné osy. Rotace se často kombinují a vznikají tak nejrůznější orientace. Matici popisující rotaci několika po sobě navazujících rotací kolem os X, Y nebo Z lze popsat jako jejich součin. Jelikož po sobě může navazovat mnoho rotací, nelze jako indexy používat písmena. Místo nich se často používají čísla. Princip je ale pořád stejný, rotační matice ${}^{n}R_k$ mapuje souřadnice ze souřadného systému k do souřadného systému n ($n, k \in \mathbb{Z}$).

$$ {}^{1}R_4 = {}^{1}R_2 * {}^{2}R_3 * {}^{3}R_4 \tag{2.17} $$

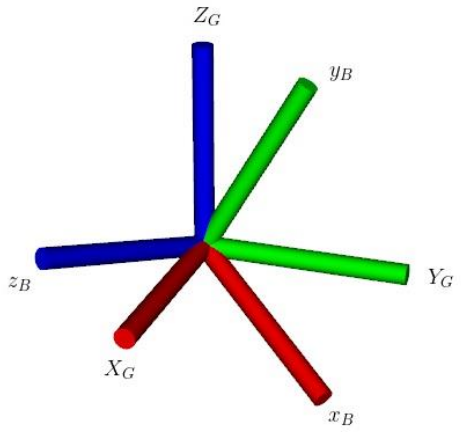
Matice v rovnici (2.17) popisuje výslednou orientaci souřadného systému po třech po sobě navazujících rotacích. Nejdříve nastala rotace popsaná maticí ${}^{1}R_2$, následovala rotace ${}^{2}R_3$, po ní se uskutečnila rotace ${}^{3}R_4$. Za dílčí rotace si lze dosadit libovolné rotační matice. Můžeme kupříkladu uvažovat rotaci kolem osy X následovanou rotací kolem nově vzniklé osy Y a poté rotací kolem osy Z. Výsledná rotační matice je v rovnici (2.18).

$$ {}^{1}R_4 = R_{X,\alpha} * R_{Y,\beta} * R_{Z,\gamma} \tag{2.18} $$

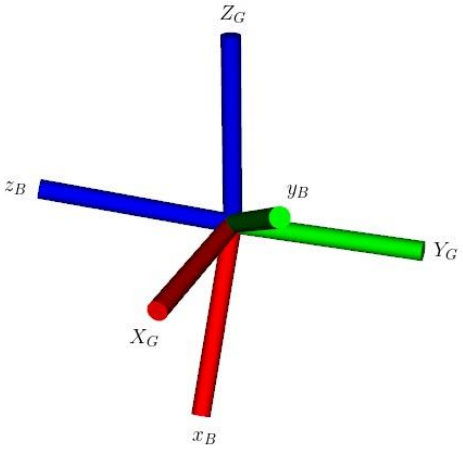
Je důležité si uvědomit, že rotace n po rotaci n – 1 probíhá kolem os souřadného systému, který se při poslední rotaci otáčel. Zachování pořadí v součinu rotačních matic je tedy nezbytné pro získání správné výsledné rotační matice popisující orientaci výsledného pootočeného souřadného systému.

$$ R_{X,\alpha} * R_{Y,\beta} * R_{Z,\gamma} \neq R_{Y,\beta} * R_{Z,\gamma} * R_{X,\alpha} \tag{2.19} $$

$$ {}^{1}R_2 * {}^{2}R_3 * {}^{3}R_4 \neq {}^{2}R_3 * {}^{3}R_4 * {}^{1}R_2 \tag{2.20} $$



Obrázek 13 – Rotace okolo osy X o 60 stupňů následovaná rotací kolem nově vzniklé osy Y o 60 stupňů



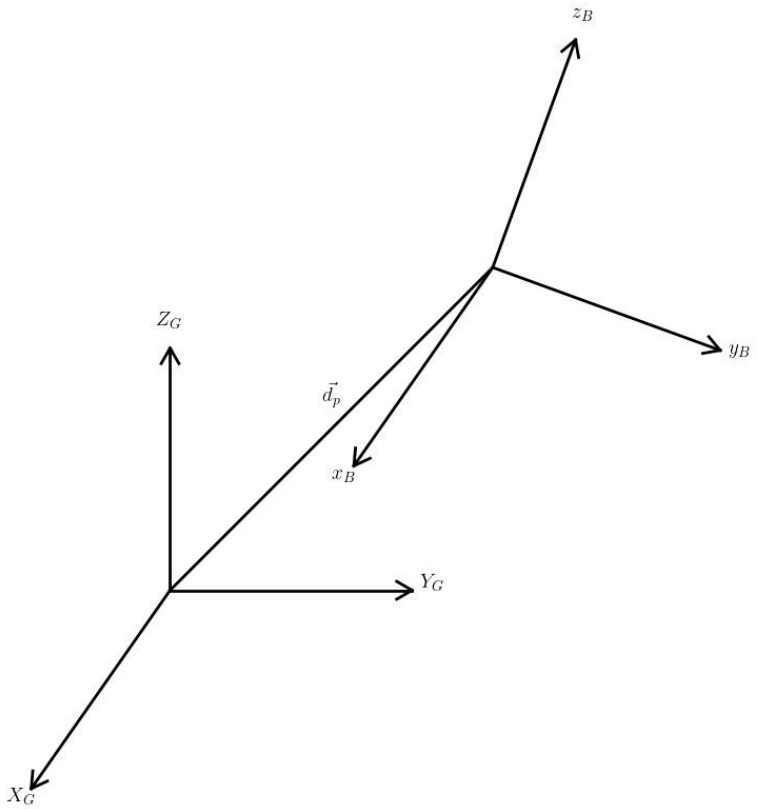
Obrázek 12 – Rotace okolo osy Y o 60 stupňů následovaná rotací kolem nově vzniklé osy X o 60 stupňů

2.1.3 Transformace v prostoru

Při popisu robotů se jen zřídka setkáváme se dvěma souřadnými systémy se společným počátkem. Většinou jsou dva souřadné systémy oproti sobě posunuty o určité vzdálenosti podél os X, Y a Z jednoho z těchto souřadných systémů. Toto posunutí současně s rotací lze popsat pomocí tzv. transformačních matic T. V prostoru má transformační matice následující tvar:

$$ {}^{G}T_{B} = \begin{pmatrix} R^{3x3} & d^{3x1} \\ 0^{1x3} & 1 \end{pmatrix} \tag{2.21} $$

Jedná se o transformační matici ${}^{G}T_{B} \in \mathbb{R}^{4x4}$, která popisuje posunutí a zároveň rotaci souřadného systému B v G. Rotační matice R je nezávislá na posunutí d a naopak, lze si tedy představit, že nejdříve nastane posunutí o hodnoty x_d, y_d, z_d podél os X_G, Y_G, Z_G a následně souřadný systém rotuje tak, aby vznikla rotační matice R.



Obrázek 14 – Transformace souřadného systému v prostoru

Rotace okolo os X, Y, Z a posunutí souřadného systému lze popsat pomocí těchto matic:

$$T_{Rx,\theta} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.22}$$

$$T_{Ry,\theta} = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.23}$$

$$T_{Rz,\theta} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.24}$$

$$T_{d_p} = \begin{pmatrix} 1 & 0 & 0 & x_d \\ 0 & 1 & 0 & y_d \\ 0 & 0 & 1 & z_d \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.25}$$

I pro transformační matice platí, že pokud chceme vytvořit jakoukoli kombinaci dílčích transformací, lze je mezi sebou násobit. Stejně jako pro rotační matice ale platí, že násobení dílčích transformačních matic není komutativní a je tedy nutné zachovávat

správné pořadí (je rozdíl, jestli nejdříve rotujeme okolo osy X o 90 stupňů a následně se posuneme podél osy Z nebo jestli tyto dvě operace provedeme naopak. Výsledkem jsou dvě různé transformační matice). Rovnici (2.21) lze zapsat jakou součin transformační matice popisující posunutí a transformační matice popisující rotaci systému.

$$ {}^{G}T_{B} = \begin{pmatrix} I^{3x3} & d^{3x1} \\ 0^{1x3} & 1 \end{pmatrix} * \begin{pmatrix} R^{3x3} & 0^{3x1} \\ 0^{1x3} & 1 \end{pmatrix} \tag{2.26} $$

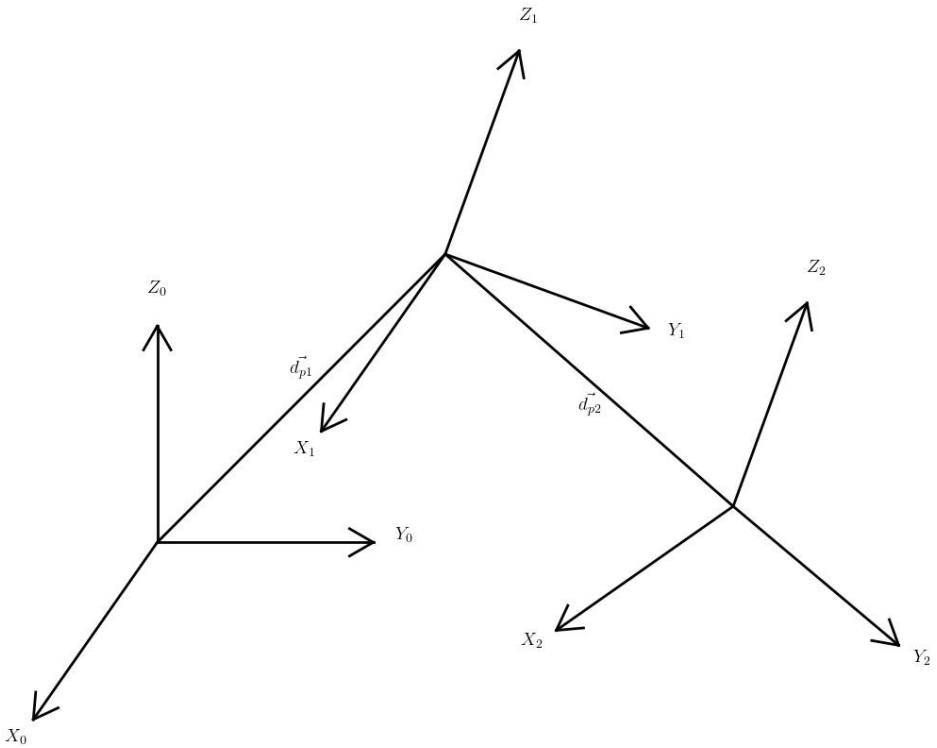
Robotické systémy se mohou skládat až z několika set souřadných systémů. Pro převod souřadnic ze souřadného systému n do souřadného systému k je nutné vypočítat matici ${}^{k}T_{n}$, která nám takový převod umožnuje.

$$ {}^{k}T_{n} = {}^{k}T_{k+1} * {}^{k+1}T_{k+2} * {}^{k+2}T_{k+3} * \dots * {}^{n-2}T_{n-1} * {}^{n-1}T_{n} \tag{2.27} $$

$$ \overrightarrow{P_k} = {}^{k}T_{n} * \overrightarrow{P_n} \tag{2.28} $$

$$ \overrightarrow{P_k} = \begin{pmatrix} x_k \\ y_k \\ z_k \\ 1 \end{pmatrix} \tag{2.29} $$

$$ \overrightarrow{P_n} = \begin{pmatrix} x_n \\ y_n \\ z_n \\ 1 \end{pmatrix} \tag{2.30} $$



Obrázek 15 – Složená transformace

2.1.4 Inverzní transformační matice

Inverzní transformační matice je důležitou součástí při výpočtech kinematiky robotů. Máme-li vypočítanou transformační matici, pro kterou chceme najít její inverzní matici, mohli bychom použít různých programů, které jsou k tomu uzpůsobeny. Tyto programy jsou ale optimalizovány k výpočtu inverzních matic pro obecné n x n matice a výpočet tedy velmi často trvá poměrně dlouho. Pro zkrácení doby výpočtu inverzních transformačních matic počítačem lze užít rovnice (2.31).

$$ {}^{k}T_{n}^{-1} = \begin{pmatrix} {}^{k}R_{n}^{T} & -{}^{k}R_{n}^{T} * d \\ 0^{1x3} & 1 \end{pmatrix} \tag{2.31} $$

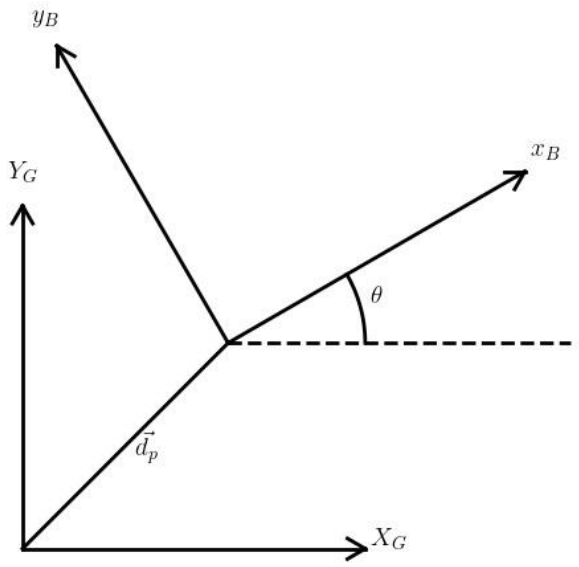
Inverzní transformační matice nám umožňuje převádět souřadnice z k do n, platí tedy, že ${}^{k}T_{n}^{-1}$ je rovna ${}^{n}T_{k}$.

$$ {}^{n}T_{k} = {}^{k}T_{n}^{-1} \tag{2.32} $$

Rovnici (2.31) lze zapsat jako součin dvou dílčích transformačních matic.

$$ {}^{k}T_{n}^{-1} = \begin{pmatrix} {}^{k}R_{n}^{T} & 0^{3x1} \\ 0^{1x3} & 1 \end{pmatrix} * \begin{pmatrix} I^{3x3} & -d \\ 0^{1x3} & 1 \end{pmatrix} \tag{2.33} $$

Podle rovnice (2.33) dochází nejdříve k otočení souřadného systému n tak, aby orientace k i n byla stejná a následuje posunutí zpět o vektor d, které přesune počátek souřadného systému n do počátku souřadného systému k.



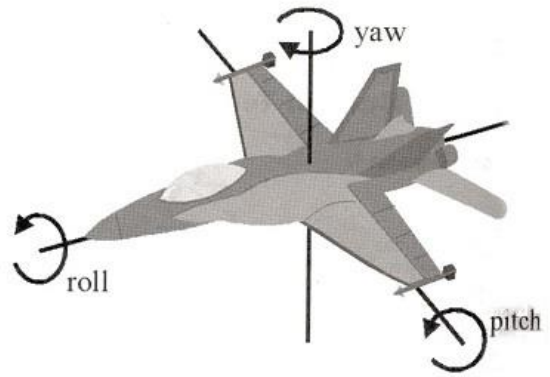
Obrázek 16 – Princip inverzní transformační matice.

Na obrázku 16 je možno vidět dva souřadné systémy, které nemají společný počátek souřadného systému ani stejnou orientaci. Pro získání transformační matice pro převod souřadnic z B do G je potřeba se nejdříve posunout o vektor $\overrightarrow{d_p}$ a následně rotovat o úhel θ. Chceme-li naopak získat inverzní transformační matici pro převod souřadnic z G do B, je třeba nejdříve rotovat o úhel $-\theta$ a následně se posunout o vektor $-\overrightarrow{d_p}$.

Rovnice (2.31) se v robotice používá pro svou jednoduchost a rychlost. Pro ovládání robotů se mnohdy používají programy, které inverzní transformační matici musí počítat i několikrát za sekundu. Je tedy nezbytné tento výpočet zefektivnit a optimalizovat. Rychlostně může být výpočet pomocí rovnice (2.31) až 10x efektivnější než výpočet pomocí komerčních programů pro generování inverzních matic.

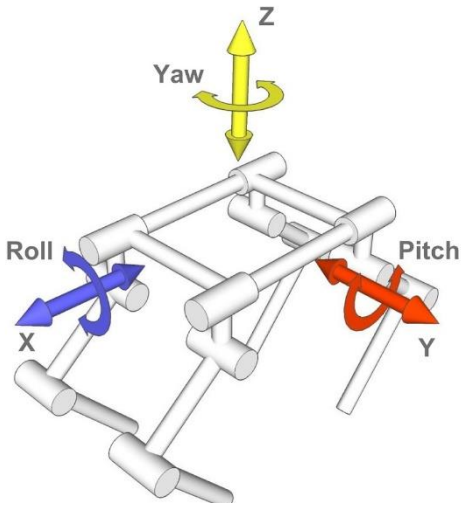
2.2 Eulerovy úhly

Eulerovy úhly jsou tři úhly, díky kterým lze jakoukoli rotační matici ${}^{k}R_{n}$, a tedy orientaci, zapsat jako součin tří rotací kolem tří různých os. Tyto úhly se nejčastěji nazývají roll, pitch a yaw.



Obrázek 17 – Eulerovy úhly na modelu letadla (25)

Eulerovy úhly lze použít nejen v letectví, ale také v robotice pro popis natočení těla čtyřnohého robota.

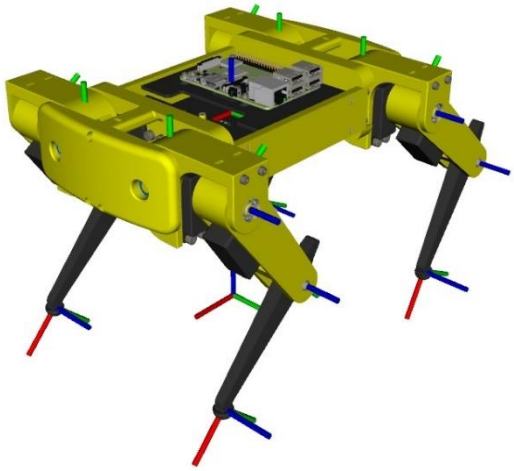


Obrázek 18 – Eulerovy úhly na modelu robota (26)

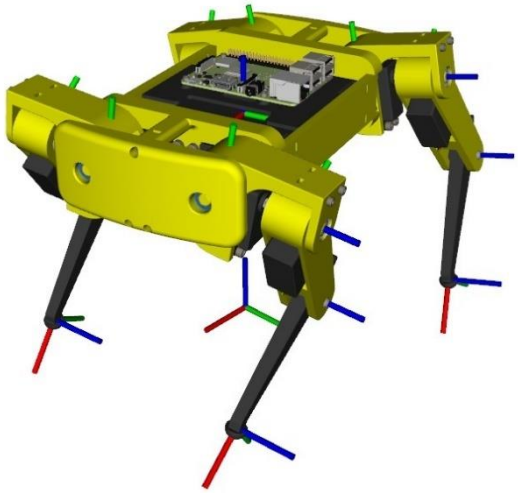
V našem případě budeme používat Eulerovy úhly tak, že vždy nejdříve nastane rotace roll kolem osy X, následovat bude rotace pitch kolem nově vzniklé osy Y, a nakonec proběhne rotace yaw kolem osy Z. Výsledkem bude rotační matice R, která bude popisovat lokální orientaci těla robota v určitém souřadném systému a bude nám sloužit pro výpočet inverzní kinematiky celého robota (viz. kapitola 2.6).

$$R = R_{X,\alpha_{roll}} * R_{Y,\beta_{pitch}} * R_{Z,\gamma_{yaw}} \tag{2.34}$$

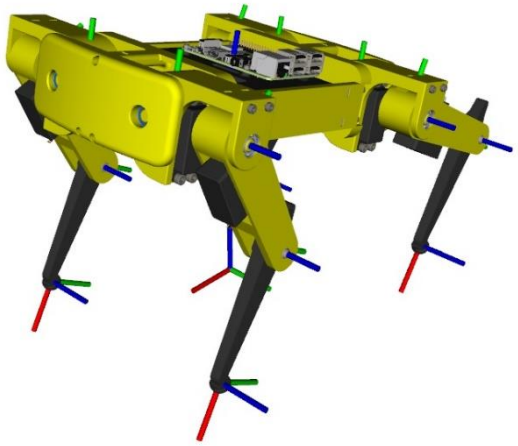
$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_{roll} & -\sin\alpha_{roll} \\ 0 & \sin\alpha_{roll} & \cos\alpha_{roll} \end{pmatrix} * \begin{pmatrix} \cos\beta_{pitch} & 0 & \sin\beta_{pitch} \\ 0 & 1 & 0 \\ -\sin\beta_{pitch} & 0 & \cos\beta_{pitch} \end{pmatrix} * \begin{pmatrix} \cos\gamma_{yaw} & -\sin\gamma_{yaw} & 0 \\ \sin\gamma_{yaw} & \cos\gamma_{yaw} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.35}$$



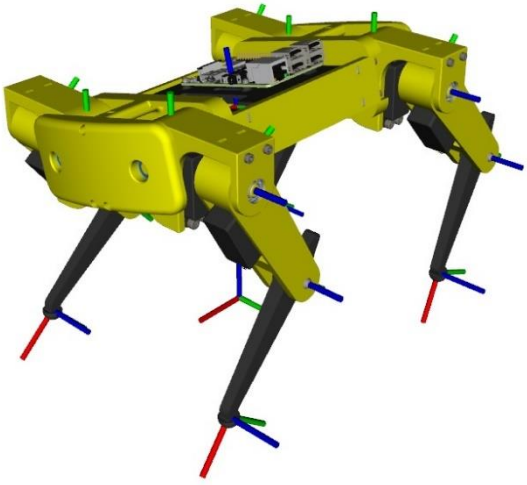
Obrázek 19 – Lokální orientace těla robota pro $\alpha_{roll} = 0°$, $\beta_{pitch} = 0°$, $\gamma_{yaw} = 0°$



Obrázek 20 – Lokální orientace těla robota pro $\alpha_{roll} = 0°$, $\beta_{pitch} = 0°$, $\gamma_{yaw} = 20°$



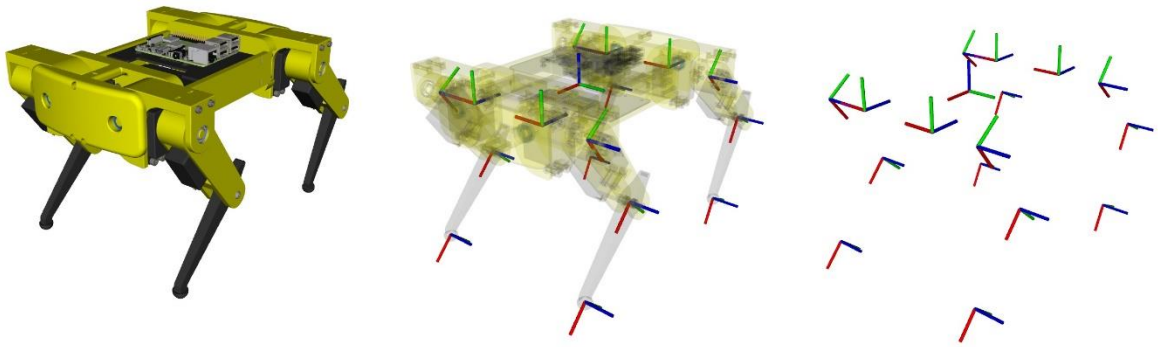
Obrázek 21 – Lokální orientace těla robota pro $\alpha_{roll} = 0°$, $\beta_{pitch} = -20°$, $\gamma_{yaw} = 0°$



Obrázek 22 – Lokální orientace těla robota pro $\alpha_{roll} = 20°$, $\beta_{pitch} = 0°$, $\gamma_{yaw} = 0°$

2.3 Denevit-Hartenbergova notace

Robotické systémy se popisují pomocí souřadných systémů, které se umisťují do míst, u kterých jednoznačně potřebujeme vědět jejich pozice, popřípadě i jejich orientace. Souřadné systémy tedy nejčastěji umisťujeme na osy rotací pohyblivých členů robota. Dále se souřadné systémy dávají na konce kinematických řetězců. V našem případě se tedy 4 souřadné systémy budou nacházet na koncových bodech jednotlivých nohou.



