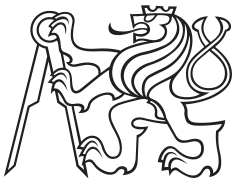
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídící techniky

Tapster robot

Matyáš Maňur

Vedoucí: Ing. Michal Tenkl
Obor: Kybernetika a robotika
Květen 2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Maňur** Jméno: **Matyáš** Osobní číslo: **498963**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Tapster robot

Název bakalářské práce anglicky:

Tapster robot

Pokyny pro vypracování:

1) Seznamte se s využitím robotů v automatizaci systémových testů v automotive
2) Seznamte se s konstrukcí robota a jeho dopřednou a inverzní kinematickou úlohou
3) Implementujte modul pro řízení robota
4) Návrhněte a implementujte uživatelské rozhraní

Seznam doporučené literatury:

[1] The Delta Parallel Robot: Kinematics Solutions, Robert L. Williams II, Ph.D., williar4@ohio.edu, Mechanical Engineering, Ohio University, October 2016, Dostupné z: https://www.ohio.edu/mechanical-faculty/williams/html/PDF/DeltaKin.pdf
[2] Clean Code in Python: Develop maintainable and efficient code

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Michal Tenkl Porsche Engineering Services, s.r.o. Radlická 714/113a, 158 00 Praha 5

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Ing. Martin Hlinovský, Ph.D. katedra řídicí techniky FEL

Datum zadání bakalářské práce: **31.01.2023** Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Michal Tenkl	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Michalu Tenklovi za cenné rady, připomínky a vstřícnost při zpracovávání téhle práce. Zároveň bych chtěl poděkovat firmě Porsche Engineering za podporu, prostředí a nástroje pro realizaci bakalářské práce.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval a realizoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

V Praze 26. května 2023

Abstrakt

Tato bakalářská práce je věnována analýze kinematiky Delta robotů spolu s vývojem řídící aplikace pro interakci s mobilním zařízením. V práci je proveden rozbor inverzní a dopředné kinematické úlohy spolu s analýzou pracovního prostoru paralelních robotů typu Delta. Pro reprezentaci interakcí robota s mobilním zařízením jsou vytvořeny datové struktury ve formátu JSON. Řídící aplikace je rozdělena do dílčích funkcionálních komponent. V závěru práce je experimentálně ověřen vytvořený koncept.

Klíčová slova: Delta robot, robot, kinematika, pracovní prostor, HiL, mobilní zařízení, systémové testování, Python, JSON-RPC, PlotLy, Telemetrix, Arduino, servo motor

Vedoucí: Ing. Michal Tenkl

Abstract

This bachelor thesis is devoted to the analysis of the kinematics of Delta robots along with the development of a control application for interaction with a mobile device. The inverse and forward kinematic problem is analyzed along with the workspace analysis of parallel Delta robots. JSON data structures are created to represent the robot's interactions with the mobile device. The control application is divided into sub-functional components. At the end of the paper, the developed concept is experimentally verified.

Keywords: Delta robot, robot, kinematics, workspace, HiL, mobile device, system testing, Python, JSON-RPC, PlotLy, Telemetrix, Arduino, servo motor

Title translation: Tapster robot

Obsah

Obrázky

Tabulky

Terminologie

- **Rameno (link)** - pevná část robota.
- **Kloub (joint)** - část robota umožňující řízený, či volný pohyb dvou ramen, které spojuje.
- **Koncový efektor, chapadlo (end effector)** - část manipulátoru, sloužící k uchopování nebo namontování dalších nástrojů.
- **Základna (rám, base)** - část manipulátoru, která je pevně spojena se zemí.
- **Kinematická dvojice (kinematic pair)** - dvojice ramen spojených kloubem.
- **Mechanismus** - kinematický řetězec, jehož alespoň jedno rameno je připevněno k zemi
- **DOF** - Počet stupňů volnosti, minimální počet nezávislých parametrů, které jednoznačně systém popisují.
- **Pracovní prostor** - prostor, kam se robot může dostat svým koncovým efektorem.
- **Rotační kloub** - kloub s jedním stupni volnosti.
- **Univerzální kloub** - kloub se dvěma stupni volnosti.
- **Sférický (kulový) kloub** - kloub se třemi stupni volnosti.
- **R-S-S** - Rotační-Sférický-Sférický (kinematický řetězec).
- **Interface** - Společná hranice mezi dvěma oddělenými komponentami aplikace.
- **PWM** - pulzně šířková modulace.
- **Třída** - datová struktura objektově orientovaného programování slouží k definování objektů.

- **Objekt** - instance třídy obsahující reálná data, jejichž struktura je popsána třídou.
- **Metoda** - metody objektu jsou funkce, které jsou zapsané v těle třídy, jejíž instancí objekt je.
- **Atribut** - proměnná třídy.
- **Konstruktor** - metoda volaná při vzniku objektu.
- **Modul** - soubor s koncovkou *.py* obsahující zdrojový kód.
- $\{\mathbf{B}\}$ - souřadný systém
- $\mathbf{X}_i^B$ - bod $\mathbf{X}$ s dolním indexem i vyjádřený v souřadném systému $\{\mathbf{B}\}$.
- $\vec{\mathbf{Y}}_i^B$ - vektor $\mathbf{Y}$ s dolním indexem i vyjádřený v souřadném systému $\{\mathbf{B}\}$.
- $[\mathbf{R}]_P^B$ - Rotační matice $[\mathbf{R}]$ ze souřadného systému $\{\mathbf{B}\}$ do souřadného systému $\{\mathbf{P}\}$.

Kapitola 1
Úvod

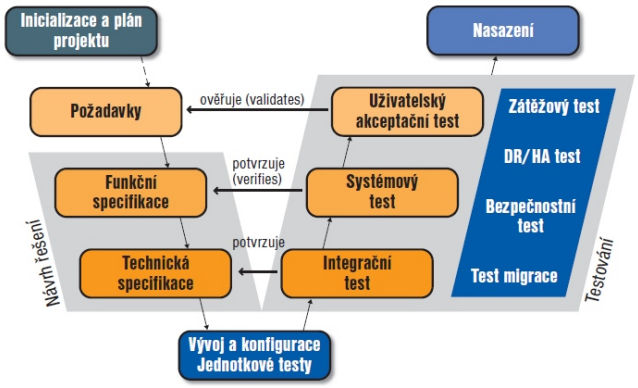
Úvodní část práce nás zasvětí do problematiky systémového testování v automobilovém průmyslu, s důrazem na simulační metodu HiL a využití robotického manipulátoru typu Delta pro automatizované testování mobilních aplikací. K pochopení fungování Delta robota nám poslouží sekce zaměřená na jeho kinematickou úlohu. Tato část se zaměřuje na popis struktury robota, jeho klíčových parametrů a analýzu vektorových smyček, na jejichž základě je možné sestavit kinematický model. Tento model nám pak umožní určit inverzní a dopřednou kinematickou úlohu pro Delta manipulátor.