Obrázek 23 – Souřadné systémy robota

Pro popis vztahu mezi dvěma souřadnými systémy potřebujeme 6 parametrů. Jedná se o posunutí ve všech osách a rotace kolem každé z těchto os. Pro jednoznačné určení pozic a orientací n souřadných systémů tedy potřebujeme 6 * (n – 1) parametrů. Robotické systémy mohou obsahovat až několik set souřadných systémů a parametrů potřebných pro popis matematického modelu takového robota by bylo hodně.

Jedním způsobem, jak snížit počet potřebných parametrů pro určení vztahů mezi jednotlivými souřadnými systémy, je použití tzv. Denevit-Hartenbergovi notace, díky které nám stačí pouze 4 parametry (posunutí v ose X, rotace kolem osy X, posunutí v ose Z, rotace kolem osy Z), chceme-li jednoznačně určit pozici a orientaci souřadného systému v prostoru. Díky tomu se Denevit-Hartenbergova notace používá ve většině robotických systému.

Denevit-Hartenbergova notace používá tzv. DH-parametry, které popisují právě již zmíněné 4 potřebné parametry k určení pozice a orientace souřadného systému v trojrozměrném prostoru. Při určování DH-parametrů je potřeba řídit se sadou několika pravidel, které nám DH-notaci umožňují používat.

1. Rotace členu $i + 1$ probíhá kolem osy z_i
2. Osa x_i je určena nejkratší vzdáleností mezi z_{i-1} a z_i
3. Osa y_i se určuje podle pravidla pravé ruky

Toto jsou 3 z několika pravidel, které je nutné dodržet pro správné umístění souřadných systémů na robota. Zmíněná 3 pravidla jsou pro tuto odbornou práci dostačující.

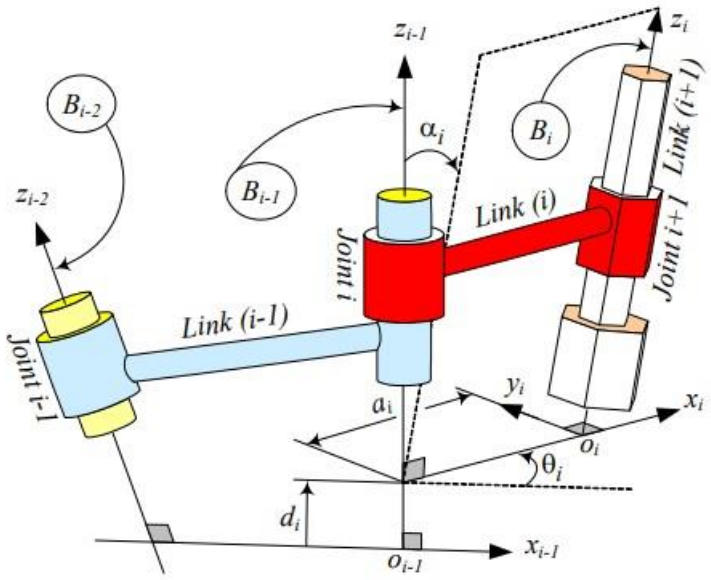
Máme-li umístěné souřadné systémy na robotickém systému, můžeme určit DH-parametry.

a_i ..... vzdálenost mezi z_{i-1} a z_i podél osy x_i

α_i ..... úhel rotace osy z_{i-1} kolem osy x_i potřebné k tomu, aby se osa z_{i-1} stala rovnoběžnou s osou z_i

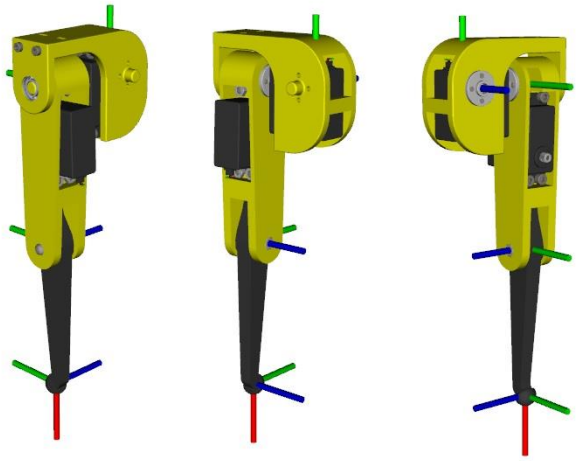
d_i ..... vzdálenost mezi x_{i-1} a x_i podél osy z_i

θ_i ..... úhel rotace osy x_{i-1} kolem osy z_{i-1} potřebné k tomu, aby se osa x_{i-1} stala rovnoběžnou s osou x_i



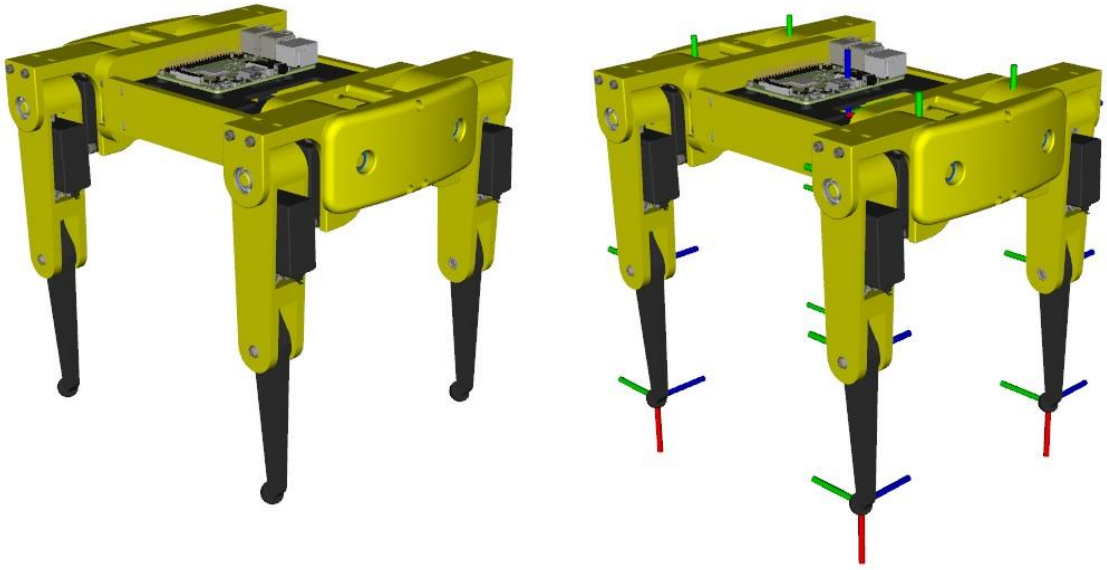
Obrázek 24 – DH-parametry (5)

Při určování DH-parametrů je nutné si zvolit referenční (neutrální) pozici, ve které se všechny kontrolovatelné parametry (rotace kolem osy Z, tzn. θ_i) rovnají nule.



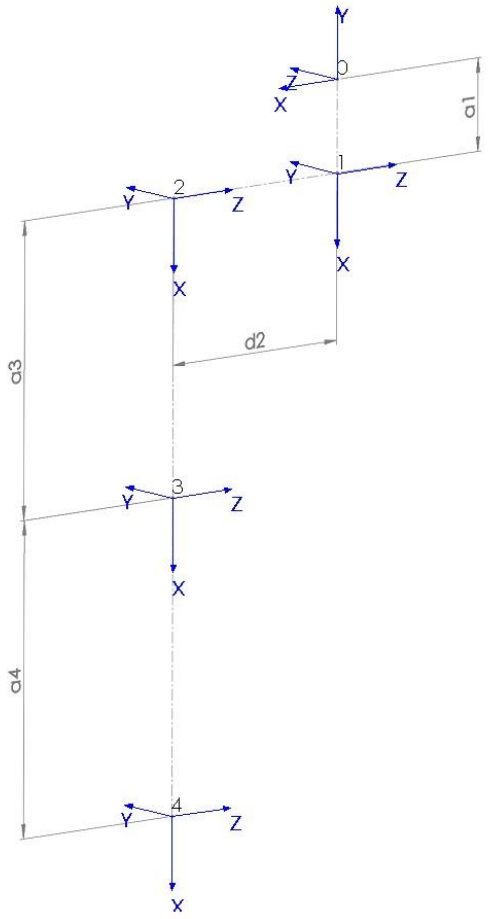
Obrázek 25 – Referenční pozice přední pravé nohy robota

Referenční pozice pro celého robota je vidět na obrázku 26.



Obrázek 26 – Referenční pozice celého robota

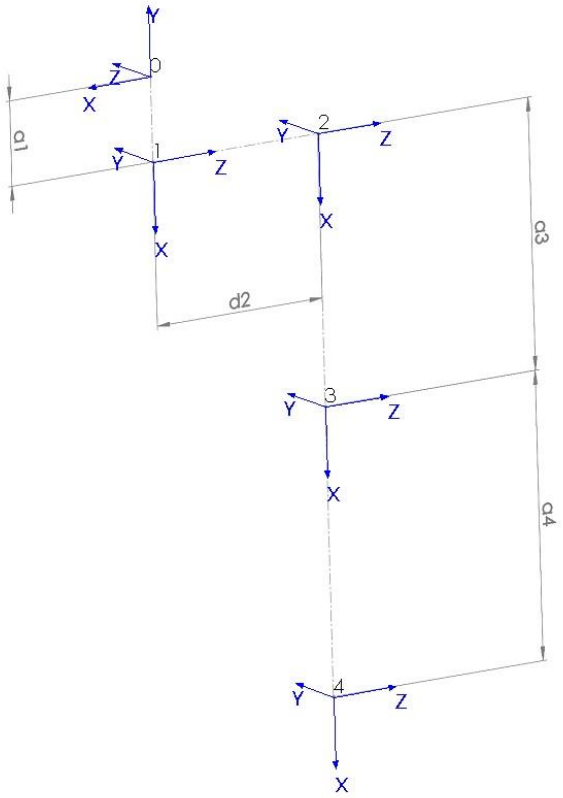
Máme-li definovanou referenční pozici, je možné zjistit DH-parametry. Na obrázku 26 je vidět, že se robotický pes skládá ze 4 poměrně stejných nohou. Ve skutečnosti jsou nohy na pravé straně z pohledu DH-parametrů stejné. To samé platí pro nohy na straně levé. Jediným rozdílem mezi nohami na obou stranách je znaménko parametru d_2 (viz. obrázek 27 a 28 a tabulky 1 a 2).



Obrázek 27 – Souřadné systémy nohou robota na pravé straně

Číslo souřadného systému i	$\boldsymbol{a_i}$ $[\boldsymbol{m}]$	$\boldsymbol{\alpha_i}$ [°]	$\boldsymbol{d_i}$ $[\boldsymbol{m}]$	$\boldsymbol{\theta_i}$ [°]
1	a_1	90	0	$\theta_1 - 90$
2	0	0	$-d_2$	0
3	a_3	0	0	θ_3
4	a_4	0	0	θ_4

Tabulka 1 – DH-parametry nohou robota na pravé straně



Obrázek 28 – Souřadné systémy nohou robota na levé straně

Číslo souřadného systému i	$\boldsymbol{a_i}$ $[\boldsymbol{m}]$	$\boldsymbol{\alpha_i}$ [°]	$\boldsymbol{d_i}$ $[\boldsymbol{m}]$	$\boldsymbol{\theta_i}$ [°]
1	a_1	90	0	$\theta_1 - 90$
2	0	0	$+d_2$	0
3	a_3	0	0	θ_3
4	a_4	0	0	θ_4

Tabulka 2 – DH-parametry nohou robota na levé straně

2.4 Dopředná kinematika

Dopředná kinematika robota je matematická funkce, která provádí zobrazení z prostoru kloubových souřadnic (to jsou tedy úhly θ_i) do prostoru souřadnic v kartézském souřadném systému (souřadnice x, y, z). Známe-li tedy úhly natočení jednotlivých kloubů, lze pomocí dopředné kinematiky zjistit, kde v souřadném systému 0 se ostatní souřadné systémy nacházejí.

I když můžeme zjistit polohu všech souřadných systémů, nejčastěji se setkáváme s případem, kdy počítáme pouze polohu a orientaci koncového bodu kinematického řetězce.

Pro získání pozice a orientace souřadného systému 4 v souřadném systému 0 potřebujeme vypočítat transformační matici ${}^0T_4$. Tuto matici lze jednoduše najít pomocí součinu dílčích transformací.

$$ {}^0T_4 = {}^0T_1 * {}^1T_2 * {}^2T_3 * {}^3T_4 \tag{2.36}$$

Po výpočtu matice ${}^0T_4$ by následoval převod souřadnic počátku souřadného systému 4 vyjádřených v systémů 4 do systému 0. K tomu lze využít rovnici (2.28). Jelikož bychom ale získaly pouze informace již obsažené v matici ${}^0T_4$, není tento převod nutný.

Obecně lze matici ${}^{i-1}T_i$ získat pomocí součinu transformací, které obsahují námi zjištěné DH-parametry.

$$ {}^{i-1}T_i = Trans_{z_i,d_i} * Rot_{z_i,\theta_i} * Trans_{x_i,a_i} * Rot_{x_i,\alpha_i} \tag{2.37}$$

$$ Trans_{z_i,d_i} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.38}$$

$$ Rot_{z_i,\theta_i} = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.39}$$

$$ Trans_{x_i,a_i} = \begin{pmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.40}$$

$$ Rot_{x_i,\alpha_i} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.41}$$

Vynásobením všech dílčích transformací získáváme následující obecnou matici.

$$ {}^{i-1}T_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i * \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i * \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.42} $$

2.4.1 Dopředná kinematika nohou na pravé straně

Dosazením parametrů z tabulky 1 do rovnice (2.42) získáme konkrétní transformační matice.

$$ {}^{0}T_1 = \begin{pmatrix} \sin\theta_1 & 0 & -\cos\theta_1 & a_1 * \sin\theta_1 \\ -\cos\theta_1 & 0 & -\sin\theta_1 & -a_1 * \cos\theta_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.43} $$

$$ {}^{1}T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.44} $$

$$ {}^{2}T_3 = \begin{pmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_3 * \cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & a_3 * \cos\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.45} $$

$$ {}^{3}T_4 = \begin{pmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & a_4 * \cos\theta_4 \\ \sin\theta_4 & \cos\theta_4 & 0 & a_4 * \cos\theta_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.46} $$

Po dosazení těchto dílčích transformačních matic do rovnice (2.36) získáme výslednou transformační matici ${}^{0}T_4$.

$$ {}^{0}T_4 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.47} $$

$$ d_x = a_1 * \sin\theta_1 + d_2 * \cos\theta_1 + a_3 * \sin\theta_1 * \cos\theta_3 + a_4 * \sin\theta_1 * \cos(\theta_3 + \theta_4) \tag{2.48} $$

$$ d_y = -a_1 * \cos\theta_1 + d_2 * \sin\theta_1 - a_3 * \cos\theta_1 * \cos\theta_3 - a_4 * \cos\theta_1 * \cos(\theta_3 + \theta_4) \tag{2.49} $$

$$ d_z = a_3 * \sin\theta_3 + a_4 * \sin(\theta_3 + \theta_4) \tag{2.50} $$

$$ r_{11} = \sin\theta_1 * \cos(\theta_3 + \theta_4) \tag{2.51} $$

$$ r_{21} = -\cos\theta_1 * \cos(\theta_3 + \theta_4) \tag{2.52} $$

$$ r_{31} = \sin(\theta_3 + \theta_4) \tag{2.53} $$

$$r_{12} = = \ -\sin\theta_1 * \sin(\theta_3 + \ \theta_4) \tag{2.54}$$

$$r_{22} = \cos\theta_1 * \ \sin(\theta_3 + \ \theta_4) \tag{2.55}$$

$$r_{32} = \ \cos(\theta_3 + \ \theta_4) \tag{2.56}$$

$$r_{13} = \ -\cos\theta_1 \tag{2.57}$$

$$r_{23} = \ -\sin\theta_1 \tag{2.58}$$

$$r_{33} = 0 \tag{2.59}$$

2.4.2 Dopředná kinematika nohou na levé straně

Stejně jako v případě nohou na pravé straně lze spočítat rovnice dopředné kinematiky pro nohy na straně levé. Jediným rozdílem ve výpočtu je znaménko před parametrem d_2, které je v případě levých nohou kladné.

Rovnice (2.43), (2.45) a (2.46) jsou v tomto případě stejné, matice v rovnici (2.44) se nepatrně změní.

$${}^1T_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & +d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.60}$$

Vynásobením jednotlivých transformačních matic získáme matici ${}^0T_4$, která se od matice v rovnici (2.47) liší pouze v hodnotách d_x a d_y. Ostatní hodnoty (tzn. d_z a orientace) zůstávají identické.

$$d_x = a_1 * \sin\theta_1 - d_2 * \ \cos\theta_1 + a_3 * \ \sin\theta_1 * \cos\theta_3 + a_4 * \sin\theta_1 * \cos(\theta_3 + \theta_4) \tag{2.61}$$

$$d_y = -a_1 * \cos\theta_1 - d_2 * \sin\theta_1 - a_3 * \cos\theta_1 * \cos\theta_3 - a_4 * \cos\theta_1 * \cos(\theta_3 + \ \theta_4) \tag{2.62}$$

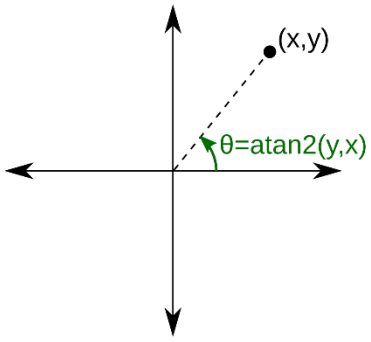
2.5 Inverzní kinematika

Inverzní kinematika je matematická funkce, která provádí zobrazení z prostoru souřadnic kartézského souřadného systému (souřadnice x, y, z) do prostoru kloubových souřadnic (úhly natočení θ_i). Existuje několik způsobů, jak inverzní kinematiku spočítat. Jelikož má jedna noha našeho robota pouze 3 stupně volnosti, lze inverzní kinematiku spočítat pomocí goniometrických funkcí a trojúhelníků, které lze v modelu nohy nalézt.

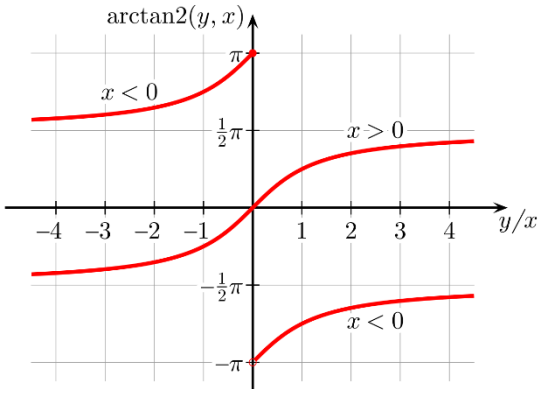
Při výpočtu inverzní kinematiky jedné nohy robota chceme spočítat úhly natočení θ_1, θ_3 *a* θ_4, díky kterým by se koncový bod nohy dostal do požadovaných souřadnic x, y a z vyjádřených v souřadném systému 0 jedné nohy (6).

2.5.1 Funkce atan2

Při sestavování rovnic inverzní kinematiky se často používá funkce atan2, což je inverzní funkce pro goniometrickou funkci tan. Od funkce atan se atan2 liší tím, že bere v potaz znaménka vstupních hodnot (čitatel a jmenovatel podílu $\frac{y}{x}$, viz. obrázek 31). Výstupní hodnotou funkce atan2 je hodnota a, která může nabývat hodnot od $-\pi$ až po $+\pi$. Obor hodnot funkce atan je pouze interval $(-\frac{\pi}{2}; +\frac{\pi}{2})$.



Obrázek 29 – Princip funkce atan2



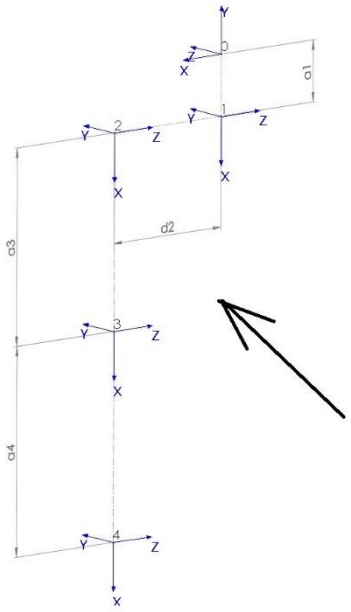
Obrázek 30 – Funkce atan2 (27)

$$\operatorname{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0, \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases}$$

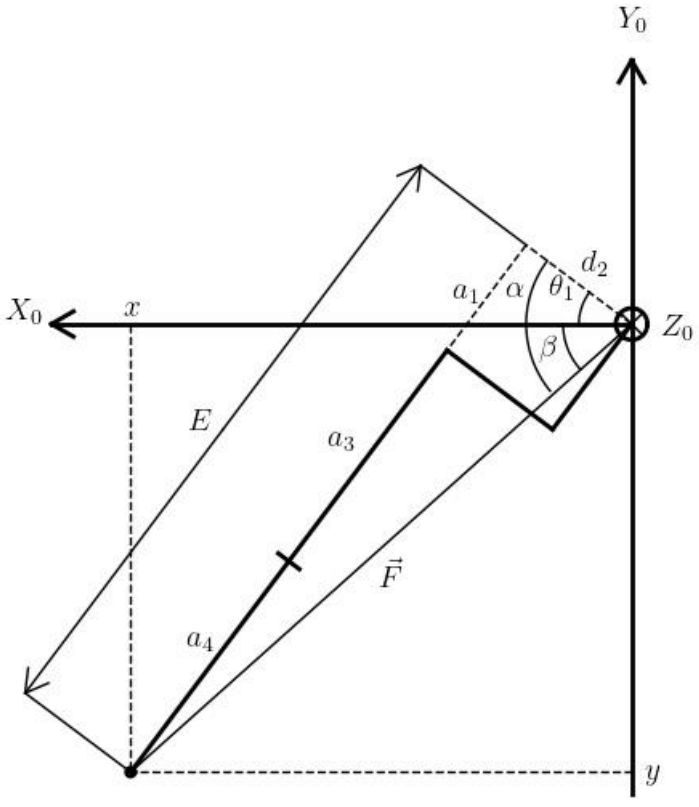
Obrázek 31 – Definice funkce atan2 (27)

2.5.2 Inverzní kinematika pravé nohy

Pro výpočet úhlu θ_1 se použije pohled zepředu, tzn. díváme se kolmo na rovinu X_0Y_0 souřadného systému 0 a Z_0 směřuje směrem dozadu.