Aby mohl robot interagovat s mobilním zařízením, vytvoříme systém, který bude založen na transformaci mezi souřadnicovými systémy. K nalezení rozměrů robota nám poslouží analýza jeho pracovního prostoru, jenž je závislý na kinematických parametrech robota. Po vyhodnocení jejich vlivu stanovíme optimální parametry pro interakci s dotykovými zařízeními různých rozměrů. Na základě těchto parametrů poté vytvoříme CAD model, který následně poslouží pro realizaci fyzického modelu pro ověření funkčnosti.

Vzhledem k tomu, že se bude jedná o testovací zařízení, kladli jsme při jeho návrhu důraz na jednoduchost mechanických komponent a elektroniky. Elektronika robota je tvořena jednodeskovým mikroprocesorovým počítačem Arduino a servomotory. K ovládání robota je nutná řídící aplikace, která musí disponovat vhodnou architekturou pro budoucí integraci do simulační metody HiL.

Tato řídící aplikace se skládá ze tří klíčových částí - Klient, Server a Robot. Klientská část interpretuje definované ovládací prvky mobilního telefonu a zpracovává uživatelské vstupy. Serverová část zahrnuje modul řídící aplikace, který řídí chod robota. Mezi jeho funkcionality patří převody mezi souřadnými systémy, kinematické výpočty, či komunikace s mikroprocesorem. Veškeré výpočty jsou provedeny v řídícím modulu a mikroprocesor pouze interpretuje kloubové souřadnice a převádí je na fyzický pohyb servomotorů. Poslední část tohoto textu se věnuje experimentálnímu ověření funkčnosti námi navrženého konceptu. Zkoušíme, jak se námi vytvořený systém osvědčí v praxi a zda splňuje všechny naše předpoklady a očekávání. Toto ověření nám poskytne cenné informace, které nám pomohou v dalších fázích vývoje a zdokonalování tohoto konceptu.

1.1 Úvod do problematiky



Obrázek 1.1: V-model vývojového řízení[1].

Vývoj v automobilovém průmyslu se řídí specifickým modelem, známým jako V-model. Tento model ilustruje proces, v němž se od počátečního návrhu až po finální implementaci postupuje od nejvyšší úrovně abstrakce směrem k detailním úrovním, přičemž v polovině procesu dochází k obratu a proces směřuje zpět k vyšším úrovním abstrakce. Tento model obsahuje řadu úrovní testování, včetně jednotkového, integračního, systémového a akceptačního testování. Mezi tyto úrovně patří také systémové testování, které se zaměřuje na testování kompletního systému jako celku. Toto je kritická fáze vývojového cyklu, která umožňuje odhalení problémů a chyb, které by se mohly projevit při běžném použití vozidla. Nyní se více zaměříme na tuto klíčovou etapu testování v automobilovém průmyslu.

Systémové testování hraje klíčovou roli při vývojovém cyklu v automotive[1]. Tato kriticky důležitá fáze při vývoji vozidla se provádí za účelem ověření ucelené funkčnosti a integrace všech dílčích subsystémů. Rozmanitost subsystémů automobilu, které zahrnují prvky jako pohon a podvozek, asistenční jízdní funkce, infotaniment, komfortní řídící jednotka, či komunikační gateway musí vzájemně kooperovat, aby byla zajištěna bezpečnost a spolehlivost celého vozidla. Systémové testování se zaměřuje na posouzení celkového chování systému za různých specifických podmínek. Tato úloha je často složitější, protože vyžaduje komplexní přehled a pochopení celého systému a jeho vzájemných vazeb a interakcí. Systémové testování umožňuje odhalení problémů, které nemusí být pozorovatelné při odděleném testování jednotlivých komponent, což je obzvlášť důležité, jelikož nesprávná vzájemná interakce komponent vozidla může vést k fatálním krizovým situacím.

Softwarové testování je jeden z hlavních aspektů systémového testování v automotive. Narozdíl od mechanických a elektronických komponent, které jsou ovlivňovány zejména fyzikálními faktory, se testování na systémové úrovni softwarových aplikací soustředí na ověření jeho funkčnosti - jako jsou stavové

[1]Automotive = Automobilový průmysl

automaty, reakce systému na neplatné stavy, chyby v komunikaci, konfigurace a interakce mezi systémy. Úkolem ověření funkčnosti je podrobné ověření správné realizace stanovených funkcích softwaru, které jsou schopny plnit uživatelské požadavky v reálném světě. Stabilita softwaru zajišťuje bezchybný chod za všech podmínek garantující absenci selhání při vytížení. Odezva na druhou stranu zkoumá reakční dobu softwaru na vstupy systému, přičemž názorný příklad je automatická detekce překážky a následné zabránění kolize.

HiL, neboli *Hardware in the Loop*[9], je důležitá metodologie užívaná v systémovém testování mající uplatnění především v automobilovém průmyslu, kde je kladen důraz na zkoumání interakce softwarových a hardwarových komponent. HiL je simulační metoda poskytující definovatelné realistické prostředí emulující podmínky skutečného světa sloužící k vyhodnocování a validaci požadovaných vlastností komponent vozidla. Tato metoda ve svém jádru obsahuje propojení fyzických hardwarových komponent se simulovaným prostředím. Tyto fyzické hardwarové komponenty mohou být z jakékoliv části vozidla, avšak zde se specifikujeme pouze na elektronické komponenty, mezi které patří například hlavní řídící jednotka ECU[2], senzory, či aktuátory. Tyto komponenty jsou zapojené ve vazební smyčce se simulovaným prostředím imitujícím podmínky reálného světa. To umožňuje rozsáhle testování řídících algoritmů, odezev elektronických komponent, zkoumání chování celého komplexního systému zatíženého širokým spektrem vnějších vlivů, či bezpečného otestování krajních případů a stavů, které jsou nereplikovatelné v reálném světě.

Mobilní telefony se v posledních letech stávají nedílnou součástí automobilů přesahující svůj původní záměr komunikačního zařízení. S rozmachem ve vývoji moderních bezdrátových komunikačních technologií mezi které patří Wifi, či Bluetooth se otevřela zcela nová doména systémů vozidla poskytující řadu uživatelských funkcionalit od jednoduchých informačních prvků, přes personalizaci nastavení automobilu, až po složité autonomní funkce ovládající celé vozidlo. Vzhledem k úzkému propojení mobilního zařízení se systémy vozidlem je nezbytné zahrnout zahrnout tuto doménu do systémového testování nevyjímaje HiL simulací. Účelem začlenění mobilních zařízení do HiL simulací je ověření bezchybné interakce mezi mobilním zařízením, softwarovými aplikacemi a elektronikou vozidla.

1.1.1 Systémové testování mobilního zařízení v interakci s automobilem

Testování mobilních zařízení v automobilovém průmyslu představuje výzvu především kvůli různorodosti jednotlivých mobilních zařízení spolu v kombinaci komplexní povahy s nimi pojenými softwarovými aplikacemi. Jeden přístup je vytvořit kompletní simulaci mobilního zařízení s testovací aplikací. Tato simulace vyžaduje zdrojový kód mobilní aplikace spolu s vývojovým prostředím umožňujícím její nasazení. Simulované zařízení plně neodráží chování a vlastnosti fyzického zařízení, což zásadně limituje její přesnost a

[2]Electronic Control Unit

aplikovatelnost, obvzlášť v kontextu systémového testování, kde je nezbytné užití reálného zařízení.

Alternativní přístup k softwarové simulaci mobilního zařízení zahrnuje fyzickou integraci zařízení do HiL simulace. Mobilní zařízení zde slouží jako uživatelské rozhraní interagující s elektronickou řídící jednotkou poskytující vstupy systému, na které musí reagovat. Mobilní zařízení zde slouží jako takzvaný "black box", neboli zařízení kde je pozorovatelné jeho vnější chování pomocí vstupů a výstupů, avšak vnitřní chování jeho funkcionalit a mechanismů zůstává neznámé[10]. V tomto kontextu se při testování HiL často využívá automatizované testování s pomocí různých nástrojů. Pozitivum tohoto přístupu se nachází v testování chování aplikace při nasazení v praxi, kde se mohou vynořit nové problémy, jež by při úplné softwarové simulaci nemusely být zřejmé. Avšak tato metoda přináší své negativa, přičemž mezi největší z nich patří potřeba fyzického operátora ovládající mobilní zařízení. Manuální operace mobilního zařízení při systémovém testování je neefektivní, časově náročná a náchylná na neúmyslné chyby. Toto negativum je umocněno při automatizaci systémového testování, čímž se zde tvoří prostor pro robotizaci a tedy hlavní motivaci využití Delta robota k tomutu úkonu.

1.2 Upřesnění pojmů

V této sekci si upřesníme pojmy a grafické značky, se kterými se budeme v práci setkávat.

1.2.1 Python

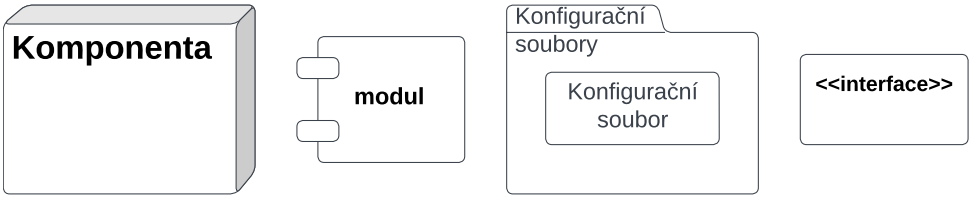
Jedná se o interpretovaný programovací jazyk umožňující objektově orientované, procedurální a v omezené míře i funkcionální programování. Díky svým zabudovaným vysokoúrovňovým datovým strukturám, množstvím rozšiřujících knihoven, kombinovanými s dynamickým psaním a linkováním se jedná o velice atraktivní jazyk použitý pro takzvaný *Rapid application development*, neboli rychlý vývoj aplikací [11]. Díky těmto vlastnostem se jedná o ideální programovací jazyk použitý k vývoji veškerých komponent aplikace.

1.2.2 Značení

Pro vhodné rozlišení jednotlivých termínů týkající se programovacího jazyka Python si zavedeme následující textové a grafické značení:

- Třídu si budeme označovat kurzívou s velkým počátečním písmenem - *Třída*
- Metodu si budeme označovat kurzívou s malým počátečním písmenem - *metoda*
- Komponentu si budeme značit tučným písmem s velkým počátečním písmenem - **Komponenta**

- Modul si budeme značit tučným písmem s kurzívou s malým počátečním písmenem - ***modul***



Obrázek 1.2: Grafická interpretace softwarových termínů.

Je nutno podotknout, že toto grafické značení nijak nesouvisí se značením používaným v softwarovém inženýrství. Účel je pro vizuálně přehlednější popis sotwarových částí v práci.

1.2.3 Telemetrix

Pro komunikaci s mikroprocesorem Arduino je použita komunikační knihovna Telemetrix. Jedná moderní nahrazení komunikačnímu protokolu Firmata[12], jenž slouží ke komunikaci mezi mikroprocesorem a počítačem. Telemetrix se skládá z Python API, použitému k vytvoření klientské aplikace v Pythonu, a z C++ serveru, se kterým komunikuje Python client přes sériovou linku, či WIFI[13]. Pro mikrokontrolér Arduino využijeme knihovnu Telemetrixu *Telemetrix4Arduino*[14], která využívá komunikace přes USB. Pro Python API vyžijeme knihovny Telemtrixu *telemetrix*[15].

1.2.4 JSON

JSON, neboli JavaScript Object Notation, je moderní jednoduchý datový formát který je lehce interpretovatelný jak lidmi, tak stroji. Je široce užívaný k přenosu dat mezi serverovými a klientskými aplikacemi. Výhoda JSON formátu je jeho nezávislost na jiných programovacích jazycích[16]. V našem případě je formát JSON použit pro tvorbu konfiguračních souborů.

1.2.5 JSON-RPC

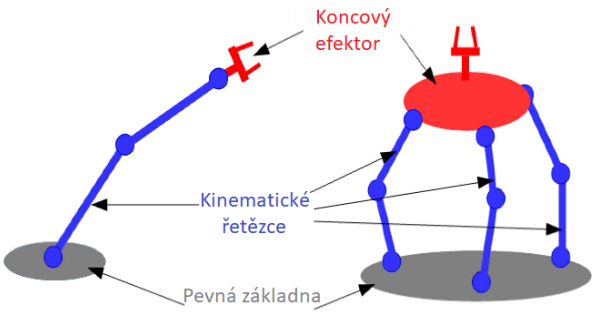
JSON-RPC (remote procedure call) je komunikační protokol užívající formátu JSON. Poskytuje klientu možnost volat metody na serveru využívající jednoduchého textově založeného formátu. JSON-RPC umožňuje komunikaci mezi aplikacemi napříč různými platformami a programovacími jazyky[17].

Kapitola 2
Teorie

V této kapitole se seznámíme s paralelním robotem typu Delta, pro nějž sestavíme kinematický model spolu s řešením inverzní a dopředné kinematické úlohy. Následně vymyslíme způsob interpretace vztahu mobilního zařízení a robota. Na závěr provedeme analýzu vlivu jednotlivých kinematických parametrů na pracovní prostor robota.