Obrázek 32 Směr pohledu na nohu robota při výpočtu úhlu θ_1.



Obrázek 33 – Pohled zepředu na nohu robota

V tomto pohledu je mnoho parametrů, díky kterým lze zjistit úhel θ_1. Délku E lze spočítat z pravoúhlého trojúhelníku s odvěsnami E a d_2 a přeponou $|\overrightarrow{F}|$.

$$\overrightarrow{F} = \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.63}$$

$$E = \sqrt{|\overrightarrow{F}|^2 - d_2^2} = \sqrt{\sqrt{x^2 + y^2}^2 - d_2^2} = \sqrt{x^2 + y^2 - d_2^2} \tag{2.64}$$

Délka E se na tomto obrázku nerovná součtu délek a_1, a_3 *a* a_4, jelikož délky těchto členů jsou v pohledu na obrázku 33 zkreslené. Délka d_2 je v tomto pohledu nezkreslená.

$|\overrightarrow{F}|$ je přeponou druhého pravoúhlého trojúhelníku na obrázku s odvěsnami x a y. Díky těmto dvěma trojúhelníkům lze spočítat úhly α *a* β.

$$\alpha = atan2(E, d_2) \tag{2.65}$$

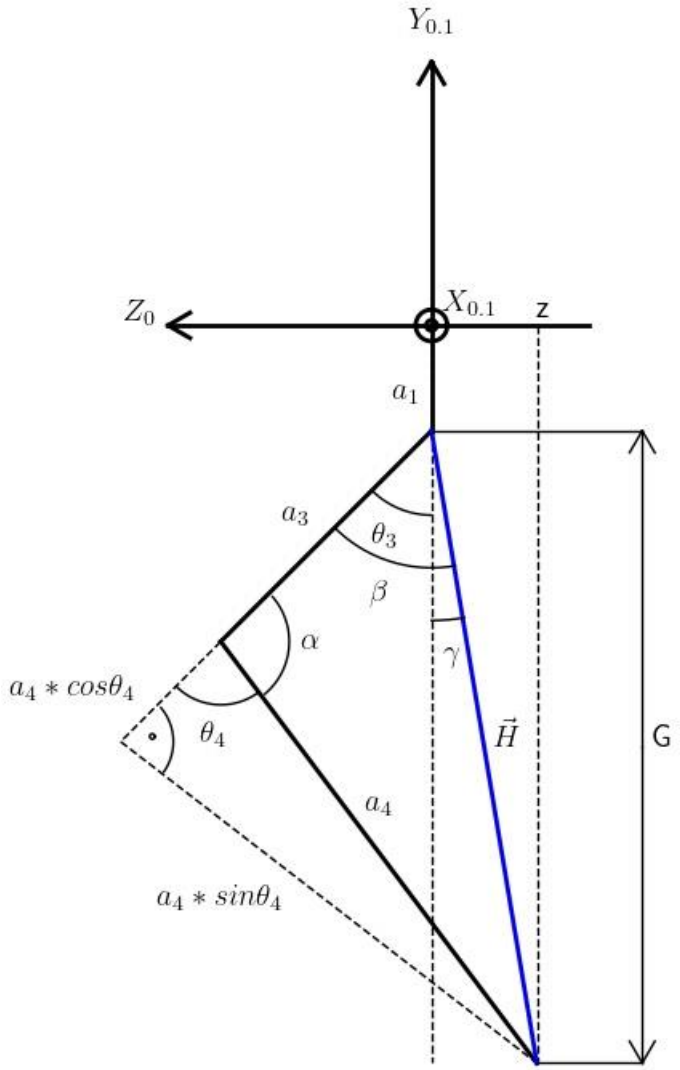
$$\beta = atan2(y, x) \tag{2.66}$$

Úhel θ_1 spočítáme součtem úhlů α *a* β. Hodnota úhlu α je vždy kladná, úhel β má podle obrázku 30 hodnotu zápornou ($x > 0, y < 0$).

$$\theta_1 = \alpha + \beta \tag{2.67}$$

$$\theta_1 = atan2(E, d_2) + atan2(y, x) \tag{2.68}$$

Nyní se na nohu podíváme zleva tak, aby délky členů a_3 *a* a_4 nebyly zkreslené. Jelikož ke zkreslení dochází kvůli rotaci kolem osy Z_0 o úhel θ_1, dojde i ke zkreslení os Y_0 *a* X_0. V obrázku jsou tedy osy $Y_{0.1}$ *a* $X_{0.1}$, která je nezkreslené. Osa Z_0 zůstala nezkreslená.



Obrázek 34 – Pohled na nohu v rovině, která nezkresluje délky a_3 a a_4

Díky tomuto obrázku a trojúhelníkům v něm obsažených lze vypočítat úhly θ_3 *a* θ_4. Délka G je rozdíl délky E a a_1 (viz. obrázek 33)

$$G = E - a_1 \tag{2.69}$$

$$\overrightarrow{H} = \begin{pmatrix} G \\ z \end{pmatrix} \tag{2.70}$$

Pro výpočet úhlu θ_4 použijeme kosinovou větu a funkci atan2.

$$\left|\overrightarrow{H}\right|^2 = a_3^2 + a_4^2 - 2 * a_3 * a_4 * \cos\alpha \tag{2.71}$$

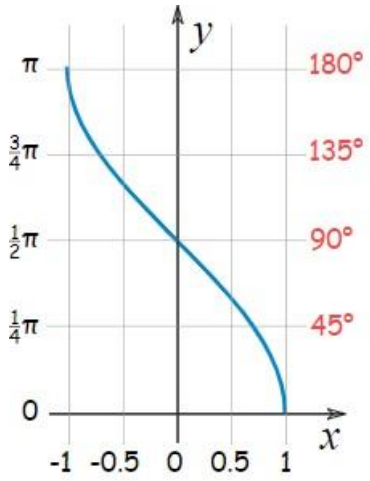
$$\cos\alpha = -\frac{\left|\overrightarrow{H}\right|^2 - a_3^2 - a_4^2}{2 * a_3 * a_4} \tag{2.72}$$

$$\alpha = 180 - \theta_4 \tag{2.73}$$

$$\cos(180 - \theta_4) = -\frac{\left|\overrightarrow{H}\right|^2 - a_3^2 - a_4^2}{2 * a_3 * a_4} \tag{2.74}$$

$$\cos\theta_4 = \frac{\left|\overrightarrow{H}\right|^2 - a_3^2 - a_4^2}{2 * a_3 * a_4} \tag{2.75}$$

Z tohoto vzorce bychom mohli spočítat úhel θ_4. Jelikož je ale obor hodnot funkce acos interval $< 0;\ +\pi >$, nikdy bychom nedostali hodnoty záporné nebo hodnoty větší než π.



Obrázek 35 – Graf funkce acos (28)

Tento problém lze obejít využitím funkce tan.

$$\tan\theta_4 = \frac{\sin\theta_4}{\cos\theta_4} \tag{2.76}$$

$$\cos^2\theta_4 + \sin^2\theta_4 = 1 \tag{2.77}$$

$$\sin^2\theta_4 = 1 - \cos^2\theta_4 \tag{2.78}$$

$$\sin\theta_4 = \sqrt{1 - \cos^2\theta_4} \tag{2.79}$$

$$\tan\theta_4 = \frac{\sqrt{1 - \cos^2\theta_4}}{\cos\theta_4} \tag{2.80}$$

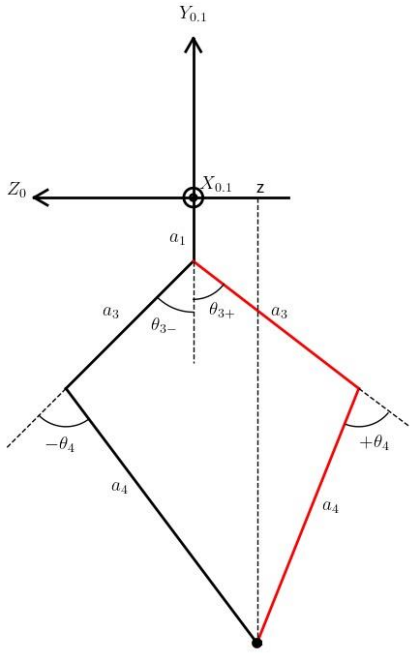
Tato rovnice nám umožňuje použít funkci atan2, čitatele i jmenovatele zlomku totiž známe.

$$\theta_4 = atan2(\sqrt{1 - \cos^2\theta_4}, \cos\theta_4) \tag{2.81}$$

$$D = \frac{\left|\overrightarrow{H}\right|^2 - a_3^2 - a_4^2}{2 * a_3 * a_4} \tag{2.82}$$

$$\theta_4 = atan2(\sqrt{1 - D^2}, D) \tag{2.83}$$

Požadované pozice koncového bodu nohy lze dosáhnout dvěma způsoby. Obě řešení je možno vidět na obrázku 36.



Obrázek 36 – Obě řešení inverzní kinematiky

Abychom získali stejné natočení členu 4 jako na obrázku 34, musí být úhel θ_4 záporný.

Výsledný vzorec pro výpočet úhlu θ_4 je tedy v rovnici (2.84)

$$\theta_4 = -\, atan2(\sqrt{1 - D^2}, D) \tag{2.84}$$

Pro výpočet úhlu θ_3 použijeme dva pravoúhlé trojúhelníky na obrázku 34. První má odvěsny G a z, přeponou je $\left|\overrightarrow{H}\right|$. Druhý trojúhelník má také přeponu $\left|\overrightarrow{H}\right|$, odvěsnami jsou ale $(a_4 * \sin\theta_4)$ a $(a_3 + a_4 * \cos\theta_4)$. Díky těmto trojúhelníkům lze spočítat úhly β a γ.

$$\beta = atan2(a_4 * \sin\theta_4\,, a_3 + a_4 * \cos\theta_4) \tag{2.85}$$

$$\gamma = atan2(z, G) \tag{2.86}$$

Výsledný úhel θ_3 zjistíme výpočtem rozdílu těchto dvou úhlů.

$$\theta_3 = \gamma - \beta \tag{2.87}$$

$$\theta_3 = atan2(z, G) - atan2(a_4 * \sin\theta_4\,, a_3 + a_4 * \cos\theta_4) \tag{2.88}$$

2.5.3 Inverzní kinematika levé nohy

Inverzní kinematika levé nohy by se počítala obdobně jako inverzní kinematika nohy pravé. Jediným rozdílem mezi nohami na obou stranách je parametr d_2, který je v případě pravé nohy záporný, v případě nohy levé je kladný. Rovnice pro výpočet jednotlivých úhlů $\theta_1, \theta_3, \theta_4$ pro levou nohu se od rovnic inverzní kinematiky pro nohu pravou liší jen ve znaménku u parametru d_2 v rovnici (2.68).

$$\theta_1 = atan2(E, -d_2) + atan2(y, x) \tag{2.89}$$

Rovnice pro výpočet úhlů θ_3 *a* θ_4 zůstávají stejné.

2.5.4 Rovnice inverzní kinematiky pro všechny nohy

Pro přehlednost lze úhly $\theta_1, \theta_3, \theta_4$ doplnit o index i, který by označoval, které noze vlastně dané úhly patří. Máme tedy úhly $\theta_{1,i}, \theta_{3,i}, \theta_{4,i}$, se kterými můžeme dále pracovat. Použité indexy pro robota v této odborné práci jsou v tabulce 3.

Noha	Index i
Pravá přední	0
Levá přední	1
Pravá zadní	2
Levá zadní	3

Tabulka 3 – Indexy pro jednotlivé nohy

Díky těmto indexům lze rovnice inverzní kinematiky přepsat tak, aby výpočet inverzní kinematiky nohou proběhl vždy pomocí správných rovnic.

$$\theta_{1,i} = atan2\left(E, d_2 * (-1)^i\right) + atan2(y, x) \tag{2.90}$$

$$\theta_{4,i} = -atan2(\sqrt{1 - D^2}, D) \tag{2.91}$$

$$\theta_{3,i} = atan2(z, G) - atan2(a_4 * \sin\theta_4, a_3 + a_4 * \cos\theta_4) \tag{2.92}$$

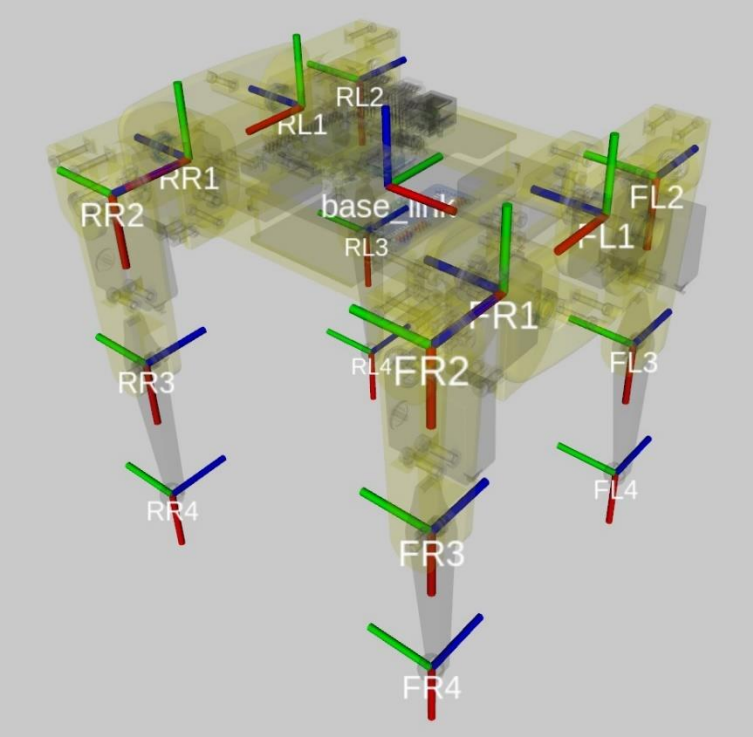
Hodnoty D, E a G se počítají stejně jako v podkapitole 2.5.2.

Rovnice (2.90), (2.91) a (2.92) jsou použity přímo v softwaru robota. Jejich správnost byla experimentálně ověřena výpočtem úhlů θ_1, θ_3 *a* θ_4 pro různé souřadnice koncových bodů a následným dosazením těchto úhlu do rovnic dopředné kinematiky. Výstupem dopředné kinematiky byly hodnoty stejné, jako souřadnice vstupující do výpočtu inverzní kinematiky na začátku.

2.6 Inverzní kinematika celého robota

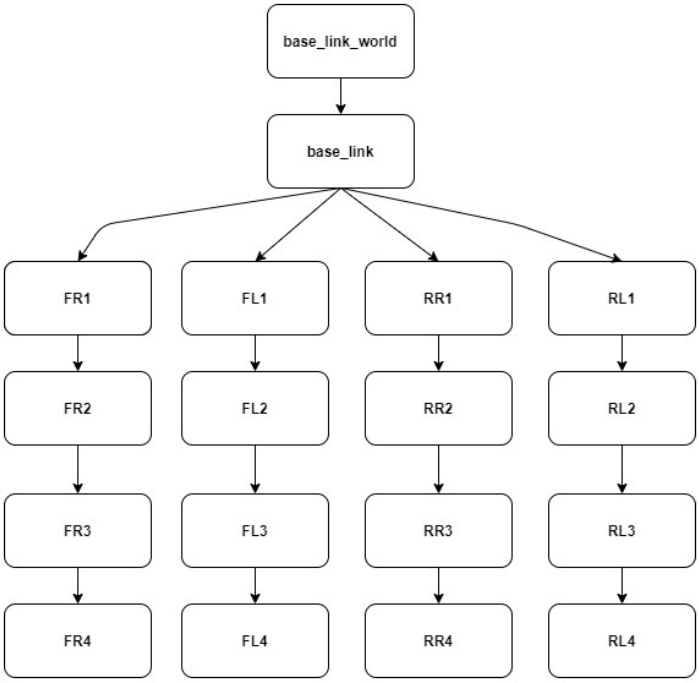
Nyní, když máme rovnice pro výpočet inverzní kinematiky pro jednotlivé nohy, lze vypočítat inverzní kinematiku celého robota. Cílem inverzní kinematiky celého robota je umožnit lokální posouvání nebo otáčení těla robota bez toho, aniž by se vůči zemi měnily polohy koncových bodů nohou robota.

Celý robot se skládá z 18 souřadných systému, které lze vidět na obrázcích 19, 23, 37 a mnoha dalších.



Obrázek 37 – Souřadné systémy

Jednotlivé souřadné systémy mají mezi sebou vztahy, které lze ilustrovat na obrázku, jenž zdánlivě připomíná stromovou datovou strukturu.



Obrázek 38 – Strom souřadných systémů

Souřadnice koncových bodů jednotlivých nohou robota jsou v softwaru vyjádřeny v souřadném systému base_link_world. Abychom mohli vypočítat inverzní kinematiku pro jednu nohu robota, potřebujeme souřadnice koncových bodů vyjádřit v souřadném systému na rameni. Pro pravou přední nohu k tomu lze použít inverzní matici transformační matice ${}^{blw}T_{FR1}$.

$$\overrightarrow{P_{FR1,FR4}} = {}^{FR1}T_{blw} * \overrightarrow{P_{blw,FR4}} = {}^{blw}T_{FR1}^{-1} * \overrightarrow{P_{blw,FR4}} \tag{2.93}$$

Vektor $\overrightarrow{P_{FR1,FR4}}$ vyjadřuje souřadnice potřebné pro výpočet inverzní kinematiky jedné nohy. Transformační matice ${}^{blw}T_{FR1}$ popisuje vztah mezi souřadným systémem base_link_world a FR1. Skládá se ze dvou dílčích transformací.

$${}^{blw}T_{FR1} = {}^{blw}T_{bl} * {}^{bl}T_{FR1} \tag{2.94}$$

Matice ${}^{blw}T_{bl}$ popisuje již zmíněnou lokální transformaci těla a může být ovlivněna šesti parametry – posunutí v ose x, posunutí v ose y, posunutí v ose z, rotace kolem osy x, rotace kolem osy y a rotace kolem osy z souřadného systému base_link_world. Tuto matici lze tedy popsat pomocí tří posunutí a tří Eulerových úhlů.

$$
{}^{blw}T_{bl} = \begin{pmatrix} 1 & 0 & 0 & x_{bl} \\ 0 & 1 & 0 & y_{bl} \\ 0 & 0 & 1 & z_{bl} \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_{roll} & -\sin\alpha_{roll} & 0 \\ 0 & \sin\alpha_{roll} & \cos\alpha_{roll} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \cos\beta_{pitch} & 0 & \sin\beta_{pitch} & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta_{pitch} & 0 & \cos\beta_{pitch} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} \cos\gamma_{yaw} & -\sin\gamma_{yaw} & 0 & 0 \\ \sin\gamma_{yaw} & \cos\gamma_{yaw} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.95}
$$

Matice ${}^{blw}T_{bl}$ se vypočítá dle rovnice (2.95), následně ji invertujeme pomocí rovnice (2.31). Je důležité, aby v rovnici (2.95) následovala rotace až po posunutí, jinak by nedošlo k požadovanému zachování souřadnic koncových bodů vůči zemi.

Transformační matice ${}^{bl}T_{FR1}$ obsahuje informace o pozici a orientaci souřadného systému FR1 v souřadném systému base_link. Je zcela popsána geometrií těla.

$$
{}^{bl}T_{FR1} = \begin{pmatrix} 0 & 0 & -1 & 0.5 * l_{robot} \\ -1 & 0 & 0 & -0.5 * w_{robot} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.96}
$$

V rovnici (2.96) je hodnotou l_{robot} vyjádřena délka těla robota, proměnná w_{robot} vyjadřuje šířku těla. Část transformační matice ${}^{bl}T_{FR1}$ popisující orientaci systému FR1 v systému base_link není jednotkovou maticí, souřadný systém FR1 je totiž vůči systému base_link určitým způsobem natočen.

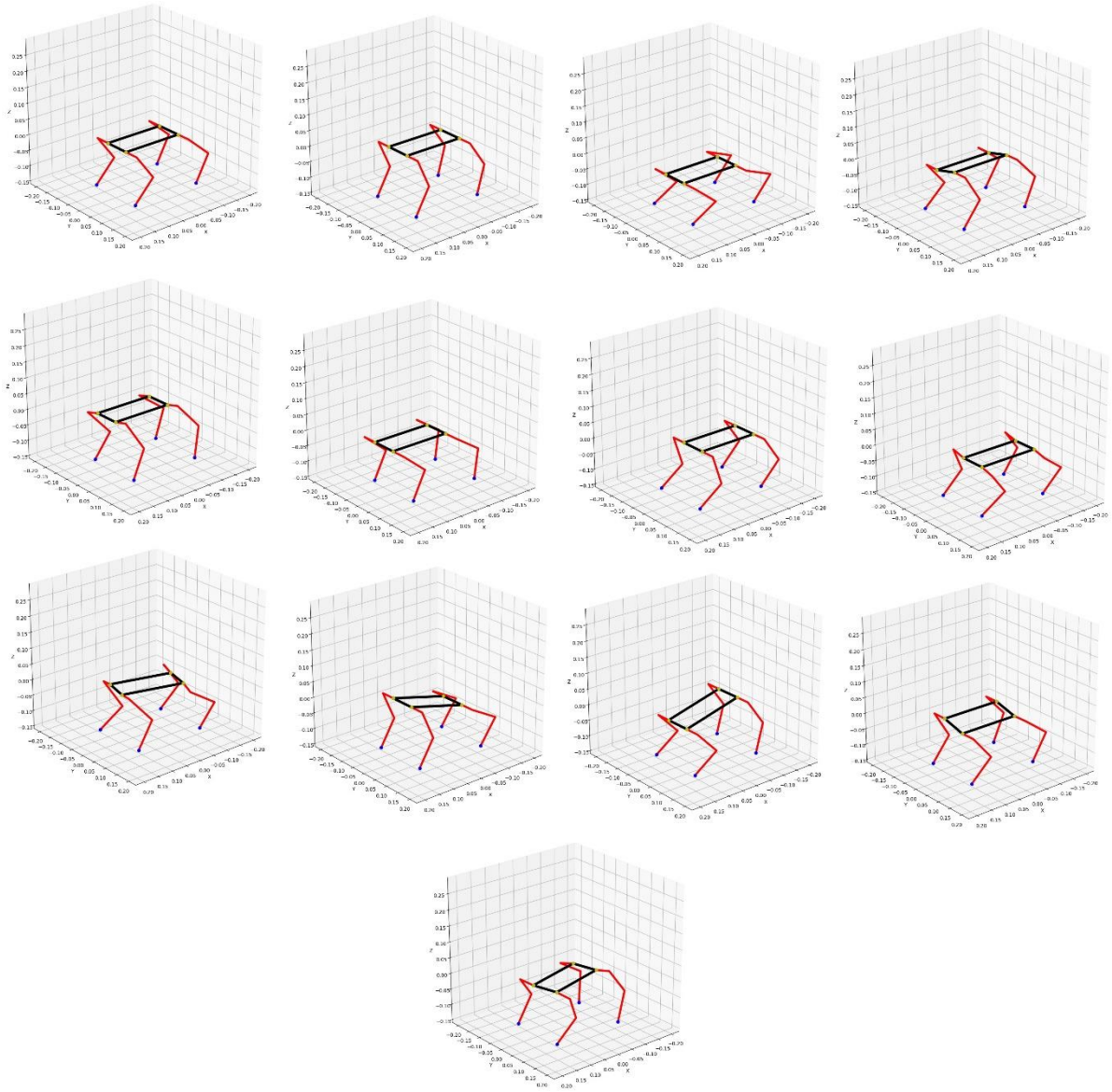
Podobně jako v rovnici (2.96) lze popsat i ostatní transformační matice, díky kterým se lze dostat ze souřadného systému base_link do souřadného systému ramena pravé zadní nohy RR1 (2.97), levé přední nohy FL1 (2.98) a levé zadní nohy RL2 (2.99).

$$
{}^{bl}T_{RR1} = \begin{pmatrix} 0 & 0 & -1 & -0.5 * l_{robot} \\ -1 & 0 & 0 & -0.5 * w_{robot} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.97}
$$

$$
{}^{bl}T_{FL1} = \begin{pmatrix} 0 & 0 & -1 & 0.5 * l_{robot} \\ -1 & 0 & 0 & 0.5 * w_{robot} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.98}
$$

$$
{}^{bl}T_{RR1} = \begin{pmatrix} 0 & 0 & -1 & -0.5 * l_{robot} \\ -1 & 0 & 0 & 0.5 * w_{robot} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.99}
$$

Tyto matice nám umožňují dosáhnout pohybů těla bez toho, aniž by se pozice koncových bodů jednotlivých nohou vyjádřených v souřadném systému base_link_world nějak změnili. Obrázek 39 zobrazuje několik póz, kterých lze pomocí inverzní kinematiky celého robota dosáhnout.



Obrázek 39 – Různé pozice a orientace těla robota při zachování souřadnic koncových bodů jednotlivých nohou

```python
#!/usr/bin/env python3
#Author: Jakub Jon

# Import potřebných funkcí
import numpy as np
from math import sqrt, atan2, sin, cos, pi
from RoboticsUtilities.Transformations import homog_transform_inverse,homog_transform

class InverseKinematics(object):
    def __init__(self, bodyDimensions, legDimensions):

        # Délka a šířka těla robota (vzdálenosti mezi souřadnými systémy na ramenech)
        self.bodyLength = bodyDimensions[0]
        self.bodyWidth = bodyDimensions[1]

        # Délky členů nohou: l1 = a1, l2 = d2, l3 = a3, l4 = a4
        self.l1 = legDimensions[0]
        self.l2 = legDimensions[1]
        self.l3 = legDimensions[2]
        self.l4 = legDimensions[3]

    def get_local_positions(self,leg_positions,dx,dy,dz,roll,pitch,yaw):
        """
        Funkce pro převod souřadnich koncových bodů nohou ze souřadného
        systému base_link_world do souřadných systému na ramenech.
        """

        # Vstup, souřadnice koncových bodů nohou
        leg_positions = (np.block([[leg_positions],[np.array([1,1,1,1])]])).T

        # Transformační matice: base_link_world => base_link
        T_blwbl = homog_transform(dx,dy,dz,roll,pitch,yaw)

        # Transformační matice: base_link_world => FR1
        T_blwFR1 = np.dot(T_blwbl, homog_transform(+0.5*self.bodyLength,
                    -0.5*self.bodyWidth,0,pi/2,-pi/2,0))

        # Transformační matice: base_link_world => FL1
        T_blwFL1 = np.dot(T_blwbl, homog_transform(+0.5*self.bodyLength,
                    +0.5*self.bodyWidth,0,pi/2,-pi/2,0))

        # Transformační matice: base_link_world => RR1
        T_blwRR1 = np.dot(T_blwbl, homog_transform(-0.5*self.bodyLength,
                    -0.5*self.bodyWidth,0,pi/2,-pi/2,0))

        # Transformační matice: base_link_world => RL1
        T_blwRL1 = np.dot(T_blwbl, homog_transform(-0.5*self.bodyLength,
                    +0.5*self.bodyWidth,0,pi/2,-pi/2,0))

        # Souřadnice koncového bodu pravé přední nohy
        pos_FR = np.dot(homog_transform_inverse(T_blwFR1),leg_positions[0])

        # Souřadnice koncového bodu levé přední nohy
        pos_FL = np.dot(homog_transform_inverse(T_blwFL1),leg_positions[1])

        # Souřadnice koncového bodu pravé zadní nohy
        pos_RR = np.dot(homog_transform_inverse(T_blwRR1),leg_positions[2])

        # Souřadnice koncového bodu levé zadní nohy
        pos_RL = np.dot(homog_transform_inverse(T_blwRL1),leg_positions[3])

        return(np.array([pos_FR[:3],pos_FL[:3],pos_RR[:3],pos_RL[:3]]))
```

```
64      def inverse_kinematics(self,leg_positions,dx,dy,dz,roll,pitch,yaw):
65          """
66          Funkce pro výpočet inverzní kinematiky robota.
67          """
68
69          # Výpočet lokálních souřadnic na ramenech
70          positions = self.get_local_positions(leg_positions,dx,dy,dz,
71                                               roll,pitch,yaw)
72          angles = []
73
74          # Výpočet inverzní kinematiky pro každou nohu
75          for i in range(4):
76
77              x = positions[i][0]
78              y = positions[i][1]
79              z = positions[i][2]
80
81              F = sqrt(x**2 + y**2 - self.l2**2)
82              G = F - self.l1
83              H = sqrt(G**2 + z**2)
84
85              theta1 = atan2(y,x) + atan2(F,self.l2 * (-1)**i)
86
87              D = (H**2 - self.l3**2 - self.l4**2)/(2*self.l3*self.l4)
88
89              theta4 = -atan2((sqrt(1-D**2)),D)
90
91              theta3 = atan2(z,G) - atan2(self.l4*sin(theta4),
92                                          self.l3 + self.l4*cos(theta4))
93
94              angles.append(theta1)
95              angles.append(theta3)
96              angles.append(theta4)
97
98          return angles
```

Obrázek 40 – Program pro výpočet inverzní kinematiky

3 Návrh a konstrukce 1 nohy robota

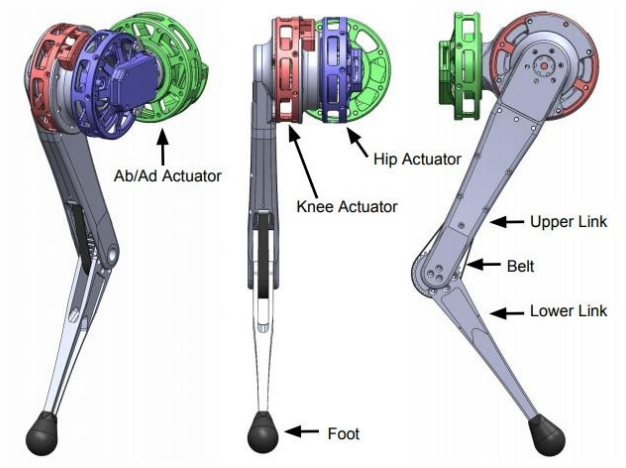
Při návrhu nohy robota se vychází z rovnice pro výpočet momentu setrvačnosti.

$$I = \int r^2 dm \tag{3.1}$$

$$M_k = I * \frac{d\omega}{dt} = I * \alpha \tag{3.2}$$

Rovnice (3.2) popisuje potřebný kroutící moment k udělení tělesu o momentu setrvačnosti I úhlové zrychlení α. Jedná se o ekvivalent Newtonova druhého pohybového zákona ($F = m * a$) pro rotační pohyb.

Cílem je vytvořit takovou konstrukci, která by moment setrvačnosti minimalizovala. Ve většině případech se nohy robotů staví tak, aby byla veškerá hmota umístěna v rameni a díky tomu lze použít lehčí a méně výkonné motory. Motory jsou totiž často těmi členy konstrukce nohy robota, které ke zvýšení momentu setrvačnosti přispívají nejvíce.

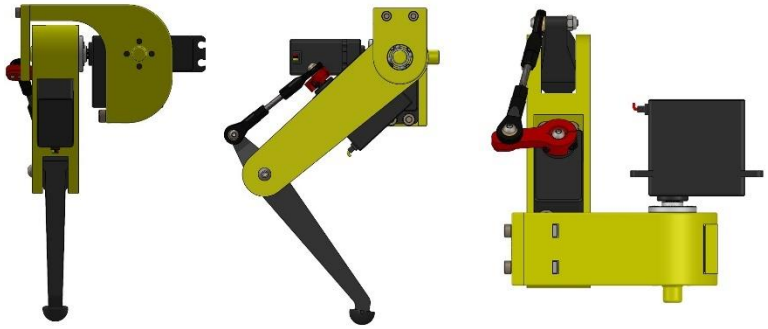


Obrázek 41 – Konstrukce nohy robota Mini Cheetah (4)

Při návrhu konstrukce jedné nohy se vycházelo z konstrukce nohy robota Mini Cheetah a konstrukce nohy hobby čtyřnohého robota od Martina Triendla.

Obrázek 42 – Čtyřnohý robot Martina Triendla (29)

Obrázek 43 – Noha robota



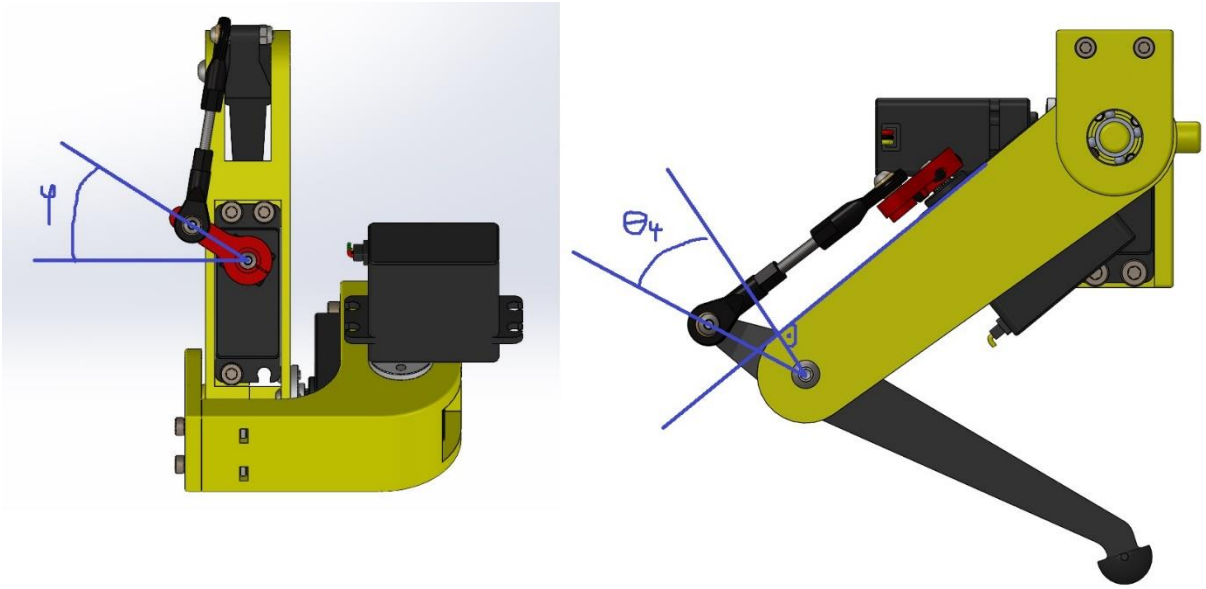
Obrázek 44 – Kolmé pohledy na nohu robota

Jak lze vidět na obrázcích 42 a 43, všechny servomotory jsou umístěny blízko ramene robota, aby se zmenšil moment setrvačnosti a díky tomu potřebný kroutící moment k otáčení nohy.

Rameno a horní část nohy mají výstupky, které se vkládají do ložisek. Díky tomu se docílí menšího ohybového momentu na hřídelích servomotorů.

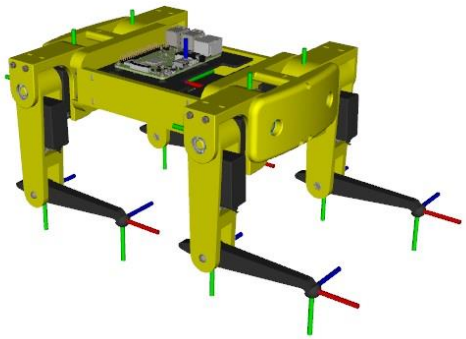
Servomotor ovládající dolní část nohy robota je integrován do horní části nohy a je spojen s dolní částí pomocí pákového mechanismu. Tento pákový mechanismus se skládá ze dvou kulových čepů a části závitové tyče.

Výsledkem inverzní kinematiky jsou tři úhly, které říkají, jak se jednotlivé členy nohy mají natočit. Výsledné úhly θ_1 *a* θ_3 lze přímo nastavit na servomotorech, pro úhel θ_4 to tak ale nelze udělat, jelikož úhel nastavení servomotoru se kvůli pákovému mechanismu nerovná úhlu nastavení spodního členu nohy. Potřebujeme tedy funkci f, která by úhel spodního členu nohy θ_4 přepočítávala na úhly servomotoru φ.



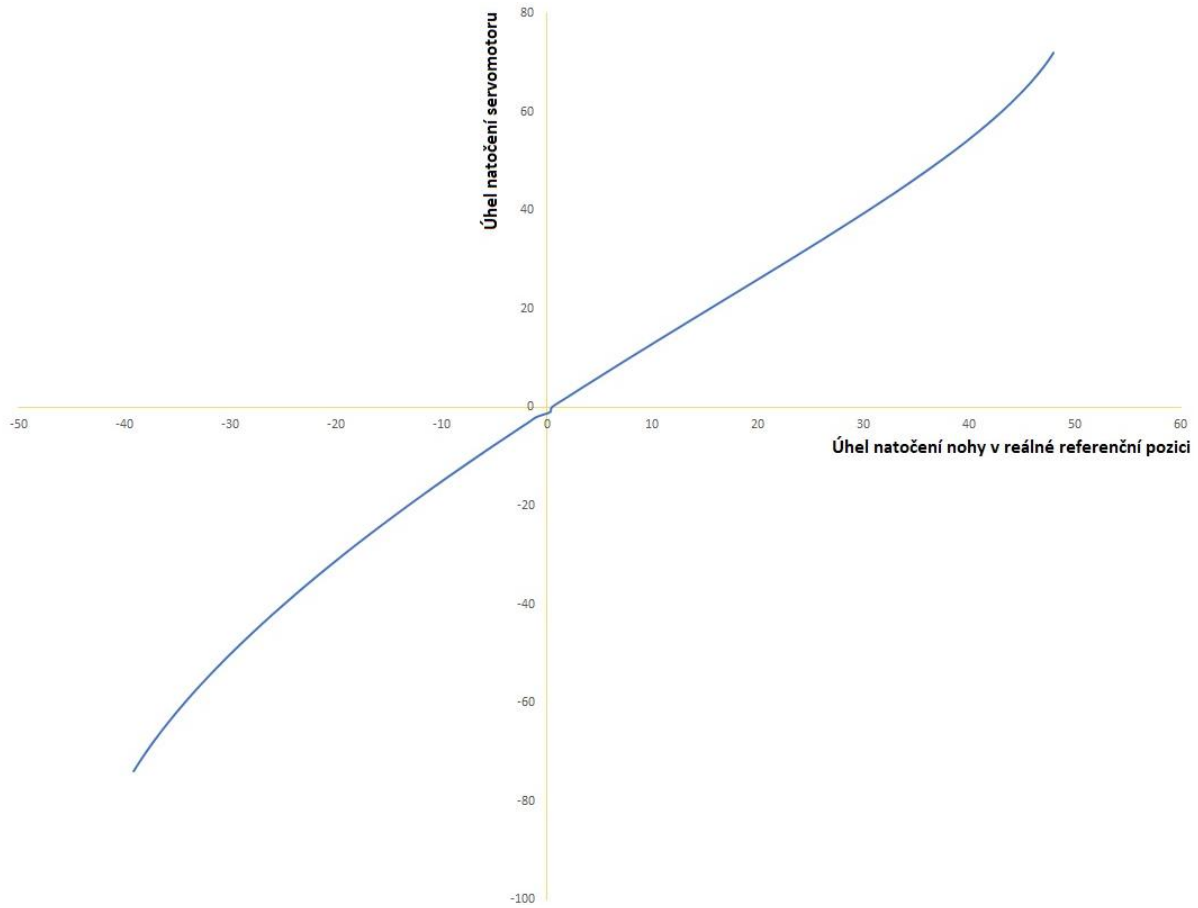
Obrázek 45 – Nutný přepočet úhlu θ na úhel φ

Předpokládejme, že určitá funkce pro přepočet úhlu θ_4 na úhel φ existuje. Jelikož se úhel θ_4 pohybuje v rozmezí – 130 až – 40 stupňů, lze tuto funkci lokálně aproximovat. Pro snadnější kalibraci servomotorů budeme uvažovat, že referenční úhel spodní části nohy není 0 stupňů, jelikož takový úhel je nedosažitelný. Referenční pozici místo toho posuneme do – 90 stupňů a k výstupní hodnotě inverzní kinematiky θ_4 tedy vždy budeme přičítat 90 stupňů. Nová referenční pozice je vidět na obrázku 45. Je to pozice, které chceme při kalibraci servomotorů dosáhnout a je to tedy stav, ve kterém mají nastavené pozice na servomotorech hodnotu 0.



Obrázek 46 – Reálná referenční pozice robota

Abychom dostali již zmíněnou funkci přepočtu f, potřebujeme znát několik hodnot úhlů nohy a jejich odpovídající úhly nastavení servomotoru. Tyto hodnoty se pohybují v blízkosti reálné referenční pozice a experimentálně lze zjistit, že daná funkce přepočtu vypadá podobně jako na obrázku 46.



Obrázek 47 – Funkce pro přepočet úhlu θ_4 na úhel servomotoru φ

Na ose x jsou hodnoty úhlu θ_4, osa y označuje úhly servomotoru φ. Každý bod na tomto grafu má tedy souřadnice $[\theta_4, \varphi]$. Pokud získáme dostatečné množství těchto bodů, můžeme funkci převodu lokálně popsat pomocí polynomické funkce. V našem případě budeme používat kubickou funkci $a * \theta_4^3 + b * \theta_4^2 + c * \theta_4 + d = \varphi$, pro zjišťění parametrů a, b, c, a d použijeme metodu nejmenších čtverců.

Pro každý bod I = $[\theta_{4,i}, \varphi_i]$ v ideálním případě platí následující vztah:

$$a * \theta_{4,i}^3 + b * \theta_{4,i}^2 + c * \theta_{4,i} + d = \varphi_i \tag{3.3}$$

V případě, že jsme našli n takovýchto bodů, lze sestavit n rovnic, které mají stejný tvar jako rovnice (3.3). Získaly bychom tedy soustavu rovnic, kterou lze zapsat ve tvaru součinu matice a vektoru.

$$P * Q = R \tag{3.4}$$

$$P = \begin{pmatrix} \theta_{4,1}^3 & \theta_{4,1}^2 & \theta_{4,1} & 1 \\ \theta_{4,2}^3 & \theta_{4,2}^2 & \theta_{4,2} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ \theta_{4,n}^3 & \theta_{4,n}^2 & \theta_{4,n} & 1 \end{pmatrix} \tag{3.5}$$

$$Q = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \tag{3.6}$$

$$R = \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_3 \end{pmatrix} \tag{3.7}$$

V případě, že je matice P invertovatelná, lze parametry a, b, c a d získat pomocí rovnice (3.8).

$$Q = P^{-1} * R \tag{3.8}$$

Bohužel ve většině případů matice P invertovatelná není, rovnici (3.8) proto použít nemůžeme. Přišlo se ale na způsob, jak získat takové parametry a, b, c a d, které minimalizují součet druhých mocnin odchylek vypočtených hodnot od potřebných hodnot. Jedná se o problém konvexní optimalizace, který má jedno optimální řešení. Toto řešení lze popsat pomocí rovnice (3.9).

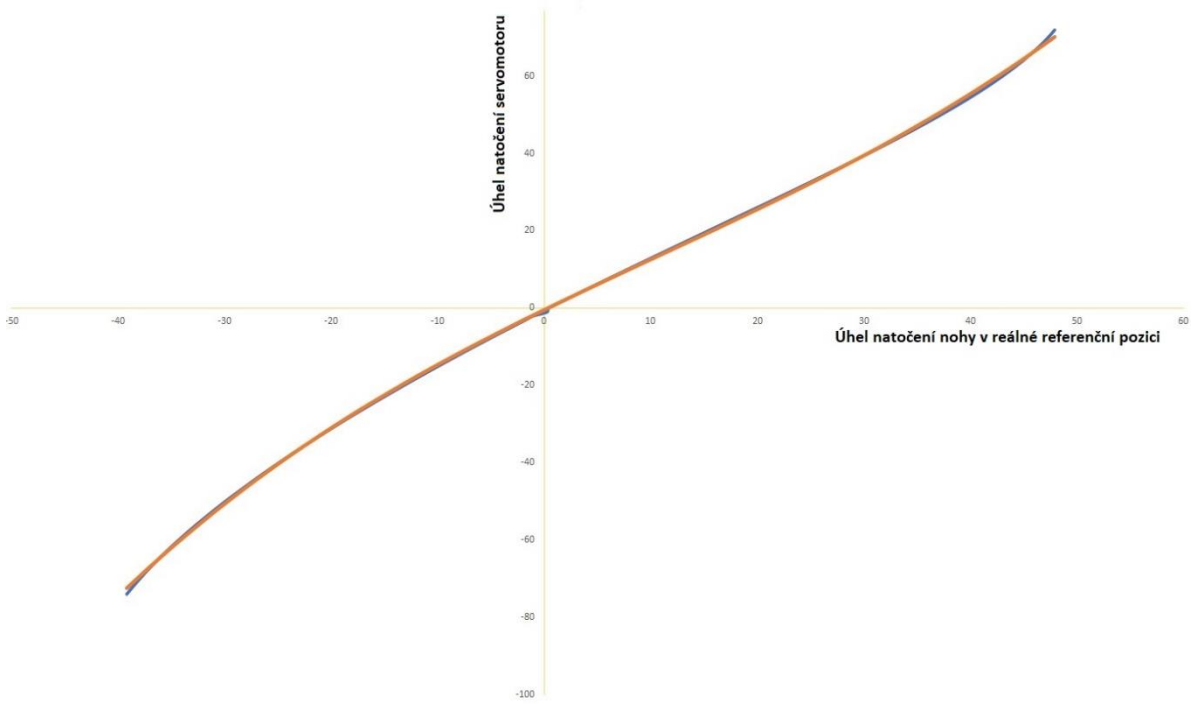
$$Q = P^{+} * R \tag{3.9}$$

Matice P^{+} je takzvaná pseudoinverzní matice, kterou lze jednoduše spočítat pomocí rovnice (3.10). Její odvození lze najít v (7) nebo (8).

$$P^{+} = (P^T * P)^{-1} * P^T \tag{3.10}$$

$$Q = (P^T * P)^{-1} * P^T * R \tag{3.11}$$

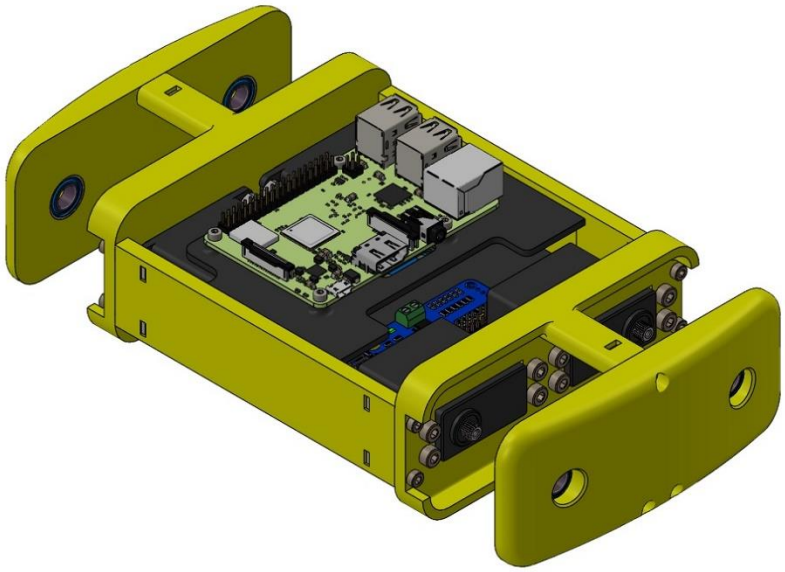
Z rovnice (3.11) získáme parametry a, b, c a d, které můžeme použít v rovnici (3.3) a získat tak aproximovanou hodnotu natočení servomotoru odpovídající úhlu natočení θ_4. Porovnání aproximované a reálné funkce f pro přepočet lze vidět na obrázku 47.