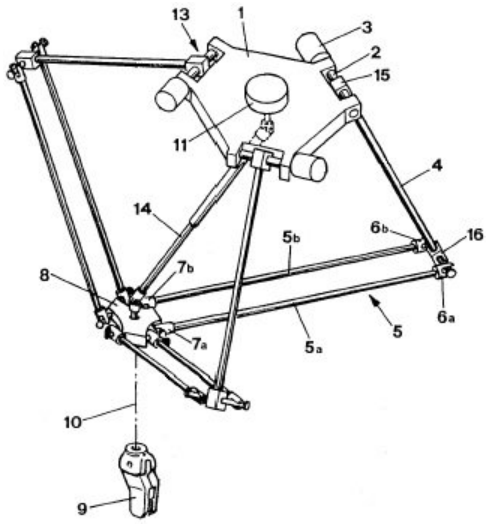
2.1 Paralelní robot typu Delta

Paralelní manipulátory jsou odvětví robotických systémů, kde je koncový efektor spojen se základnou několika ekvivalentními paralelními kinematickými řetězci. Rozdíl mezi sériovým a paralelním manipulátorem pozorujeme na obrázku 2.1.



Obrázek 2.1: Rozdíl mezi sériovým a paralelním manipulátorem[2].

Poprvé byly paralelní manipulátory představeny Goughem a Stewardem roku 1965 vývojem Gough-Stewardovy platformy[18]. Typ Delta je jedním z nejvýznamnějších zástupců paralelních robotů. Tento robot byl vynalezen a patentován v roce 1985 profesorem Reymondem Clavellem na Švýcarské federální technické škole v Lausanne[19]. Konstrukce Delta robota se skládá ze dvou trojúhelníkových platforem reprezentujících základnu a koncový efektor, které jsou spojené třemi kinematickými řetězci. Kinematický řetězec je tvořen otočným ramenem a paralelogramem[20], které v této kombinaci tvoří řetězec typu **RSS**. Rameno je spojeno rotační vazbou s nepohyblivou základnou. Paralelogram je spojen s opačným koncem ramene a koncovým efektorem. Tato



Obrázek 2.2: Patentovaný Clavelův Delta robot[3].

konstrukce eliminuje rotační pohyb koncového efektoru a zajišťujě translační pohyb se třemi stupni volnosti. Tento typ robota poskytuje nejednu výhodu oproti sériovému robotu - dosažení vysoké dynamiky, přesnosti, robustnosti a snížení setrvačnosti koncového efektoru díky své odlehčené konstrukci. Své uplatnění naleznou ve velké škále průmyslových odvětví, ze kterých si uvedeme některé příklady:

- **Montáž a manipulace**: Díky své vysoké přesnosti a rychlosti se často používají v montážních linkách, kde je třeba rychle a přesně manipulovat s malými díly.
- **Balení a třídění**: Jsou ideální pro balení produktů v rychlých výrobních linkách, jako je například balení potravin, léků nebo kosmetiky. Taktéž se využívají pro třídění objektů podle různých kritérií.
- **Pick and place aplikace**: Díky svému unikátnímu kinematickému designu a rychlému pohybu jsou Delta roboti vhodné pro "pick and place"aplikace, kde je třeba rychle přesunovat položky z jednoho místa na druhé.
- **Laboratorní a farmaceutický průmysl**: V laboratořích a farmaceutickém průmyslu se používají pro přesné a opakované manipulace, jako je například pipetování.
- **Elektronický průmysl**: Ve výrobě elektronických součástek, jako jsou desky plošných spojů, se používají pro rychlou a přesnou manipulaci s malými součástkami.

2.2 Kinematická úloha Delta manipulátoru

Kinematická úloha umožňuje pochopení a kontrolu pohybu koncového efektoru robota na základě aktuace jednotlivých kloubů. Kinematická úloha typicky obsahuje dva hlavní aspekty: dopřednou a inverzní kinematiku. Dopředná kinematika je proces umožňující vypočítat polohu koncového efektoru na základě vstupních kloubových souřadnic. Inverzní kinematická úloha umožňuje určit potřebné kloubové souřadnice k dosažení referenční pozice koncového efektoru.

2.2.1 Stupně volnosti Delta robota

Tato sekce dokazuje, že mobilita (počet stupňů volnosti) Delta robota jsou 3 DoF. Využijeme Čebyšev-Grübler-Kutzbachovy rovnice pro Delta robot zobrazený na obrázku 2.3 [5]:

$$M = 6(N-1) - 5J_1 - 4J_2 - 3J_3, \tag{2.1}$$

kde:

M je mobilita, či počet stupňů volnosti,
N je počet členů mechanismu včetně základny,
J_1 je počet kloubů s jedním stupněm volnosti,
J_2 je počet kloubů se dvěma stupni volnosti,
J_3 je počet kloubů se třemi stupni volnosti,

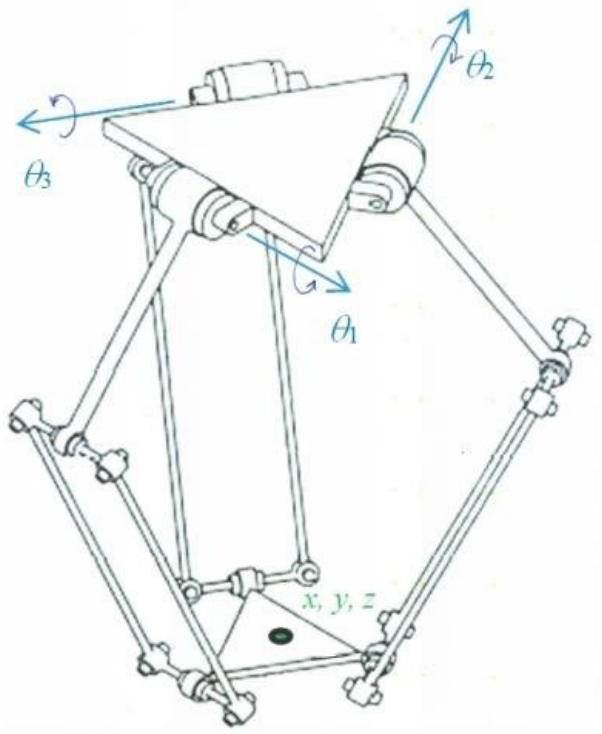
Pro určení počtu stupňů volnosti budeme uvažovat zjednodušenou verzi Delta robota, kde je paralelogram nahrazen jednoduchým ramenem u kterého uvažujeme sférický kloub na obou stranách. Pro takto zjednodušený typ Delta robota vyjdou z Kutzbachovy rovnice 3 stupně volnosti:

$$\begin{aligned} &N = 8, \\ &J_1 = 3, \\ &J_2 = 6, \\ &J_3 = 0. \end{aligned} \qquad M = 6(14-1) - 5(15) - 4(0) - 3(0) = 3, \tag{2.2}$$

2.2.2 Popis Delta robota s otočným ramenem

Na obrázku 2.3 lze pozorovat Delta robota se třemi stupni volnosti složeného ze tři identických **RSS** kinematických řetězců mezi fixovanou základnou a spodním pohybujícím se koncovým efektorem. Vrchní 3 rotační klouby jsou popsány obecnými souřadnicemi θ_i, $i = 1, 2, 3$. V tomto modelu jsou θ_i měřeny pravidlem pravé ruky[1], přičemž směr palce koreluje se zobrazením na obrázku 2.3. Nulový úhel je definován při poloze hnaného ramene v horizontální rovině základny.