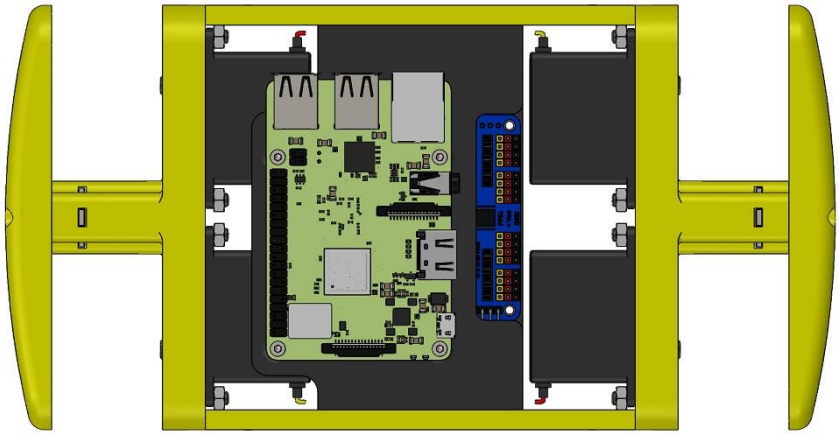
Obrázek 48 – Porovnání reálné funkce přepočtu (modrá) a aproximované funkce (oranžová)

4 Návrh a konstrukce těla robota

Konstrukce těla robota musí být navržena tak, aby se na ní lehce daly přidělat jednotlivé nohy a zároveň aby měla dostatečně prostoru pro veškerou elektroniku potřebnou pro napájení a ovládání robota.



Obrázek 49 – Tělo robota - 3D pohled



Obrázek 50 – Tělo robota - pohled shora

Servomotory pohybující ramenem robota se pomocí šroubů a matic připevňují k rámu tak, aby byla hřídel servomotoru souosá s ložiskem, do kterého se nasouvá výstupek na rameni, díky kterému, stejně jako u nohy robota, se zmenšuje ohybový moment na hřídeli servomotoru.

Uprostřed těla jsou dvě černé desky, na které se upíná veškerá elektronika. Tyto desky zároveň zpevňují celou konstrukci a díky gumičkách, které jsou nasazeny na spodní černé desce lze použitou baterii jednoduše připevnit k tělu robota.

5 Elektronika

Náš čtyřnohý robot obsahuje několik elektronických komponentů, které jsou představeny v této kapitole. Schéma zapojení elektroniky je v příloze v souboru „Schéma zapojení elektroniky.jpg“.

5.1 Raspberry Pi 3 B+

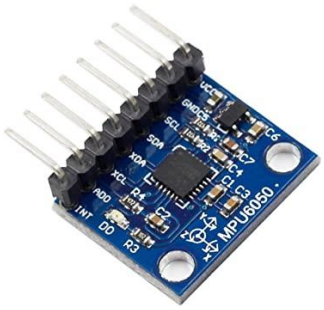
Raspberry Pi 3 B+ je jednočipový počítač, který byl vyvinut britskou firmou Raspberry Pi Foundation (9). Je to zařízení, se kterým se často setkáváme v menších projektech, jelikož je levné a poměrně výkonné. Výhodou je možnost instalace operačního systému, což byl také jeden z důvodů pro použití tohoto jednodeskového počítače při stavbě vytvářeného robota.

Obrázek 51 – Raspberry Pi 3B+ (30)

Jako operační systém byl pro tento počítač zvolen Linux Server 20.04 (Focal Fossa), jelikož je rychlý, jednoduchý k používání a podporuje programovací jazyky Python i C++, což jsou jazyky, které byly při psaní softwaru pro robota použity. Hlavní funkcí Raspberry Pi je sbírání dat z ostatních komponentů (MPU6050, PS4 ovladač), výpočet inverzní kinematiky s ohledem na přijímaná data a výpočet výsledného signálu pro modul PCA9685, ovládajícího servomotory CLS6336HV.

5.2 MPU6050

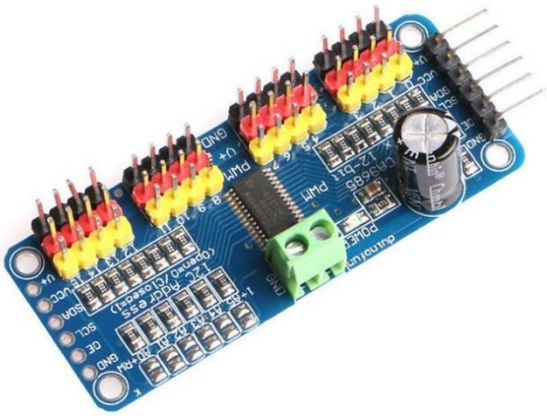
Modul MPU6050 je takzvanou inerciální měřicí jednotkou, která se skládá z gyroskopu a akcelerometru. Díky gyroskopu lze získat úhlové rychlosti kolem os X, Y a Z lokálního souřadného systému. Akcelerometr měří lineární zrychlení v těchto osách. Získané hodnoty lze využít pro zjištění orientace senzoru a díky tomu můžeme při výpočtu inverzní kinematiky kompenzovat případné naklápění robota. Tento modul používá pro komunikaci s Raspberry Pi sériovou sběrnici I2C.



Obrázek 52 – Modul MPU6050 (31)

5.3 PCA9685

Modul PCA9685 je elektronický komponent, který umožňuje ovládat až 16 servomotorů najednou. Stejně jako MPU6050 používá PCA9685 pro komunikaci s počítačem sériovou sběrnici I2C. Pro napájení jednotlivých servomotorů používá externí zdroj napětí.



Obrázek 53 – Modul PCA9685 (32)

Jednotlivé servomotory se připojují třemi vodiči. Jedná se o GND, V+ a PWM vodiče. PWM vodič slouží pro zasílání PWM signálu, který umožňuje ovládání úhlu natočení hřídelí připojených servomotorů.

5.4 BEC 5A FLYCOLOR

Spínaný stabilizátor napětí BEC 5A FLYCOLOR je elektronický modul, jehož hlavním úkolem je přeměna vstupního napětí na 5 V, což je napětí potřebné k napájení Raspberry Pi a modulu PCA9685. Vstupní napětí do modulu se může pohybovat od 6 V až do 33.6 V. Tento modul je vhodný pro napájení komponentů z baterií, u kterých vlivem vybíjení dochází ke snižování napětí. Byl zvolen po několika neúspěšných pokusech o použití integrovaného obvodu LM338, u kterého vlivem zátěže docházelo ke snižování napětí na výstupu.

Obrázek 54 – BEC 5A FLYCOLOR (33)

5.5 Power X6 3300 mAh 2S 35C (70C)

Pro napájení servomotorů a ostatní použité elektroniky byla použita Li-Po baterie Power X6 3300 mAh 2S 35C. Tato baterie byla zvolena pro svou kapacitu 3300 mAh a nízkou hmotnost 158 g. Jmenovité napětí je 7.4 V, což umožňuje servomotory napájet napřímo. Paralelně k napájení servomotorů je připojen modul BEC 5A FLYCOLOR, který vstupní napětí mění na pět voltů pro ostatní elektroniku.

Obrázek 55 – Baterie Power X6 3300 mAh 2S 35C (70C) (34)

Pro upevnění baterie k tělu robota byly použity gumičky, které umožňují velmi jednoduchou a rychlou montáž, případně i demontáž akumulátoru.

5.6 Servomotory CLS6336HV

Servomotory CLS6336HV jsou použity pro tvorbu pohybů jednotlivých členů robota. Jedná se o digitální servomotory s poměrně velkým kroutícím momentem (36 kg*cm) a úhlovou rychlostí (0.11 $s/60°$) při napětí 7.4 V. Při výběru servomotorů hrál roly kroutící moment a rychlost otáčení výstupního hřídele. Na trhnu existuje mnoho servomotorů s velkým kroutícím momentem, ale malou rychlostí otáčení a naopak. Cílem bylo získat servomotor, který by udržel pozici při zatížení vahou robota a zároveň by byl dostatečně rychlý. Motory CLS6336HV byly již v minulosti použity při stavbě čtyřnohých robotů, proto jsme se rozhodli právě pro ně.

Obrázek 56 – Servomotor CLS6336HV (35)

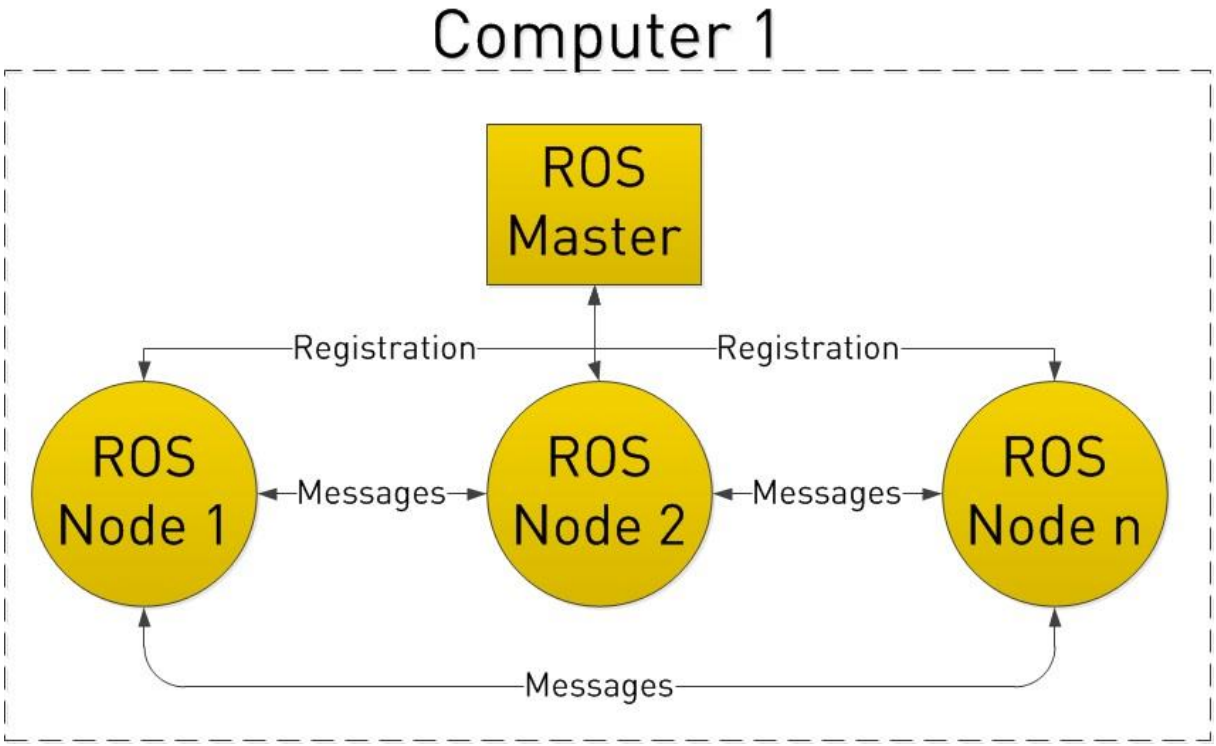
6 Software

Abychom mohli postaveného robota rozpohybovat, potřebujeme k tomu řídící systém, který bude robotovi říkat, jak se pohybovat a bude při tom brát v potaz přijatá data z ostatních komponentů (PS4 ovladač, MPU6050). Základ celého ovládacího systému bude tvořit ROS (Robotic Operating System).

Obrázek 57 – Logo systému ROS (36)

6.1 Robotic Operating System

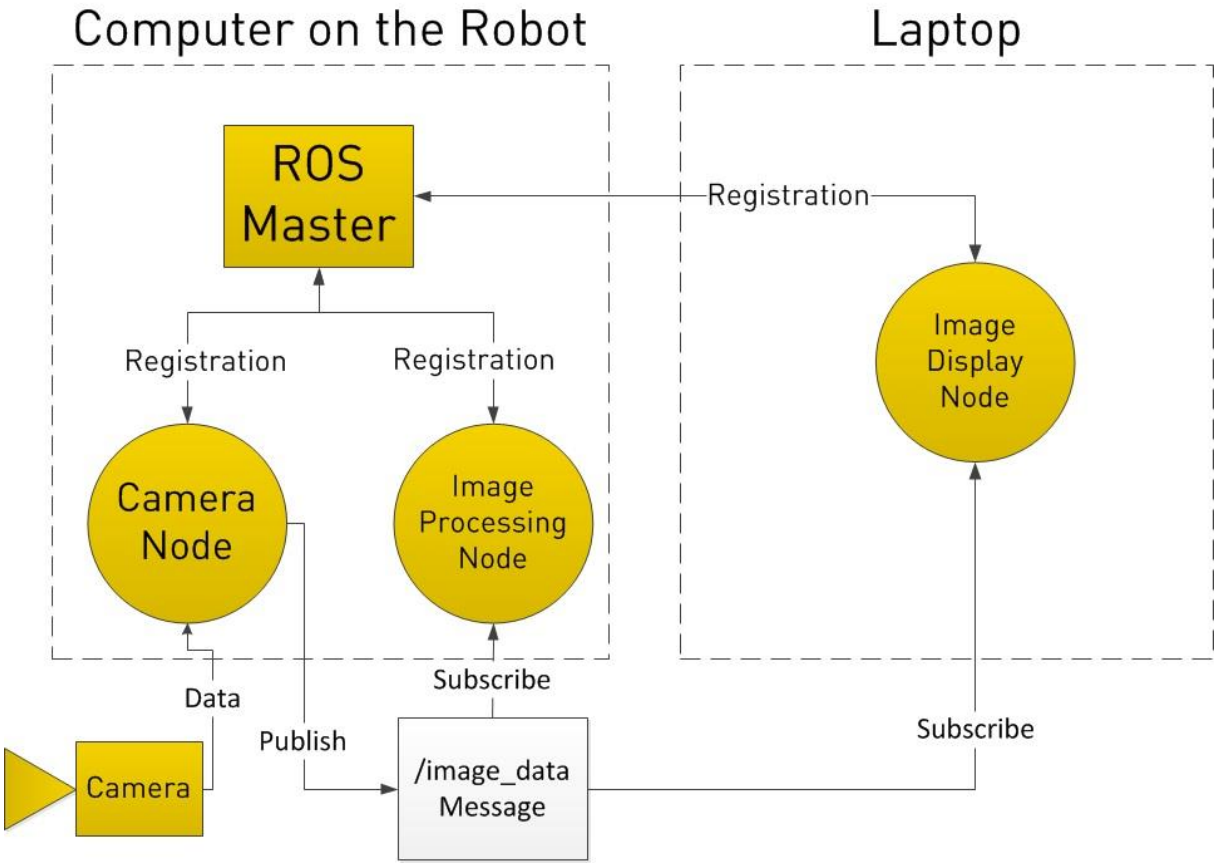
Robotic Operating System, neboli ROS, je open-source framework vyvíjený firmou Open Source Robotics Foundation (10), jehož účelem je usnadnění psaní softwaru pro robotické systémy. Hlavní myšlenkou tohoto frameworku je rozdělení komplexního programu na menší části, které současně běží v pozadí a komunikují spolu pomocí zpráv, které mají určité jméno a obsahují určitý datový typ či datovou strukturu s daty. Výsledná architektura programu vypadá jako graf.



Obrázek 58 – Princip ROS (11)

Základem celého systému je uzel zvaný ROS Master, který slouží k registraci ostatních uzlů grafu. ROS Master si ukládá informace o tom, jaký typ zprávy uzel posílá nebo přijímá. Najde-li se dvojice uzlů, z nichž jeden přijímá zprávy se jménem x a datovým typem y a druhý tu samou zprávu posílá, pak je ROS Master spojí pomocí hrany. Po vytvoření hrany spolu tyto dva programy mohou komunikovat.

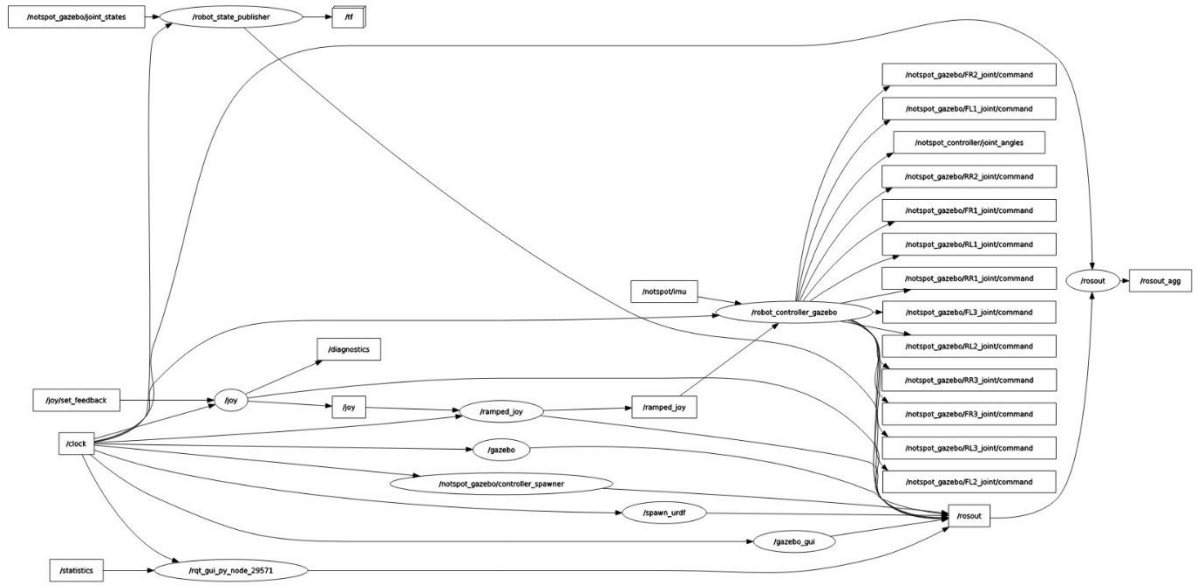
Jednou z mnoha výhod tohoto frameworku je možnost použití mnoha počítačů, které komunikují se stejným ROS Masterem. Díky tomu může program na počítači A komunikovat s programem na počítači B, jelikož oba mají stejného ROS Mastera. Výběr ROS Mastera se provádí volbou jeho adresy.



Obrázek 59 – Komunikace mezi uzly na více počítačích (11)

Další výhoda spočívá v normalizaci zpráv. Pokud již nějaký developer vytvořil program pro detekci obličejů, který přijímá určitou zprávu x a posílá zprávy y, pak tento program mohu použít i bez toho, aniž bych věděl, co se děje uvnitř programu. To je výhodné nejen v tom, že i ty nejkomplikovanější programy může použít úplný začátečník, ale také v tom, že často používané algoritmy a programy se již nemusí programovat znovu pro každý projekt, který daný člověk nebo daná firma vytváří. Stačí tedy použít starý program a komunikovat s ním pomocí patřičných zpráv.

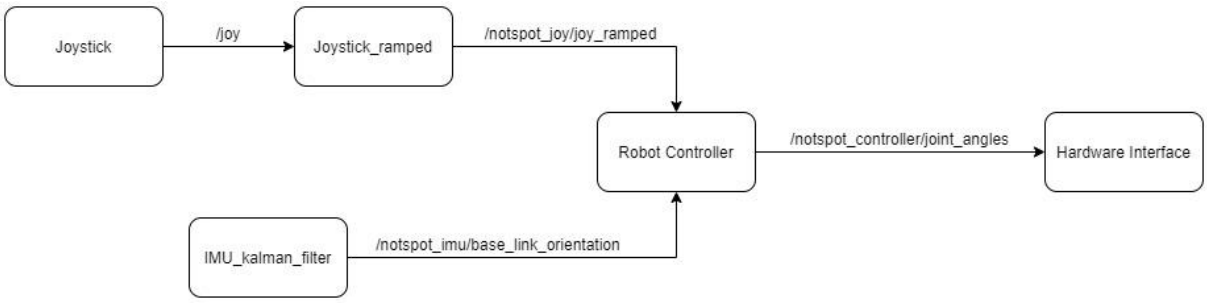
Výsledný program může mít až několik set uzlů, pokud se jedná o velmi komplikovaný systém, uzlů v grafu bývá až tisíc.



Obrázek 60 – Příklad většího programu v systému ROS

Existuje několik verzí systému ROS, v této odborné práci byla použita verze Noetic.

6.2 Struktura programu čtyřnohého robota



Obrázek 61 – Struktura použitého programu ROS

Na obrázku 60 je vidět použitá struktura programu v systému ROS. Na schématu chybí ROS Master, v komunitě ROS se ale předpokládá, že je součástí jakéhokoli programu, a proto není nutné ho zakreslovat.

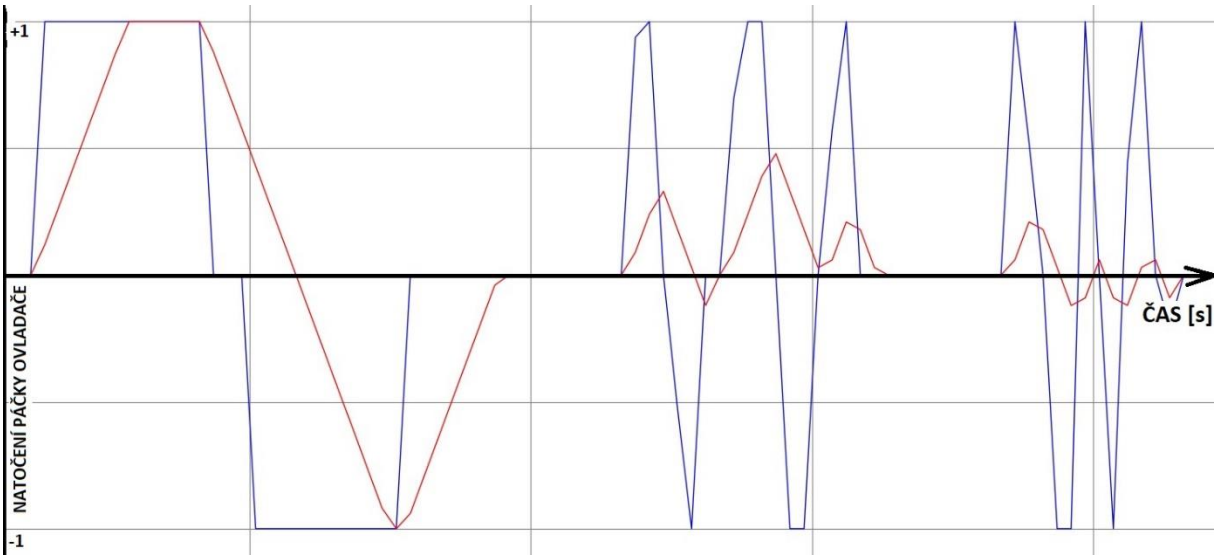
6.2.1 Joystick

Program Joystick je součástí systému ROS, který je používán v mnoha programech. Jedná se o uzel, který komunikuje s připojeným PS4 ovladačem a posílá zprávy s jménem /joy, které obsahují stav jednotlivých páček a tlačítek na ovladači. Tato data jsou následně použita v programu Joystick_ramped.

Obrázek 62 – PS4 ovladač (37)

6.2.2 Joystick_ramped

Joystick_ramped je uzel, který zpracovává příchozí data z PS4 ovladače tak, že veškeré pohyby páček „zjemňuje“. Pokud tedy uživatel velmi rychle překlopí páčku PS4 ovladače z jedné strany na druhou, dochází k okamžité přeměně stavu páčky, mi ale chceme pozvolnou změnu. Tato pozvolná změna je realizována právě v tomto uzlu.



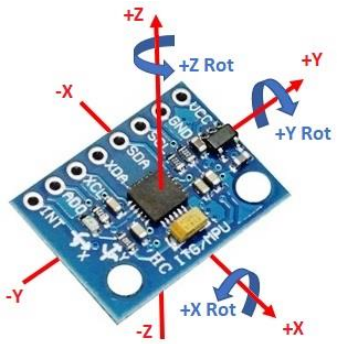
Obrázek 63 – Porovnání dat z uzlu Joystick (modrá čára) a uzlu Joystick_ramped (červená čára) pro jednu páčku ovladače

Výstupní data z uzlu Joystick_ramped (list čísel s hodnotami od -1 do + 1) jsou použita například pro plynulou změnu rychlosti chůze robota. Rychlost přechodu ze stavu A v čase t_1 do stavu B v čase t_2 ($t_2 > t_1$) je nastavitelná.

6.2.3 IMU_kalman_filter

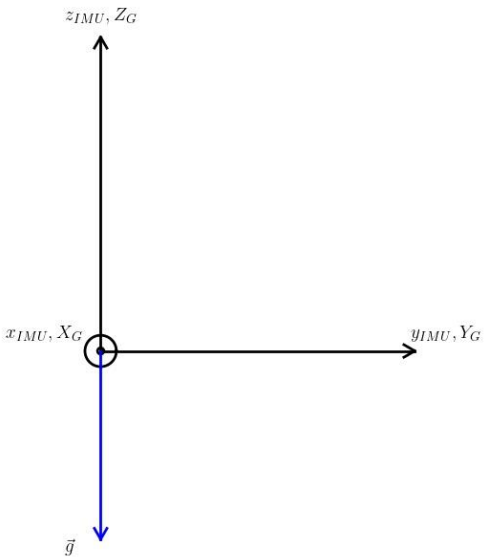
Tento uzel grafu se stará o získávání dat z modulu MPU6050 pomocí sériové sběrnice I2C, odhad orientace těla robota a následné zasílání tohoto odhadu orientace ve zprávě se jménem /notspot_imu/base_link_orientation.

Na obrázku 63 je vidět souřadný systém modulu MPU6050, ve kterém senzor vyjadřuje jednotlivé hodnoty zrychlení a úhlových rychlostí.



Obrázek 64 – Souřadný systém modulu MPU6050 (38)

V neutrální pozici je souřadný systém modulu shodný s imaginárním pevným souřadným systémem G, ve kterém budeme vyjadřovat úhly natočení modulu, a tedy těla robota. Při nulovém naklopení těla je tento imaginární souřadný systém shodný se systémem base_link_world.



Obrázek 65 – Neutrální orientace modulu MPU6050

Pokud na senzor nepůsobí žádné zrychlení způsobené pohybem těla, snímáme pouze zrychlení gravitační, které má složky g_x, g_y *a* g_z.

V pevném souřadném systému G je gravitační zrychlení vždy popsáno vektorem $\overrightarrow{g_G} = (0, 0, g_{Gz})^T$. Natočením těla nedochází k natočení souřadného systému G, a proto je vektor $\overrightarrow{g_G}$ konstantní. V lokálním souřadném systému inerciální měřící jednotky je vektor zrychlení popsán vektorem $\overrightarrow{g_{IMU}} = \left(g_{IMU_x}, g_{IMU_y}, g_{IMU_z}\right)^T$.

Orientaci senzoru budeme vyjadřovat jako rotaci kolem osy X_G o úhel α_{roll} následovanou rotací kolem osy Y_G o úhel β_{pitch}. Je nutné podotknout, že druhá rotace o úhel β_{pitch} probíhá kolem osy Y_G pevného souřadného systému G, a ne kolem nově vzniklé osy y. Tuto rotaci lze obecně popsat pomocí rovnice (2.7), kdy díky vektoru $\overrightarrow{P_B}$ a rotační matice $R_{X_G,\alpha_{roll}}$ vyjádříme vektor $\overrightarrow{P_G}$, a tento vektor následně využijeme znovu ve stejné rovnici jako vektor $\overrightarrow{P_B}$, který vynásobíme rotační maticí $R_{Y_G,\beta_{pitch}}$ a získáme tak výsledný vektor $\overrightarrow{P_G}$. Dostaneme souřadnice vektoru, který rotoval kolem dvou různých os pevného souřadného systému.

$$\overrightarrow{g_{G,\alpha_{roll}}} = R_{X_G,\alpha_{roll}} * \overrightarrow{g_{IMU}} \tag{6.1}$$

$$\overrightarrow{g_G} = R_{Y_G,\beta_{pitch}} * \overrightarrow{g_{G,\alpha_{roll}}} \tag{6.2}$$

$$\overrightarrow{g_G} = R_{Y_G,\beta_{pitch}} * R_{X_G,\alpha_{roll}} * \overrightarrow{g_{IMU}} \tag{6.3}$$

Matice $R_{Y_G,\beta_{pitch}} * R_{X_G,\alpha_{roll}}$ vyjadřuje orientaci systému po dvou rotacích kolem os X_G a Y_G pevného souřadného systému G. Z rovnice (6.3) vyjádříme $\overrightarrow{g_{IMU}}$.

$$\overrightarrow{g_{IMU}} = \left(R_{Y_G,\beta_{pitch}} * R_{X_G,\alpha_{roll}}\right)^T * \overrightarrow{g_G} \tag{6.4}$$

$$\overrightarrow{g_{IMU}} = \left(\begin{pmatrix} \cos\beta_{pitch} & 0 & \sin\beta_{pitch} \\ 0 & 1 & 0 \\ -\sin\beta_{pitch} & 0 & \cos\beta_{pitch} \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_{roll} & -\sin\alpha_{roll} \\ 0 & \sin\alpha_{roll} & \cos\alpha_{roll} \end{pmatrix}\right)^T * \begin{pmatrix} 0 \\ 0 \\ g_{Gz} \end{pmatrix} \tag{6.5}$$

$$\overrightarrow{g_{IMU}} = \begin{pmatrix} \cos\beta_{pitch} & \sin\alpha_{roll} * \sin\beta_{pitch} & \cos\alpha_{roll} * \sin\beta_{pitch} \\ 0 & \cos\alpha_{roll} & -\sin\alpha_{roll} \\ -\sin\beta_{pitch} & \sin\alpha_{roll} * \cos\beta_{pitch} & \cos\alpha_{roll} * \cos\beta_{pitch} \end{pmatrix}^T * \begin{pmatrix} 0 \\ 0 \\ g_{Gz} \end{pmatrix} \tag{6.6}$$

$$\overrightarrow{g_{IMU}} = \begin{pmatrix} -\sin\beta_{pitch} * g_{Gz} \\ \sin\alpha_{roll} * \cos\beta_{pitch} * g_{Gz} \\ \cos\alpha_{roll} * \cos\beta_{pitch} * g_{Gz} \end{pmatrix} \tag{6.7}$$

Známe-li úhly natočení α_{roll} a β_{pitch}, můžeme spočítat teoretický výstup z modulu MPU6050. Naším cílem je ale zjistit právě tyto úhly natočení, když známe vektor zrychlení

vyjádřený v souřadném systému modulu MPU6050. Toho lze docílit pomocí rovnic (6.8) a (6.9). (12) (13)

$$\alpha_{roll} = \text{atan}\left(\frac{g_{IMU_y}}{g_{IMU_z}}\right) \tag{6.8}$$

$$\beta_{pitch} = \text{atan}\left(\frac{-g_{IMU_x}}{\sqrt{g_{IMU_y}^2 + g_{IMU_z}^2}}\right) \tag{6.9}$$

Toto je jeden ze způsobů, jak zjistit natočení těla. Druhým způsobem je odhad orientace na základě úhlových rychlostí a krátkých časových úseků mezi jednotlivými měřeními rychlostí.

$$\alpha_{roll} = \alpha_{roll} + {}^{G}\omega_{roll} * \Delta t \tag{6.10}$$

$$\beta_{pitch} = \beta_{pitch} + {}^{G}\omega_{pitch} * \Delta t \tag{6.11}$$

V rovnicích (6.10) a (6.11) je použita úhlová rychlost vyjádřená v pevném souřadném systému G. To je důležité, jelikož veškeré úhly natočení těla stahujeme právě k tomuto souřadnému systému, a ne k souřadnému systému senzoru. Mezi úhlovými rychlostmi v těchto dvou souřadných systémech platí vztah, který je vyjádřen rovnicí (6.12).

$$\begin{pmatrix} {}^{G}\omega_{roll} \\ {}^{G}\omega_{pitch} \\ {}^{G}\omega_{yaw} \end{pmatrix} = R_{Y_G,\beta_{pitch}} * R_{X_G,\alpha_{roll}} * \begin{pmatrix} {}^{IMU}\omega_{roll} \\ {}^{IMU}\omega_{pitch} \\ {}^{IMU}\omega_{yaw} \end{pmatrix} \tag{6.12}$$

Z této rovnice lze vyjádřit hodnoty ${}^{G}\omega_{roll}$ a ${}^{G}\omega_{pitch}$, které následně můžeme dosadit do rovnic (6.10) a (6.11).

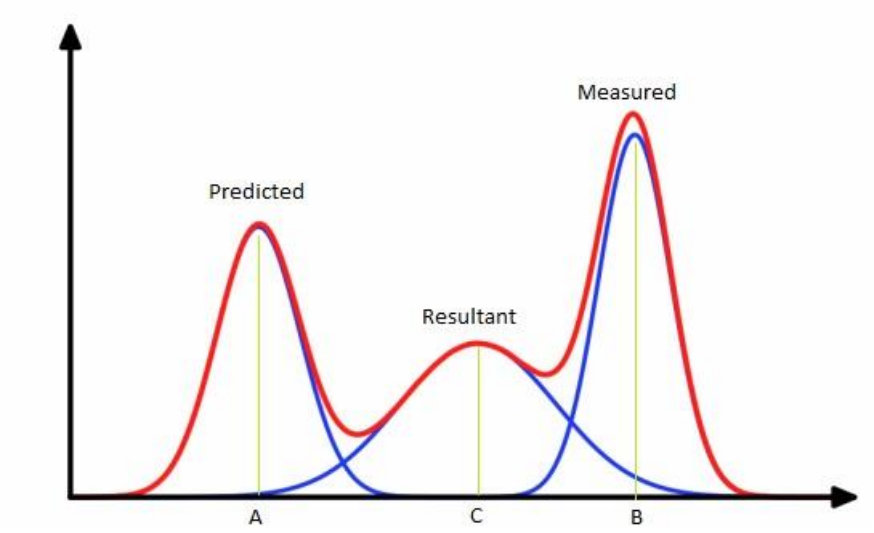
$${}^{G}\omega_{roll} = \cos\beta_{pitch} * {}^{IMU}\omega_{roll} + \sin\alpha_{roll} * \sin\beta_{pitch} * {}^{IMU}\omega_{pitch} + \cos\alpha_{roll} * \sin\beta_{pitch} * {}^{IMU}\omega_{yaw} \tag{6.13}$$

$${}^{G}\omega_{pitch} = \cos\alpha_{roll} * {}^{IMU}\omega_{pitch} - \sin\alpha_{roll} * {}^{IMU}\omega_{yaw} \tag{6.14}$$

Je-li tedy senzor otočen o 90 stupňů kolem osy Y_G, pak ${}^{IMU}\omega_{roll}$ nepřispívá nic k celkové úhlové rychlosti kolem osy X_G. Osa X_G splývá s osou Z_{IMU} a proto ${}^{IMU}\omega_{yaw}$ k úhlové rychlosti ${}^{G}\omega_{roll}$ přispívá.

U modulů, jako je MPU6050, se často setkáváme s nepříznivým šumem ve výstupních datech, v našem případě tedy v odhadu úhlů natočení těla robota. Z tohoto důvodu potřebujeme použít filtr, který by nežádoucí hodnoty odfiltroval a lépe tak aproximovat reálné natočení těla robota. V této odborné práci byl použit Kalmanův filtr, což je filtr kombinující teorii pravděpodobnosti a slučování informací z více zdrojů (pro nás tedy

gyroskop a akcelerometr). Hlavní myšlenkou je odhad stavu systému (natočení těla robota) v čase t, měření stavu systému v čase t a následná kombinace obojího.

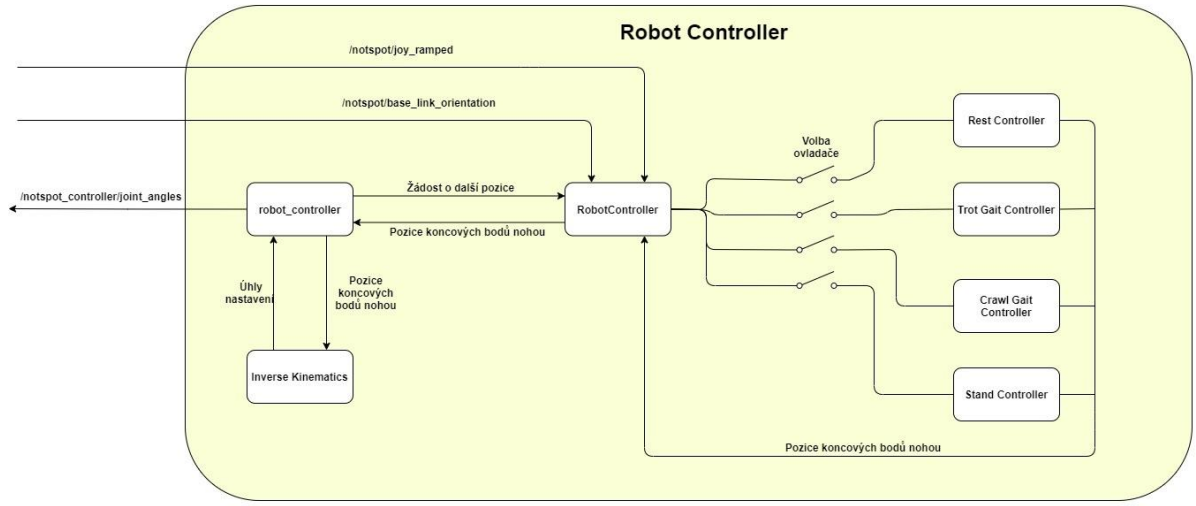


Obrázek 66 – Princip funkce Kalmanova filtru (39)

Díky tomuto filtru se podařilo částečně zbavit se šumu a dosáhnout tedy lepšího odhadu natočení těla v čase t. Více informací k teorii a použití Kalmanova filtru lze najít v (14).

6.2.4 Robot Controller

Uzel Robot Controller je program, který přijímá data z uzlů Joystick_ramped a IMU_kalman_filter. Na základě přijatých dat vypočítává úhly nastavení jednotlivých členů nohou robota ($\theta_{1,i}, \theta_{3,i}, \theta_{4,i}$; $i \in \{0,1,2,3\}$). Jedná se o nejkomplexnější program celého kontrolního systému robota a je rozdělen na několik podprogramů.



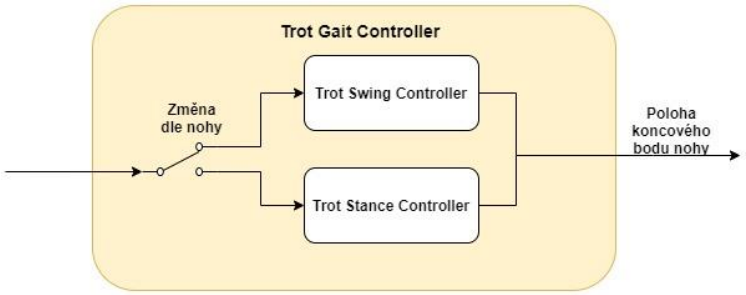
Obrázek 67 – Struktura uzlu Robot Controller

Hlavní myšlenkou bylo vytvořit sadu ovladačů, pomocí kterých by bylo možné generovat určité pohyby robota. Díky tomu, že výstupem každého ovladače jsou souřadnice koncových bodů nohou, které jsou vyjádřené v souřadném systému base_link_world (viz. kapitola 2.6), je velmi jednoduché vytvářet nové ovladače pro různé účely bez toho, aniž by byly ovlivněny ovladače ostatní, již vytvořené.

Nejjednodušším z použitých ovladačů je tzv. **Rest Controller**, což je ovladač, který použijeme v případě, kdy chceme, aby robot stál na místě a pouze prováděl lokální rotaci a posunutí těla. Výstupem tohoto ovladače jsou konstantní hodnoty souřadnic koncových bodů nohou, které definují neutrální polohu robota při stání. Na základě dat z PS4 ovladače dochází k modifikaci hodnot rotace a posunutí kolem os souřadného systému base_link_world, které jsou následně zohledněny při výpočtu inverzní kinematiky celého robota (viz. kapitola 2.6) (video).

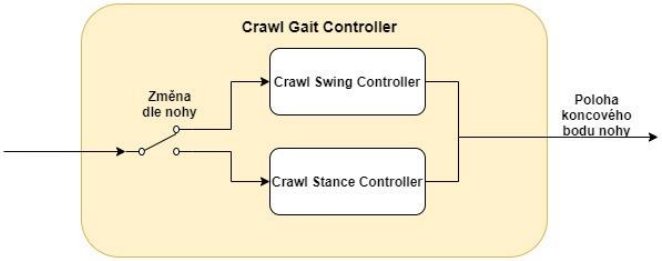
Dalším z použitých ovladačů je **Trot Gait Controller**. Jedná se o ovladač, který je použit pro generování takových pohybů nohou, abychom dosáhli dynamické chůze robota, tzn. v jednom okamžiku jsou ve vzduchu dvě nohy robota najednou. Sám tento ovladač obsahuje dva podovladače, které se jmenují Trot Swing Controller a Trot Stance Controller (15) (16). Trot Stance Controller je ovladač použitý pro generování pohybu

nohou, které se v daném okamžiku nacházejí na zemi a při chůzi se pohybují proti pohybu robota. Trot Swing Controller je naopak kontrolér použitý pro generování pohybů nohou, které se pohybují ve směru pohybu robota a zároveň provádí pohyb v ose Z souřadného systému base_link_world. Na základě toho, má-li se v daném okamžiku noha nacházet na zemi nebo ve vzduchu, je zvolen vhodný ovladač (video).

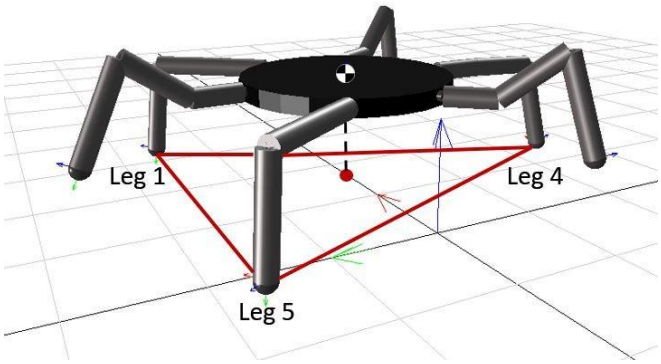


Obrázek 68 – Trot Gait Controller

Druhým z použitých ovladačů pro generování pohybů chůze je **Crawl Gait Controller**. Generovaný pohyb je staticky stabilní chůze, což znamená, že v jednom okamžiku je ve vzduchu pouze jedna noha a těžiště robota se nachází nad trojúhelníkem vzniklým spojením bodů dotyků nohou robota se zemí (viz obrázek 69). Stejně jako Trot Gait Controller se Crawl Gait Controller skládá ze dvou podovladačů, které se jmenují Crawl Swing Controller a Crawl Stance Controller (15) (16). Volba mezi nimi závisí na tom, má-li se v daném okamžiku daná noha nacházet na zemi, či ve vzduchu (video).