[1] Palec je směru v osy otáčení, prsty ukazují ukazují směr pohybu



Obrázek 2.3: Delta robot s vyznačenými rotačními klouby[4].

Delta robot se třemi stupni volnosti je schopen xyz translačního pohybu koncového efektoru. Základní kinematické parametry modelu jsou:

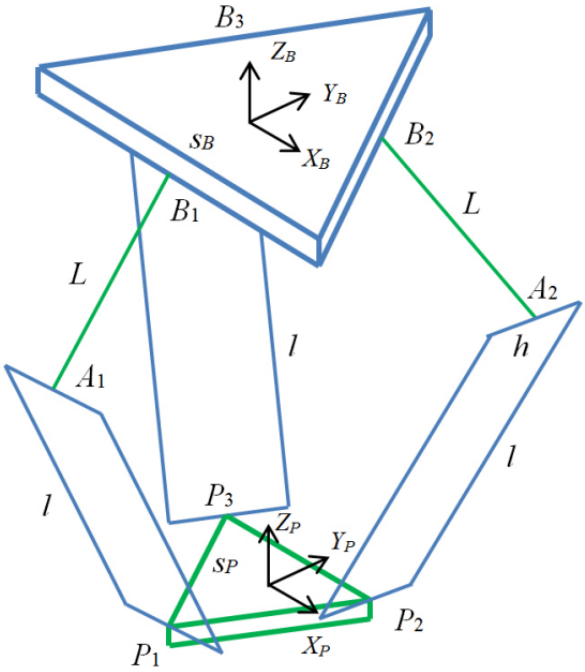
- $\mathbf{s_b}$ - délka strany rovnostranného trojúhelníku základny,
- $\mathbf{s_p}$ - délka strany rovnostranného trojúhelníku koncového efektoru,
- $\mathbf{L}$ - délka horního ramene,
- $\mathbf{l}$ - délka paralelogramu.

Rovnostranný trojúhelník koncového efektoru je invertován vůči rovnostrannému trojúhelníku základny jako ukázáno na obrázku 2.4.

Fixovaný počátek kartézského systému $\{\mathbf{B}\}$ je umístěn ve středu rovnostranného trojúhelníku základny. Počátek kartézského systému pohybujícího se koncové efektoru $\{\mathbf{P}\}$ je umístěn ve středu jeho rovnostranného trojúhelníku. Orientace $\{\mathbf{P}\}$ je vždy totožná s orientací $\{\mathbf{B}\}$, tudíž rotační matice $[\mathbf{R}]_P^B = [\mathbf{I}]_3$ je jednotková. Kloubové souřadnice jsou $\vec{\mathbf{\Theta}} = \begin{bmatrix}\theta_1 & \theta_2 & \theta_3\end{bmatrix}^T$. Design zobrazený výše je silně symetrický se třemi horními rameny délky a třemi spodními paralelogramy.

Umístění otočných kloubů, které reprezentují body $\mathbf{B}_i^B$, jsou konstantní v souřadném systému základny $\{\mathbf{B}\}$:

$$\mathbf{B}_1^B = \begin{bmatrix} 0 \\ -w_B \\ 0 \end{bmatrix}, \qquad \mathbf{B}_2^B = \begin{bmatrix} \frac{\sqrt{3}}{2} w_B \\ \frac{1}{2} w_B \\ 0 \end{bmatrix}, \qquad \mathbf{B}_3^B = \begin{bmatrix} -\frac{\sqrt{3}}{2} w_B \\ \frac{1}{2} w_B \\ 0 \end{bmatrix}. \tag{2.3}$$



Obrázek 2.4: Kinemtický diagram Delta robota[5].

Polohy univerzálních kloubů $\mathbf{P}_i^P$ jsou konstantní v souřadném systému koncového efektoru $\{\mathbf{P}\}$:

$$\mathbf{P}_1^P = \begin{bmatrix} 0 \\ -u_P \\ 0 \end{bmatrix}, \qquad \mathbf{P}_2^P = \begin{bmatrix} \frac{s_P}{2} \\ w_P \\ 0 \end{bmatrix}, \qquad \mathbf{P}_3^P = \begin{bmatrix} -\frac{s_P}{2} \\ w_P \\ 0 \end{bmatrix}, \tag{2.4}$$

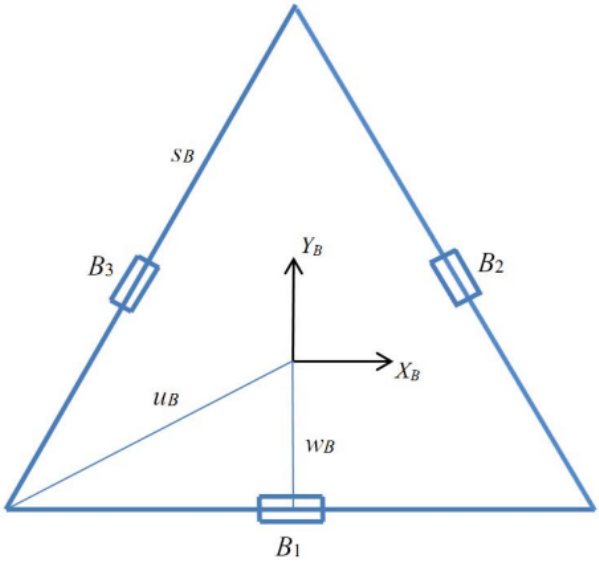
kde:

$$w_B = \frac{\sqrt{3}}{6} s_B, \qquad u_B = \frac{\sqrt{3}}{3} s_B, \qquad w_P = \frac{\sqrt{3}}{6} s_P, \qquad u_P = \frac{\sqrt{3}}{3} s_P. \tag{2.5}$$

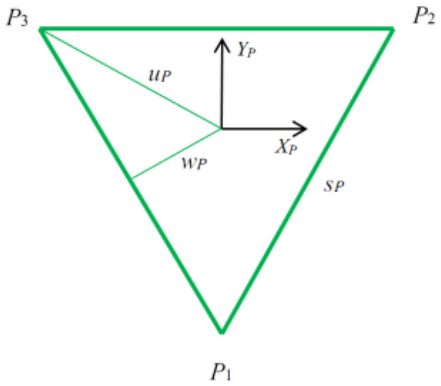
Název veličiny	**Význam**
s_B	Délka strany rovnostranného trojúhelníku základny
s_P	Délka strany rovnostranného trojúhelníku koncového efektoru
L	Délka rotačního ramene
l	Délka paralelogramu
w_B	Vzdálenost od počátku $\{\mathbf{B}\}$ k nejbližší straně trojúhelníku
u_B	Vzdálenost od počátku $\{\mathbf{B}\}$ k vrcholu trojúhelníku
w_P	Vzdálenost od počátku $\{\mathbf{P}\}$ k nejbližší straně trojúhelníku
u_P	Vzdálenost od počátku $\{\mathbf{P}\}$ k vrcholu trojúhelníku