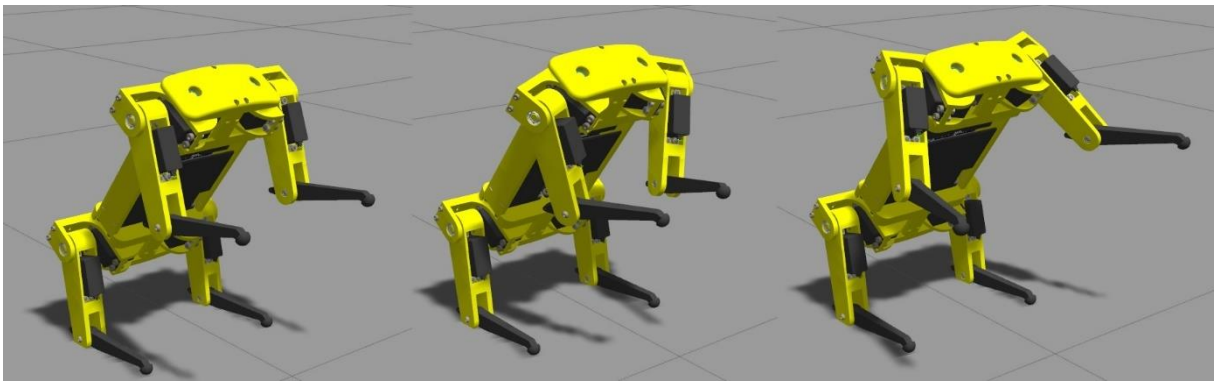


Obrázek 69 – Crawl Gait Controller



Obrázek 70 – Stav statické stability šestinohého robota (40)

Posledním ovladačem, který vznikl při práci na tomto projektu je **Stand Controller**. Úkolem tohoto ovladače je měnit pozice koncových bodů předních nohou a zachovat pozice koncových bodů nohou zadních. Toho se využije v případě, že robota pomocí ovladače Rest Controller postavíme na zadní nohy a přední nohy chceme ovládat na základě pokynů z PS4 kontroléru (viz. obrázek 70) (video).



Obrázek 71 – Robot stojící na zadních nohách

Důležitým programem uzlu Robot Controller je **RobotController**. Jedná se o program, který přijímá data o stavu tlačítek a páček PS4 ovladače a o odhadu natočení těla robota. Při generaci nových souřadnic koncových bodů nohou jsou tato data přeposlána právě zvolenému ovladači (Rest Controller, Trot Gait Controller, Crawl Gait Controller, Stand Controller), který tyto informace zohledňuje a na jejich základě generuje nové souřadnice koncových bodů nohou robota v souřadném systému base_link_world. Sám program RobotController čte data z PS4 ovladače a jsou-li zmáčknuta předem definovaná tlačítka, je provedeno zvolení patřičného ovladače robota.

Program **robot controller** se stará a zasílání žádostí o nové pozice koncových bodů programu RobotController. Při získání nových souřadnich koncových bodů jsou tato data předána programu **Inverse Kinematics**, který provádí výpočet inverzní kinematiky robota. Výsledkem jsou úhly natočení jednotlivých členů nohou, které jsou publikovány ve zprávě se jménem /notspot_controller/joint_angles. Zasílání žádostí o nové souřadnice nohou, výpočet inverzní kinematiky robota a publikace uhlů natočení je prováděno s frekvencí 30 Hz.

```python
#!/usr/bin/env python3
#Author: Jakub Jon

# Import knihovny ROS pro jazyk Python3
import rospy

# Import programu RobotController
from RobotController import RobotController

# Import programu Inverse Kinematics
from InverseKinematics import robot_IK

# Import potřebných zpráv pro komunikaci mezi programy
from sensor_msgs.msg import Joy
from notspot_msgs.msg import JointAngles
from geometry_msgs.msg import Vector3

USE_IMU = True
RATE = 30

# Vytvoření uzlu Robot_Controller
rospy.init_node("Robot_Controller")

# Robot geometry
body = [0.1908,0.08] # meters
legs = [0.0,0.04,0.1,0.094333] # meters

# Vytvoření instance objektu RobotController
notspot_robot = RobotController.Robot(body, legs, USE_IMU)

# Vytvoření instance objektu InverseKinematics
inverseKinematics = robot_IK.InverseKinematics(body, legs)

"""
Registrace pro zasílání zpráv, které obsahují úhly natočení členů nohou robota.
Jméno zprávy: notspot_controller/joint_angles
Datová struktura: JointAngles (obsahuje výsledek výpočtu inverzní kinematiky robota, tzn. 12 úhlů)
"""
jp = rospy.Publisher("notspot_controller/joint_angles",JointAngles,queue_size=1)

if USE_IMU:
    """
    Registrace pro přijímání zpráv, které ubsahují informace o natočení těla robota.
    Jméno zprávy: notspot_imu/base_link_orientation
    Datová struktura: Vector3 (úhly natočení těla robota = roll, pitch a yaw)
    Volaná funkce při přijetí nové zprávy: notspot_robot.imu_orientation => uložení dat do paměti
    """
    rospy.Subscriber("notspot_imu/base_link_orientation",Vector3,notspot_robot.imu_orientation)

"""
Registrace pro přijímání zpráv, které ubsahují informace o stavu páček a tlačítek PS4 ovladače.
Jméno zprávy: notspot_joy/joy_ramped
Datová struktura: Joy (stavy páček PS4 ovladače)
Volaná funkce při přijetí nové zprávy: notspot_robot.joystick_command => zpracování dat z PS4 ovladače
"""
rospy.Subscriber("notspot_joy/joy_ramped",Joy,notspot_robot.joystick_command)

# Frekvence, se kterou bude smyčka programu běžet
rate = rospy.Rate(RATE)

del body
del legs
del USE_IMU
del RATE
```

```
66  # Hlavní smyčka programu
67  while not rospy.is_shutdown():
68
69      # Získání pozic koncových bodů nohou robota
70      leg_positions = notspot_robot.run()
71
72      # Změna používaného ovladače robota, pokud dojde ke zmáčknutí určených tlačítek na PS4 kontroléru.
73      notspot_robot.change_controller()
74
75      # Získání lokálního posunutí systému base_link v systému base_link_world (posunutí těla robota)
76      dx = notspot_robot.state.body_local_position[0]
77      dy = notspot_robot.state.body_local_position[1]
78      dz = notspot_robot.state.body_local_position[2]
79
80      # Získání lokální orientace systému base_link v systému base_link_world (natočení těla robota)
81      roll = notspot_robot.state.body_local_orientation[0]
82      pitch = notspot_robot.state.body_local_orientation[1]
83      yaw = notspot_robot.state.body_local_orientation[2]
84
85      try:
86          # Výpočet inverzní kinematiky celého robota a získání úhlů natočení členů nohou
87          joint_angles = inverseKinematics.inverse_kinematics(leg_positions,
88                                                              dx, dy, dz, roll, pitch, yaw)
89
90          # Zaslání zpráv s daty úhlů natočení
91          jp.publish(joint_angles)
92      except:
93          pass
94
95      # Zastavení smyčky tak, aby běžela s definovanou frekvencí
96      rate.sleep()
```

Obrázek 72 – Program robot_controller.py

6.2.5 Hardware Interface

Hardware Interface je uzel, který přímo komunikuje s modulem PCA9685. Přijímá zprávy se jménem /notspot_controller/joint_angles, které obsahují hodnoty úhlů natočení členů nohou robota. Tyto úhly jsou přepočítávány na úhly servomotorů, následuje generace PWM signálů, které jsou posléze posílány modulu PCA9685, aby došlo k nastavení požadovaných úhlů na servomotorech (video). Jako jediný program ze všech je uzel Hardware Interface napsán v jazyce C++, jelikož bylo potřeba zajistit rychlou a spolehlivou komunikaci počítače s module PCA9685. Ostatní programy a ovladače jsou napsány v jazyce Python.

```cpp
// hardware_interface.cpp
// Author: Jakub Jon

// Import použitých "header files"
#include <ros/ros.h>
#include <iostream>
#include <unistd.h>
#include <PiPCA9685/PCA9685.h>  // Credit: https://github.com/barulicm/PiPCA9685
#include <cmath>
#include <notspot_msgs/JointAngles.h>


// Deklarace použitých funkcí
int map_angle(float servo_min, float servo_max, float angle_in);
float servo_3_convert(float angle_leg);
void callback(const notspot_msgs::JointAngles::ConstPtr& msg);

// instance PCA9685 pro komunikaci s modulem
PCA9685 pwm{};

int main(int argc, char* argv[])
{
    // nastavení pwm frekvence
    pwm.set_pwm_freq(50.0);

    // Vytvoření uzlu Hardware Interface
    ros::init(argc, argv, "Hardware_interface");

    /*
    Registrace pro přijímání zpráv s informacemi o úhlen natočení členů nohou.
    Jméno zprávy: notspot_controller/joint_angles
    Datová struktura: JointAngles
    Volaná funkce při přijetí nové zprávy: callback => zpracování dat a zaslání pwm signálů
    */
    ros::NodeHandle n;
    ros::Subscriber sub = n.subscribe("notspot_controller/joint_angles", 1, callback);

    // "zastavení" funkce main
    ros::spin();
    return 0;
}
int map_angle(float servo_min, float servo_max, float angle_in)
{
    // Funkce k mapování pwm signálů k úhlům
    int temp = angle_in * (servo_max - servo_min) / 180. + servo_min;
    return(temp);
}

float servo_3_convert(float angle_leg)
{
    // Přepočet úhlů natočení dolního členu nohy na natočení páky servomotoru
    return 0.000176158337726 * pow(angle_leg,3) - 0.005724009421754 * pow(angle_leg,2) + 1.345285133192364 * angle_leg - 0.533033148447196;
}

double degrees(float angle_in)
{
    // Převod úhlu v radiánech na úhel ve stupních
    return angle_in * 180.0 / 3.14159265358;
}
```

```
62  void callback(const notspot_msgs::JointAngles::ConstPtr& msg)
63  {
64      // FR1
65      pwm.set_pwm(0,0,map_angle(110,520,94 - degrees(msg->joint_angles[0])));
66
67      // FR2
68      pwm.set_pwm(1,0,map_angle(110,520,118 - degrees(msg->joint_angles[1])));
69
70      // FR3
71      pwm.set_pwm(2,0,map_angle(110,520,101 + servo_3_convert(degrees(msg->joint_angles[2]) + 90)));
72
73      // FL1
74      pwm.set_pwm(4,0,map_angle(110,520,86 - degrees(msg->joint_angles[3])));
75
76      // FL2
77      pwm.set_pwm(5,0,map_angle(110,520,71 + degrees(msg->joint_angles[4])));
78
79      // FL3
80      pwm.set_pwm(6,0,map_angle(110,520,91 - servo_3_convert(degrees(msg->joint_angles[5]) + 90)));
81
82      // RR1
83      pwm.set_pwm(8,0,map_angle(110,520,93 + degrees(msg->joint_angles[6])));
84
85      // RR2
86      pwm.set_pwm(9,0,map_angle(110,520,118 - degrees(msg->joint_angles[7])));
87
88      // RR3
89      pwm.set_pwm(10,0,map_angle(110,520,93 + servo_3_convert(degrees(msg->joint_angles[8]) + 90)));
90
91      // RL1
92      pwm.set_pwm(12,0,map_angle(110,520,95 + degrees(msg->joint_angles[9])));
93
94      // RL2
95      pwm.set_pwm(13,0,map_angle(110,520,80 + degrees(msg->joint_angles[10])));
96
97      // RL3
98      pwm.set_pwm(14,0,map_angle(110,520,86 - servo_3_convert(degrees(msg->joint_angles[11]) + 90)));
99
100     ros::spinOnce();
101 }
```

Obrázek 73 – Program hardware_interface.cpp

7 Simulace robota

Při tvorbě a vývoji robotických systému se vědci a inženýři snaží využívat simulaci pro generování potřebných dat k vylepšení dosavadních systémů. Simulace robotů je ideální možnost pro debuggování programů a zjišťování, zda se v nově naprogramovaných ovladačích neobjevuje chyba. Zároveň simulace umožňují dostat systém do stavů, ve kterých si nejsme jisti, jak bude kontrolér reagovat, a pomocí toho zjistit vlastnosti použitých ovladačů. I když je simulace dobrým nástrojem pro vylepšování softwaru robotů, není možné spoléhat se pouze na výsledky dosažené v simulovaném prostředí a je tedy potřeba software testovat i na reálných robotech. Ideální je využívat výhody obou možností.

Tato odborná práce nebyla výjimkou, proto část vývoje softwaru probíhala právě v simulaci. Jako simulační program byl zvolen program Gazebo, jelikož se jedná o projekt vyvíjený firmou Open Source Robotics Foundation, firmou, která současně vyvíjí systém ROS. Díky tomu, že jsou oba zmíněné projekty vyvíjeny stejnou firmou, je velmi jednoduché v programu Gazebo simulovat robota, který využívá ROS.

Obrázek 74 – Logo simulátoru Gazebo (41)

7.1 Gazebo

V programu Gazebo lze simulovat nejrůznější druhy robotů, od těch mobilních až po stacionární. Každý robot, kterého chceme simulovat, musí být popsán pomocí jednoho z několika normalizovaných formátů. Mezi nejznámější patří formáty URDF (Unified Robot Description Format) a SDF (Simulation Description Format). Tyto formáty obsahují informace o hmotnostech členů robota, pozicích těžišť vzhledem k určitým souřadným systémům, momentech setrvačností a mnoho dalších dat potřebných pro správnou simulaci dynamiky robota. Jedná se o soubor, který musí být pečlivě a bezchybně vytvořen, jinak v simulaci nedosáhneme reálných výsledků.

V této odborné práci byl použit formát URDF, který lze vygenerovat z modelu robota v programu SolidWorks. Pro export formátu URDF byl použit rozšiřující program SolidWorks to URDF exporter (17).

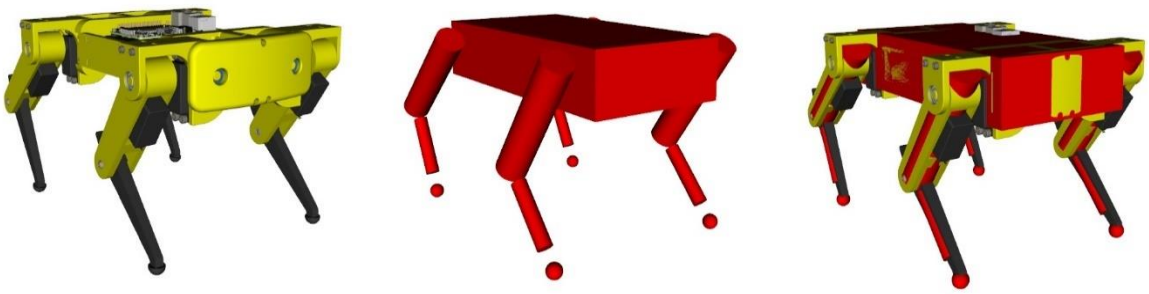
Při exportu z CAD programu je nutné nastavit materiály jednotlivých dílů. Podle nastavených materiálů totiž dochází k výpočtu hmotností jednotlivých součástí a jejich momentů setrvačností.

```
<link
  name="FR2">
  <inertial>
    <origin
      xyz="0.038008 0.0024973 4.1594E-05"
      rpy="0 0 0" />
    <mass
      value="0.080008" />
    <inertia
      ixx="7.4759E-05"
      ixy="-5.6852E-07"
      ixz="7.3034E-09"
      iyy="1.1848E-05"
      iyz="-4.6245E-07"
      izz="7.6656E-05" />
  </inertial>
  <visual>
    <origin
      xyz="0 0 0"
      rpy="0 0 0" />
    <geometry>
      <mesh
        filename="package://notspot_description/meshes/obj/legR.obj" />
    </geometry>
  </visual>
  <collision>
    <origin
      xyz="0.04 0 0"
      rpy="0 1.57075 0" />
    <geometry>
      <cylinder length="0.125" radius="0.014"/>
    </geometry>
  </collision>
</link>
```

```
<joint
  name="FR2_joint"
  type="revolute">
  <origin
    xyz="0 -0.04615 0"
    rpy="1.5708 0 0" />
  <parent
    link="FR1" />
  <child
    link="FR2" />
  <axis
    xyz="0 0 1" />
  <limit
    lower="-3"
    upper="3"
    effort="${effort}"
    velocity="3" />
</joint>
```

Obrázek – 75 Příklad formátu URDF

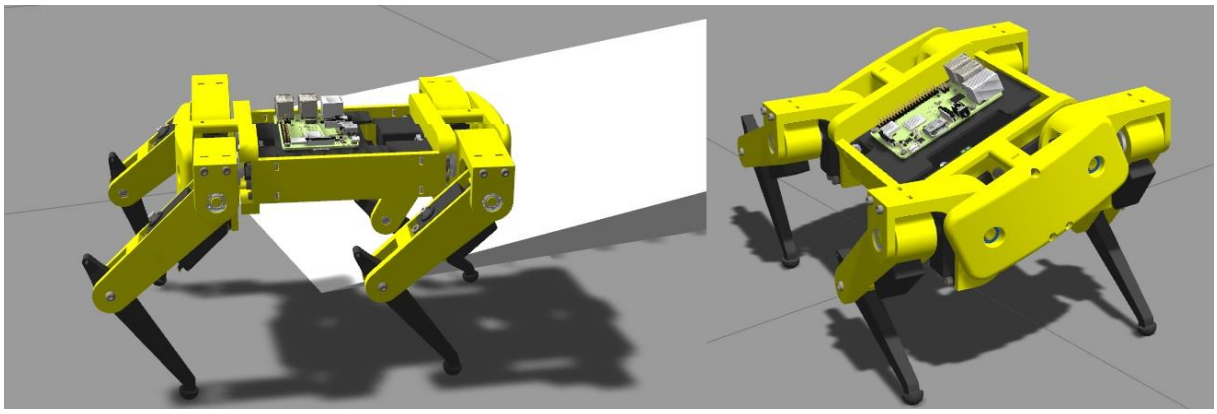
Součástí formátu URDF jsou adresy modelů pro vizualizaci robota a také modely pro výpočet kolizí robota s prostředím. Nejčastěji se pro vizualizaci používají vyexportované modely dílu, pro kolize jsou použity jednoduché tvary (válce, obdélníky, …), pomocí kterých může simulátor jednodušeji detekovat kolize s prostředím. Díky tomu může simulace běžet rychleji (často v reálném čase, tzn. jedna sekunda v simulaci odpovídá jedné sekundě v realitě). Používají se tvary, které nejvíce odpovídají tvaru reprezentovaného členu.



Obrázek 76 – Porovnání modelů pro vizualizaci a modelů pro kolize

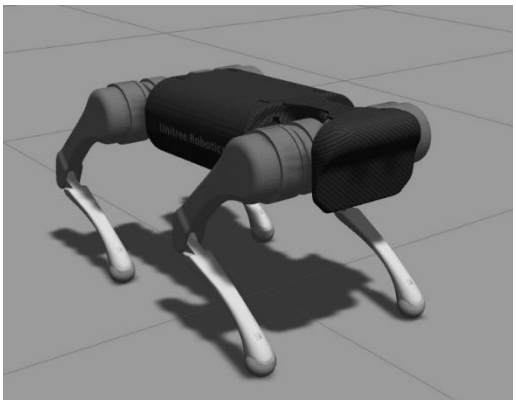
Vizuální modely dílů robota byly vyexportovány z programu SolidWorks ve formátu obj. Pro export byl použit rozšiřující program Free-Solidworks-OBJ-Exporter (18).

Po vygenerování všech potřebných souborů lze robota simulovat (video).

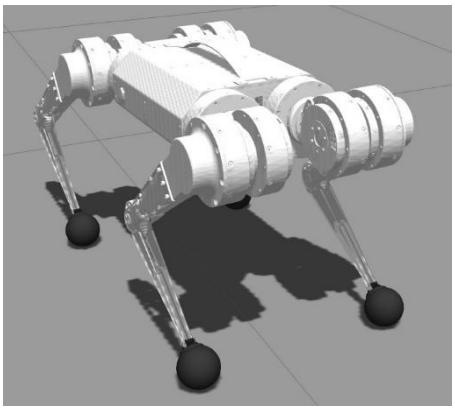


Obrázek 77 – Čtyřnohý robot v simulaci

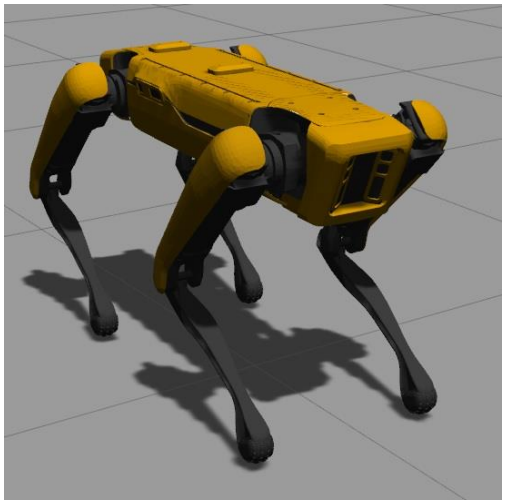
Díky open-source komunitám si lze na internetu najít a stáhnout modely a soubory URDF dalších čtyřnohých robotů.



Obrázek 78 – Robot A1 firmy Unitree Robotics v simulaci (42)

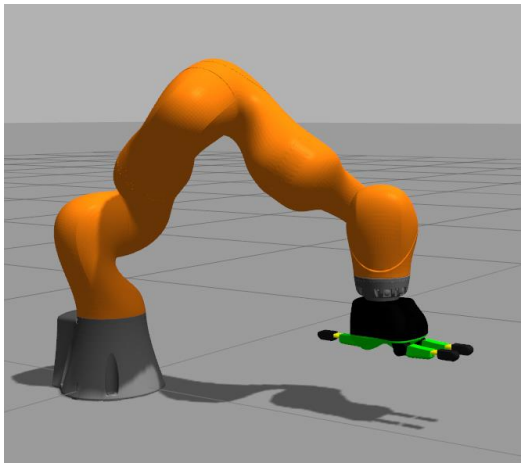


Obrázek 79 – Robot Mini Cheetah univerzity MIT v simulaci (42)

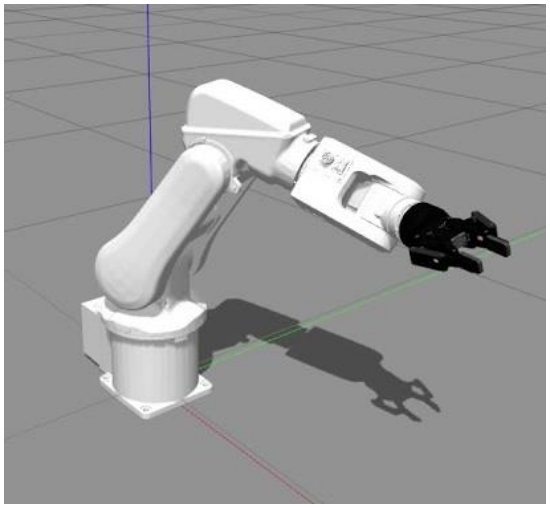


Obrázek 80 – Robot Spot firmy Boston Dynamics v simulaci (42)

V programu Gazebo lze simulovat nejen čtyřnohé roboty, ale i robotická ramena, která jsou často používána v průmyslu.