Tabulka 2.1: Popis kinematických veličin[5]

Geometrické detaily fixované základny a koncového efektoru jsou:

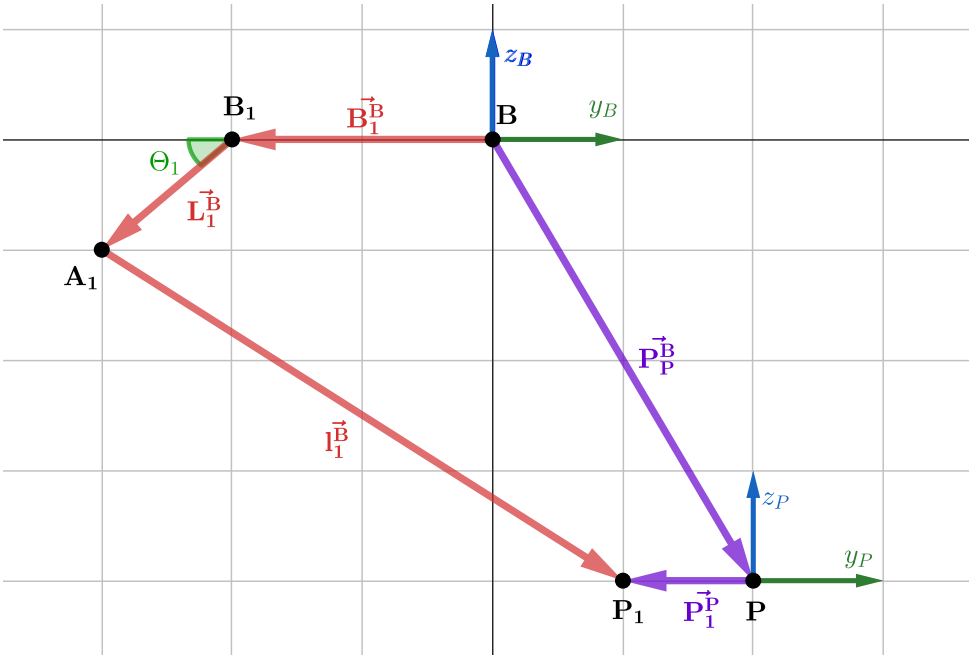


Obrázek 2.5: Detail fixované základny[5].



Obrázek 2.6: Detail koncového efektoru[5].

2.2.3 Kinematická analýza Delta robota s rotačními rameny



Obrázek 2.7: Detail vektorové smyčky v YZ rovině.

Z kinematického diagramu 2.4 platí následující trojice vektorových smyček popisující jednotlivá ramena Delta robota:

$$\vec{\mathbf{B}}_i^B + \vec{\mathbf{L}}_i^B + \vec{\mathbf{l}}_i^B = \vec{\mathbf{P}}_P^B + [\mathbf{R}]_P^B \vec{\mathbf{P}}_i^P = \vec{\mathbf{P}}_P^B + \vec{\mathbf{P}}_i^P, \qquad i = 1, 2, 3, \tag{2.6}$$

kde $[\mathbf{R}]_P^B = [\mathbf{I}]_3$ jelikož nedochází k rotaci koncového efektoru. Kartézské souřadnice středu koncového efektoru (počátek souřadného systému $\{\mathbf{P}\}$) vyjádřeného v souřadném systému $\{\mathbf{B}\}$ jsou $\vec{\mathbf{P}}_P^B = \begin{bmatrix} x & y & z \end{bmatrix}^T$. Konstantní vektorové hodnoty pro body $\mathbf{P}_i$ a $\mathbf{B}_i$ jsou již definovány v rovnicích 2.4 a 2.5. Vektory $\vec{\mathbf{L}}_i^B$ jsou závislé na hodnotě kloubových souřadnic $\vec{\mathbf{\Theta}} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \end{bmatrix}^T$.

Na Obrázku 2.7 je znázorněna vektorová smyčka pro rotační kloub θ_1 v yz rovině souřadného systému $\{\mathbf{B}\}$. Ze znázorněné smyčky je snadné najít vektor $\vec{\mathbf{L}}_1^B$. Pro nalezení vektoru $\vec{\mathbf{L}}_2^B$ a $\vec{\mathbf{L}}_3^B$ je využito symetrie.

$$\begin{aligned}
\vec{\mathbf{L}}_1^B &= \begin{bmatrix} 0 \\ -L\cos\theta_1 \\ -L\sin\theta_1 \end{bmatrix}, \\
\vec{\mathbf{L}}_2^B &= [\mathbf{R_z}](120°) \begin{bmatrix} 0 \\ -L\cos\theta_2 \\ -L\sin\theta_2 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} L\cos\theta_2 \\ \frac{1}{2} L\cos\theta_2 \\ -L\sin\theta_2 \end{bmatrix}, \\
\vec{\mathbf{L}}_3^B &= [\mathbf{R_z}](-120°) \begin{bmatrix} 0 \\ -L\cos\theta_3 \\ -L\sin\theta_3 \end{bmatrix} = \begin{bmatrix} -\frac{\sqrt{3}}{2} L\cos\theta_3 \\ \frac{1}{2} L\cos\theta_3 \\ -L\sin\theta_3 \end{bmatrix}.
\end{aligned} \tag{2.7}$$

Kde L je délka horní části ramene a rotační matice je:

$$[\mathbf{R_z}](\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.8}$$

Dosazením výše uvedených hodnot do rovnice popisující vektorovou smyčku 2.6 jsou vektory $\vec{\mathbf{l}_i^B}$ rovny:

$$\vec{\mathbf{l}_i^B} = \vec{\mathbf{P}_P^B} + \vec{\mathbf{P}_i^P} - \vec{\mathbf{B}_i^B} - \vec{\mathbf{L}_i^B} \tag{2.9}$$