Obrázek – 81 Robot KUKA LBR iiwa 7 (43)



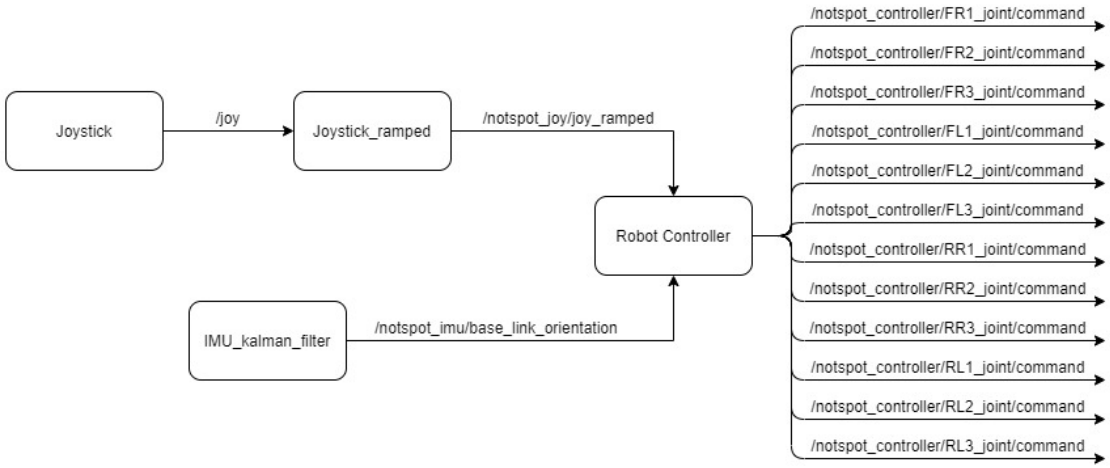
Obrázek – 82 Robot ABB IRB 120 (44)

Díky své univerzálnosti a možnosti simulovat jakýkoli typ robotů je Gazebo velmi populárním simulátorem v komunitě ROS. Jedná se o software, který lze využít ve výuce robotiky, pro simulaci řídících programů robotů nebo dokonce i pro trénování algoritmů strojového učení.

7.2 Simulace softwaru robota

V programu Gazebo lze jednoduše simulovat řídící program představený v kapitole 6.2. Pro každý člen, se kterým chceme pohybovat vytvoříme Gazebo-kontrolér, který na základě odchylky od požadované pozice aplikuje kroutící moment. Velikost kroutícího momentu je určena PID kontrolérem.

Požadovaný úhel natočení členu se nastaví zasláním zprávy se jménem /notspot_controller/*/command, která obsahuje námi zvolenou pozici v radiánech. Místo hvězdičky * je jméno členu, se kterým chceme pohybovat. Struktura použitého programu pro řízení robota v simulaci je na obrázku 80.

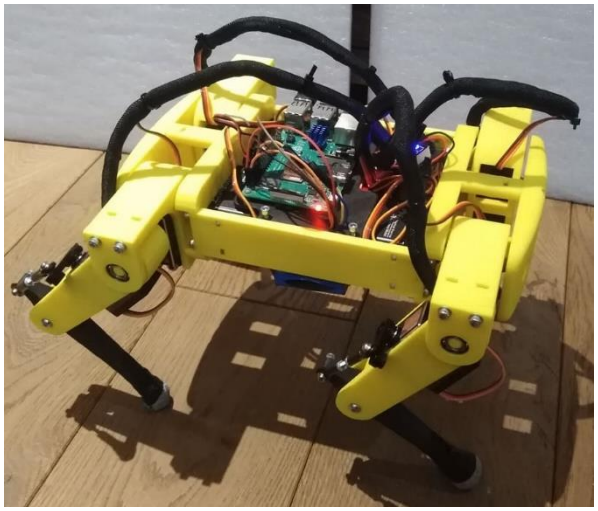


Obrázek 83 – Struktura programu v ROS pro simulaci robota

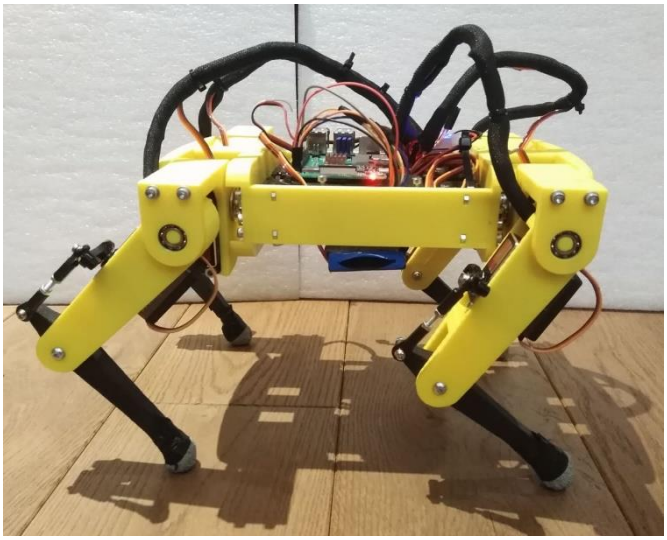
Jak je na obrázku vidět, simulovaný robot používá stejný řídící systém jako reálný robot. Došlo pouze k odstranění uzlu Hardware Interface a ke změně jmen zpráv, které obsahují úhly jednotlivých členů nohou robota, tak, aby vyhovovaly požadavkům simulátoru Gazebo. Drtivá většina softwaru byla navržena v simulaci, poslední úpravy řídícího systému proběhly na reálném robotu. Díky simulaci bylo možné otestovat různé změny softwaru bez nebezpečí vzniku škod na hardwaru reálného robota.

Závěr

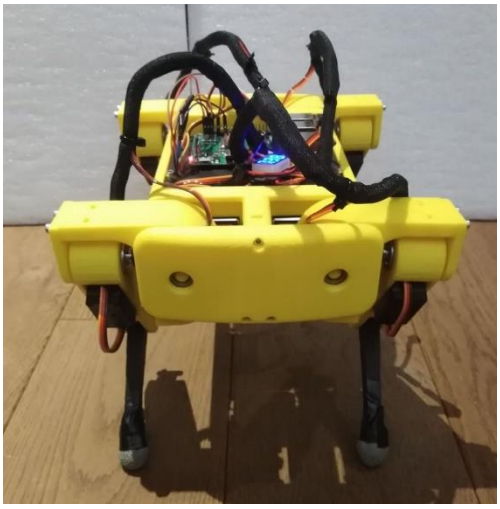
Výsledkem této práce je čtyřnohý robot, kterého lze na naší škole využít pro výuku mechatroniky, robotiky a programování. I když se nejedná o zařízení, které by se dalo využít v průmyslu, je tento robot ideální pro demonstraci základů dopředné a inverzní kinematiky robotů, DH-parametrů a obecně matematických metod pro popis vzájemných poloh a natočení dvou a více souřadných systémů.



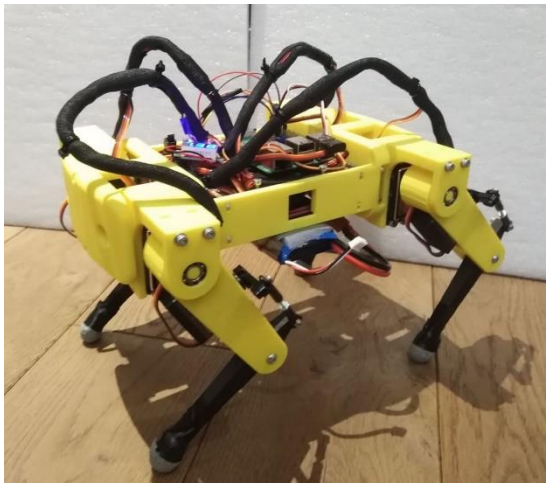
Obrázek 84 – Pohled z pravé strany shora na robota



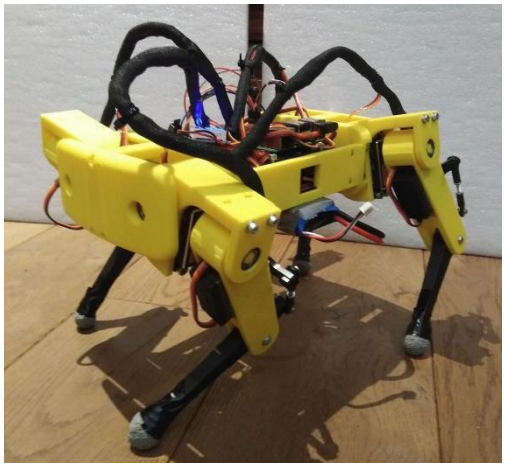
Obrázek 85 – Pohled z pravé strany na robota



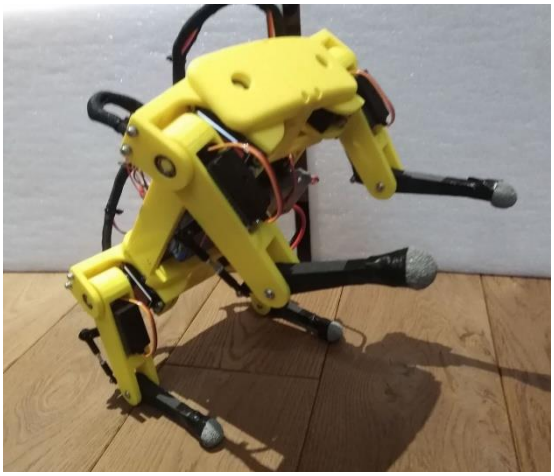
Obrázek 86 – Pohled zepředu na robota



Obrázek 87 – Pohled zleva na robota



Obrázek 88 – Pohled zleva na natočeného robota



Obrázek 89 – Robot stojící na zadních nohách

Během vývoje a stavby robota vzniklo mnoho problému, které se museli řešit. Většinu z nich se vyřešit podařilo, jeden z nich, kterého jsme si všimli již na začátku stavby robota, stále přetrvává.

Tento problém spočívá v kmitání servomotorů. Myslíme si, že kmitání je způsobeno nevhodným kontrolérem uvnitř servomotorů, který se stará o snižování rozdílu mezi aktuální hodnotou natočení hřídele servomotoru a hodnotou požadovanou. Při malé výchylce se kontrolér snaží motor ovládat tak, aby se hřídel pootočila do požadované polohy. Bohužel dojde k přejetí požadované pozice a hřídel se vychýlí na stranu druhou. V tomto okamžiku nastane stejný problém jako na začátku, hřídel servomotoru proto začne oscilovat.

Všechny cíle, které byly stanoveny na počátku práce byly splněny, během vývoje ale začali vznikat nápady, jak robota ještě vylepšit. Zde je seznam těchto nápadů:

1) Normalizované délky šroubů: Při návrhu robota v CAD programu SolidWorks byly použity šrouby s nenormalozovanými délkami, což způsobilo, že nakoupené šrouby se často museli krátit. Návrhem je použití takových délek šroubů, které jsou volně dostupné na trhu. Tato změna by výrázně snížila časovou náročnost sestavování robota.

2) Kryt: Bylo by vhodné vymodelovat a na 3D tiskárně vytisknout kryt na tělo robota, který by zakryl elektroniku a kabely. Zároveň by se jednalo a novou plochu, na kterou by se dali přidělat různé senzory nebo dokonce i robotická ruka postavené z několika servomotorů.

Tato odborná práce mi umožnila naučit se mnoho o robotice jako takové. I když postavený robot nemá žádné praktické využití, práce na něm mi dala mnoho zkušeností. Díky ní jsem se naučil vyhledávat potřebné informace a řešit nastalé problémy. Veškeré neúspěchy i úspěchy při stavbě a vývoji robota mě přesvědčily o tom, že robotika je zajímavý obor, kterým bych se chtěl v budoucnosti zabývat.

Seznam obrázků

Seznam tabulek

Použitá literatura

1. **Boston Dynamics.** *Home | Boston Dynamics.* [Online] https://www.bostondynamics.com/.

2. **MIT.** *MIT Biomimetic Robotics Lab.* [Online] https://biomimetics.mit.edu/.

3. **Eidgenössische Technische Hochschule Zürich.** *Homepage – Robotic Systems Lab | ETH Zurich.* [Online] https://rsl.ethz.ch/.

4. **Katz, Benjamin G.** *A Low Cost Modular Actuator for Dynamic Robots.* [Master's Thesis] Cambridge, USA : Massachusetts Institute of Technology, 2018.

5. **Jazar, Reza N.** *Theory of Applied Robotics: Kinematics, Dynamics, and Control (2nd Edition).* New York : Springer US , 2010. 978-1-4419-1749-2.

6. **Wilk, Florian.** Basic simulation by user Florian Wilk/Kinematics · master · Custom Robots / SpotMicro - Boston Dynamics Spot inspired robot / Simulation · GitLab. *DevOps Platform Delivered as a Single Application | GitLab.* [Online] [Citace: 18. 2 2021.] https://gitlab.com/custom_robots/spotmicroai/simulation/-/tree/master/Basic%20simulation%20by%20user%20Florian%20Wilk/Kinematics.

7. **Tedrake, Russ.** Robot Manipulation: Perception, Planning, and Control (Course Notes for MIT 6.881). [Online] [Citace: 18. 2 2021.] http://manipulation.csail.mit.edu/.

8. **Strang, Gilbert.** *Introduction to Linear Algebra: Fifth Edition.* Wellesley : Wellesley-Cambridge Press, 2016. 978-0-9802327-7-6.

9. **RASPBERRY PI FOUNDATION.** *The Raspberry Pi Foundation.* [Online] [Citace: 26. 2 2021.] https://www.raspberrypi.org/.

10. **Open Source Robotics Foundation.** *Open Robotics.* [Online] [Citace: 26. 2 2021.] https://www.openrobotics.org/.

11. **Clearpath Robotics.** Intro to ROS — ROS Tutorials 0.5.2 documentation. *Clearpath Robotics: Mobile Robots for Research & Development.* [Online] [Citace: 17. 2 2021.] http://www.clearpathrobotics.com/assets/guides/kinetic/ros/Intro%20to%20the%20Robot%20Operating%20System.html.

12. **Wetzstein, Gordon.** EE 267 Virtual Reality, Course Notes: 3-DOF Orientation Tracking with IMUs. *Stanford University.* [Online] [Citace: 18. 2 2021.] https://stanford.edu/class/ee267/notes/ee267_notes_imu.pdf.

13. **Salmony, Philip.** IMU Attitude Estimation. *Philip Salmony.* [Online] [Citace: 18. 2 2021.] http://philsal.co.uk/projects/imu-attitude-estimation.

14. **Sebastian Thrun, Wolfram Burgard, Dieter Fox.** *Probabilistic Robotics.* Cambridge, Massachusetts : The MIT Press, 2006. 978-0-262-20162-9.

15. **mike4192.** GitHub - mike4192/spotMicro: Spot Micro Quadruped Project. *GitHub: Where the world builds software · GitHub.* [Online] [Citace: 18. 2 2021.] https://github.com/mike4192/spotMicro.

16. **Stanford Student Robotics.** GitHub - stanfordroboticsclub/StanfordQuadruped. *GitHub: Where the world builds software · GitHub.* [Online] [Citace: 18. 2 2021.] https://github.com/stanfordroboticsclub/StanfordQuadruped.

17. **Open Source Robotics Foundation.** sw_urdf_exporter - ROS Wiki. *Documentation - ROS Wiki.* [Online] [Citace: 18. 2 2021.] http://wiki.ros.org/sw_urdf_exporter.

18. **Larsen, Neil.** Free Solidworks OBJ Exporter v2.0 | SOLIDWORKS Forums. *Welcome | SOLIDWORKS Forums.* [Online] [Citace: 18. 2 2021.] https://forum.solidworks.com/thread/54270.

19. **IEEE Spectrum.** Mini Cheetah - ROBOTS: Your Guide to the World of Robotics. *ROBOTS: Your Guide to the World of Robotics.* [Online] [Citace: 17. 2 2021.] https://robots.ieee.org/robots/minicheetah/.

20. **Statt, Nick.** Boston Dynamics' Spot isn't quite the terrifying robot hunter you think it is - The Verge. *The Verge.* [Online] 19. 2 2020. [Citace: 17. 2 2021.] https://www.theverge.com/2020/2/19/21144648/boston-dynamics-spot-robot-mass-state-police-trial-issues.

21. **Tek Deeps.** See how Unitree A1, the newest robo-dog straight from China, dances [FILM] – NeeWS. *Latest Breaking News & Headlines all the World | Tek Deeps.* [Online] [Citace: 17. 2 2021.] https://tekdeeps.com/see-how-unitree-a1-the-newest-robo-dog-straight-from-china-dances-film-neews/.

22. **Atwell, Cabe.** Champ, the Open Source Quadrupedal Robot with Autonomous Navigation. *Hackster.io - The community dedicated to learning hardware.* [Online] [Citace: 17. 2 2021.] https://www.hackster.io/news/champ-the-open-source-quadrupedal-robot-with-autonomous-navigation-9781d4991e34.

23. **Stanford Student Robotics.** Stanford Pupper. *Stanford Student Robotics.* [Online] [Citace: 17. 2 2021.] https://stanfordstudentrobotics.org/pupper.

24. **SpotMicroAI.** Home - SpotMicroAI. *SpotMicroAI.* [Online] [Citace: 17. 2 2021.] https://spotmicroai.readthedocs.io/en/latest/.

25. **Lucky's Notes.** Euler Angles; Lucky's Notes. *Lucky's Notes; Notes on math, coding, and other stuff.* [Online] [Citace: 17. 2 2021.] https://luckytoilet.wordpress.com/tag/euler-angles/.

26. **Coretech Robotics.** Coretech Robotics: A simple Quadruped Robot. *Coretech Robotics.* [Online] [Citace: 17. 2 2021.] http://coretechrobotics.blogspot.com/2014/10/a-simple-quadruped-robot.html.

27. **Contributors, Wikipedia.** Atan2. *Wikipedia, the free encyclopedia.* [Online] [Citace: 17. 2 2021.] https://en.wikipedia.org/w/index.php?title=Atan2&oldid=994208318.

28. **MathIsFun.com.** Inverse Sine, Cosine, Tangent. *MathIsFun.com.* [Online] [Citace: 17. 2 2021.] https://www.mathsisfun.com/algebra/trig-inverse-sin-cos-tan.html.

29. **Triendl, Martin.** DIY quadruped robot. [Online] [Citace: 17. 2 2021.] https://www.youtube.com/watch?v=LDWCn3uT7jo&ab_channel=MartinTriendl.

30. **Elektor.** Raspberry Pi 3 B+ - Elektor. *Elektor Website.* [Online] [Citace: 17. 2 2021.] https://www.elektor.com/raspberry-pi-3-model-b-plus.

31. **Amazon.com.** Amazon.com: SunFounder MPU6050 Module for Arduino and Raspberry Pi, 3-axis Gyroscope and 3-axis Accelerator: Electronics. *Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more.* [Online] [Citace: 17. 2 2021.] https://www.amazon.com/SunFounder-MPU6050-Raspberry-Gyroscope-Accelerator/dp/B0151GI5VI/ref=sr_1_6?dchild=1&keywords=MPU6050&qid=1613574756&sr=8-6.

32. **Makerlab Electronics.** 16-Channel 12-bit PWM Servo Driver - PCA9685 Philippines. *Makerlab Electronics - Distributor Electronic Product and Equipments.* [Online] [Citace: 17. 2 2021.] https://www.makerlab-electronics.com/product/16-channel-12bit-pwm-servo-driver-pca9685/.

33. **Big hobby.cz.** BEC 5A FLYCOLOR ☆| RC BigHobby.cz. *BigHobby.cz.* [Online] [Citace: 17. 2 2021.] https://www.bighobby.cz/bec-5a--flycolor/.

34. **BigHobby.cz.** Power X6 3300 mAh 2S 35C (70C) ☆| RC BigHobby.cz. *BigHobby.cz.* [Online] [Citace: 17. 2 2021.] https://www.bighobby.cz/power-x6-3300-mah-2s-35c--70c/.

35. **HobbyKing.** JX CLS6336HV High Voltage Coreless Metal Gear High Torque Servo 35.6kg/0.11sec/63g. *Radio Control Planes, Drones, Cars, FPV, Quadcopters and more - Hobbyking.* [Online] [Citace: 17. 2 2021.] view-source:https://hobbyking.com/en_us/jx-cls6336hv-high-voltage-coreless-metal-gear-high-torque-servo-35-6kg-0-11sec-63g.html?___store=en_us.

36. **Wikimedia Commons.** File:Ros logo.svg --- Wikimedia Commons{,} the free media repository. *Wikimedia Commons.* [Online] [Citace: 17. 2 2021.] https://commons.wikimedia.org/w/index.php?title=File:Ros_logo.svg&oldid=504808180.

37. **Amazon.com.** DualShock 4 Wireless Controller for PlayStation 4 - Jet Black. *Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more.* [Online] [Citace: 17. 2 2021.] https://www.amazon.com/DualShock-Wireless-Controller-PlayStation-Black-4/dp/B01LWVX2RG.

38. **ProtoSupplies.com.** MPU-6050 GY-521 3-Axis Accel & Gryo Sensor Module - ProtoSupplies. *ProtoSupplies.* [Online] [Citace: 17. 2 2021.] https://protosupplies.com/product/mpu-6050-gy-521-3-axis-accel-gryo-sensor-module/.

39. **Saxena, Ujjwal.** Kalman Filter and Data Fusion. Before seeing how Kalman works, let's… | by Ujjwal Saxena | Medium. *Medium – Where good ideas find you.* [Online] [Citace: 17. 2 2021.] https://medium.com/@er.ujjwalsaxena/kalman-filter-and-data-fusion-f33ea813f4b6.

40. **Gurel, Canberk Suat.** 3. Fundamentals of Hexapod Robot | Details | Hackaday.io. *Hackaday.io | The world's largest collaborative hardware development community.* [Online] [Citace: 17. 2 2021.] https://hackaday.io/project/21904-hexapod-modelling-path-planning-and-control/log/62326-3-fundamentals-of-hexapod-robot.

41. **Open Source Robotics Foundation.** Gazebo : Media. *Gazebo.* [Online] [Citace: 17. 2 2021.] http://gazebosim.org/media.

42. **grassjelly, eborghi10, tarasborov.** GitHub - chvmp/robots: Collection of quadrupedal robots configured to work in CHAMP development framework. *GitHub: Where the world builds software · GitHub.* [Online] [Citace: 17. 2 2021.] https://github.com/chvmp/robots.

43. **Daestan.** Collision sometime with the ground · Issue #1561 · ros-planning/moveit · GitHub. *GitHub: Where the world builds software · GitHub.* [Online] [Citace: 17. 2 2021.] https://github.com/ros-planning/moveit/issues/1561.

44. **JdeRobot.** IndustrialRobotics/industrial_robots/irb120_robotiq85 at master · JdeRobot/IndustrialRobotics · GitHub. *GitHub: Where the world builds software · GitHub.*

[Online] [Citace: 17. 2 2021.] https://github.com/JdeRobot/IndustrialRobotics/tree/master/industrial_robots/irb120_robotiq85.

45. **učitelé SPŠSE.** Úvod. *SPŠSE a VOŠ Liberec.* [Online] 19. 2 2020. [Citace: 19. 2 2020.] https://www.pslib.cz.

A. Obsah přiloženého USB

- Písemná verze odborné práce v PDF
- Písemná verze odborné práce v DOCX
- 3D sestava a modely dílů robota ve formátech SLDPRT a SLDASM
- Schéma zapojení elektroniky v JPG
- Balíček ROS softwaru pro reálného robota
- Balíček ROS softwaru pro simulaci robota
- Balíček ROS s popsanými programy
- Video RestController v MP4
- Video TrotGaitController v MP4
- Video CrawlGaitController v MP4
- Video StandController v MP4
- Video RealnyRobot v MP4
- Video Simulace v MP4