$$\begin{aligned}
\vec{\mathbf{l}_1^B} &= \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ -u_P \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ -w_B \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ -L\cos\theta_1 \\ -L\sin\theta_1 \end{bmatrix} = \begin{bmatrix} x \\ y + L\cos\theta_1 + a \\ z + L\sin\theta_1 \end{bmatrix}, \\
\vec{\mathbf{l}_2^B} &= \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \frac{s_P}{2} \\ w_P \\ 0 \end{bmatrix} - \begin{bmatrix} \frac{\sqrt{3}}{2} w_B \\ \frac{1}{2} w_B \\ 0 \end{bmatrix} - \begin{bmatrix} \frac{\sqrt{3}}{2} L\cos\theta_2 \\ \frac{1}{2} L\cos\theta_2 \\ -L\sin\theta_2 \end{bmatrix} = \begin{bmatrix} x - \frac{\sqrt{3}}{2} L\cos\theta_2 + b \\ y - \frac{1}{2} L\cos\theta_2 + c \\ z + L\sin\theta_2 \end{bmatrix}, \\
\vec{\mathbf{l}_3^B} &= \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} -\frac{s_P}{2} \\ w_P \\ 0 \end{bmatrix} - \begin{bmatrix} -\frac{\sqrt{3}}{2} w_B \\ \frac{1}{2} w_B \\ 0 \end{bmatrix} - \begin{bmatrix} -\frac{\sqrt{3}}{2} L\cos\theta_3 \\ \frac{1}{2} L\cos\theta_3 \\ -L\sin\theta_3 \end{bmatrix} = \begin{bmatrix} x - \frac{\sqrt{3}}{2} L\cos\theta_3 - b \\ y - \frac{1}{2} L\cos\theta_3 + c \\ z - L\sin\theta_3 \end{bmatrix},
\end{aligned} \tag{2.10}$$

kde:

$$\begin{aligned}
a &= w_B - u_P, \\
b &= \frac{s_P}{2} - \frac{\sqrt{3}}{2} w_B, \\
c &= w_P - \frac{1}{2} w_B.
\end{aligned} \tag{2.11}$$

Využijeme faktu, že známe Euklidovskou normu vektoru $\vec{\mathbf{l}_i^B}$, jež je délka paralelogramu l:

$$l = \|\vec{\mathbf{l_i}}\| = \sqrt{\vec{\mathbf{l_i}}^T \vec{\mathbf{l_i}}}. \tag{2.12}$$

Pro vyvarování se odmocnině v Euklidovské normě jsou obě strany rovnice umocněny na druhou:

$$l^2 = \vec{\mathbf{l}_i}^T \vec{\mathbf{l}_i} = l_{ix}^2 + l_{iy}^2 + l_{iz}^2 \tag{2.13}$$

Nyní je již možné sestavit kinematické rovnice popisující Delta manipulátor:

$$\begin{aligned}
&2L(y+a)\cos\theta_1 + 2zL\sin\theta_1 + x^2 + y^2 + z^2 + a^2 + L^2 + 2ya - l^2 = 0 \\
&-L(\sqrt{3}(x+b) + y + c)\cos\theta_2 + 2zL\sin\theta_2 + x^2 + y^2 + z^2 + b^2 + c^2 \\
&\qquad + L^2 + 2xb + 2yc - l^2 = 0 \\
&L(\sqrt{3}(x-b) - y - c)\cos\theta_3 + 2zL\sin\theta_3 + x^2 + y^2 + z^2 + b^2 + c^2 \\
&\qquad + L^2 - 2xb + 2yc - l^2 = 0
\end{aligned} \tag{2.14}$$

2.2.4 Inverzní kinematická úloha

Inverzní kinematická úloha (IKT) se zabývá převedením požadovaných kartézských souřadnic koncového efektoru $\mathbf{P}_P^{\vec{B}} = \begin{bmatrix} x & y & z \end{bmatrix}^T$ na tři potřebné souřadnice rotačních kloubů $\vec{\mathbf{\Theta}} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \end{bmatrix}^T$. Analytické řešení IKT spočívá v řešení kinematických rovnic Delta robota 2.14. IKT je vypočítáno pro každé rameno robota zvlášť. Geometrické řešení IKT spočívá v nalezení průniku známé kružnice o poloměru L, se středem v bodě $\mathbf{B}_i^B$ a sféry o poloměru l se středem v bodě $\mathbf{P}_i^P$. IKT bude pro potřeby práce vyřešeno analyticky.

Tři kinematické rovnice Delta robota (2.14) jsou ve tvaru:

$$E_i \cos\theta_i + F_i \sin\theta_i + G_i = 0 \qquad i = 1, 2, 3, \tag{2.15}$$

kde:

$$\begin{aligned} E_1 &= 2L(y+a) \\ F_1 &= 2zL \\ G_1 &= x^2 + y^2 + z^2 \end{aligned}$$

$$\begin{aligned} E_2 &= -L(\sqrt{3}(x+b) + y + c) \\ F_2 &= 2zL \\ G_2 &= x^2 + y^2 + z^2 + b^2 + c^2 + L^2 + 2xb + 2yc - l^2 \end{aligned} \tag{2.16}$$

$$\begin{aligned} E_3 &= L(\sqrt{3}(x-b) - y - c) \\ F_3 &= 2zL \\ G_3 &= x^2 + y^2 + z^2 + b^2 + c^2 + L^2 - 2xb + 2yc - l^2 \end{aligned}$$

Rovnice $E_i \cos\theta_i + F_i \sin\theta_i + G_i = 0$ se často objevuje v robotice a lze vyřešit pomocí substituci za tangens poloviční úhlu[5]. Pokud zadefinujeme $t_i = \tan\frac{\theta_i}{2}$, pak $cos\theta_i = \frac{1-t_i^2}{1+t_i^2}$ a $\sin\theta_i = \frac{2t_i}{1+t_i^2}$. Substitucí rovnice 2.15 dostaneme:

$$\begin{aligned} E_i\left(\frac{1-t_i^2}{1+t_i^2}\right) + F_i\left(\frac{2t_i}{1+t_i^2}\right) + G_i = 0, \\ (G_i - E_i)t_i^2 + (2F_i)t_i + (G_i + E_i) = 0 \end{aligned} \tag{2.17}$$

s řešením kvadratické rovnice:

$$t_{i_{1,2}} = \frac{-F_i \pm \sqrt{E_i^2 + F_i^2 - G_i^2}}{G_i - E_i}. \tag{2.18}$$

Pro získaní θ_i invertujeme původní substituci:

$$\theta_i = 2\tan^{-1} t_i \tag{2.19}$$

Obecně jsou dvě řešení pro všechna θ_i vycházející z $\pm$ kvadratické rovnice. Obě dvě řešení jsou správně - vnitřní a vnější pozice natočení ramene. Celkem

existuje 8 IKT různých validních řešení. Pro robota budeme volit vždy vnější pozici natočení ramene, což znamená volbu záporného znaménka v kvadratické rovnici. Tímto dostaneme pouze jedno validní řešení IKT.

2.2.5 Dopředná kinematická úloha

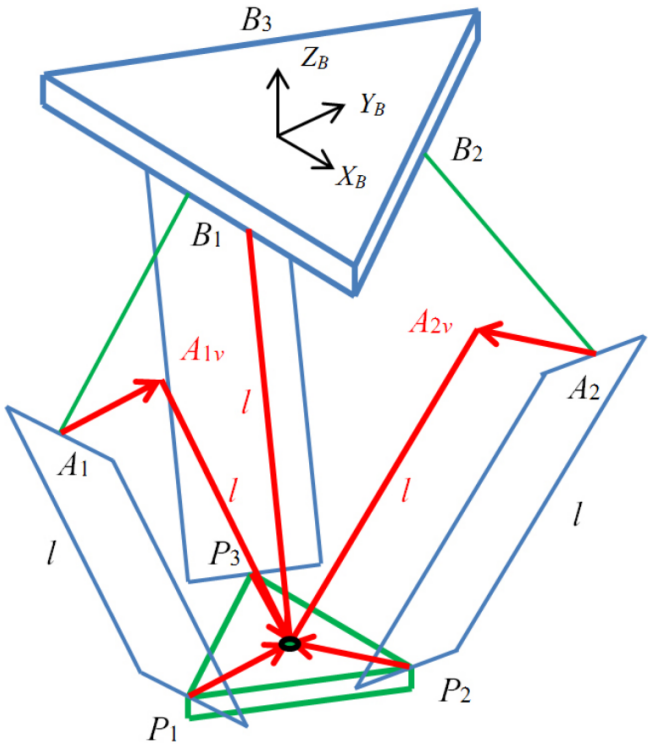
Dopředná kinematická úloha (DKT) se zabývá převedení kloubových souřadnic $\vec{\mathbf{\Theta}} = \begin{bmatrix}\theta_1 & \theta_2 & \theta_3\end{bmatrix}^T$ na kartézské souřadnice koncového efektoru $\vec{\mathbf{P}}_P^B = \begin{bmatrix}x & y & z\end{bmatrix}^T$. Obecně je DKT pro paralelní manipulátory velmi složitá. Její řešení vyžaduje šetření spolusouvisejících nelineárních algebraických rovnic, které vycházejí z kinematických rovnic, jež jsou aplikovaný na vektorové smyčky.

Avšak díky pouhému translačnímu pohybu koncového efektoru se třemi stupni volnosti existuje jednoznačné analytické řešení této úlohy. Jelikož $\vec{\mathbf{\Theta}} = \begin{bmatrix}\theta_1 & \theta_2 & \theta_3\end{bmatrix}^T$ je dáno, dopočítáme si hodnoty vektorů $\vec{\mathbf{A}}_i^B = \vec{\mathbf{B}}_i^B + \vec{\mathbf{L}}_i^B$, $i = 1, 2, 3$. Jelikož víme, že orientace natočení koncového efektoru je konstantní a rovna $[\mathbf{R}]_P^B = [\mathbf{I}]_3$, definujeme si tři teoretické koule se středem $\vec{\mathbf{A}}_{iV}^B = \vec{\mathbf{A}}_i^B - \vec{\mathbf{P}}_i^P$, $i = 1, 2, 3$ pozorovatelné na obrázku 2.8:

$$\vec{\mathbf{A}}_{1V}^B = \begin{bmatrix} 0 \\ -w_B - L\cos\theta_1 + u_p \\ -L\sin\theta_1 \end{bmatrix}, \qquad \vec{\mathbf{A}}_{2V}^B = \begin{bmatrix} \frac{\sqrt{3}}{2}(w_B + L\cos\theta_2) - \frac{s_B}{2} \\ \frac{1}{2}(w_B + L\cos\theta_2) - w_P \\ -L\sin\theta_2 \end{bmatrix},$$
$$\vec{\mathbf{A}}_{3V}^B = \begin{bmatrix} -\frac{\sqrt{3}}{2}(w_B + L\cos\theta_3) + \frac{s_B}{2} \\ \frac{1}{2}(w_B + L\cos\theta_3) - w_P \\ -L\sin\theta_3 \end{bmatrix}. \tag{2.20}$$

Výsledné řešení DKT úlohy pro Delta robota je průsečík tří sfér. Definujme si sféru vektorem reprezentující středový bode $\vec{c}$ a skalárem repretezentující poloměr r, tedy výsledná sféra bude $(\vec{c}, r)$. Řešení DKT, neboli bod $\mathbf{P}_P^B$, je průsečík sfér:

$$(\vec{\mathbf{A}}_{1V}^B, l), \qquad (\vec{\mathbf{A}}_{2V}^B, l) \qquad (\vec{\mathbf{A}}_{3V}^B, l). \tag{2.21}$$



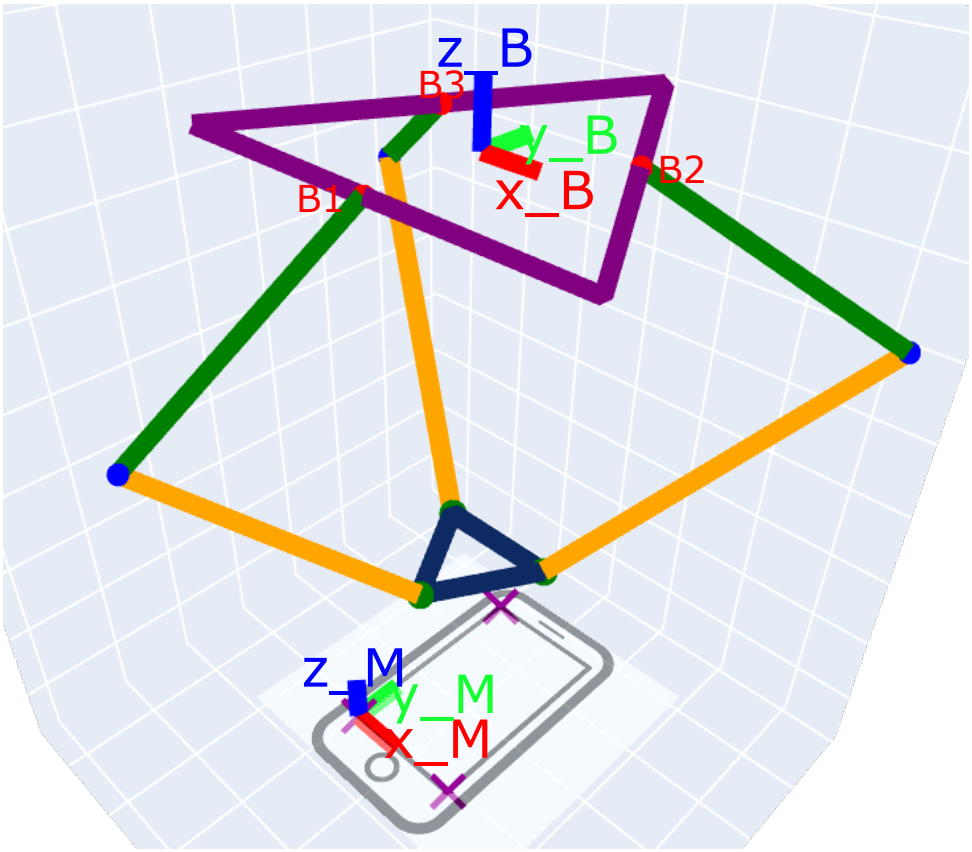
Obrázek 2.8: Kinematický diagram Delta robota pro DKT[5].

2.3 Interpretace dotykového zařízení

V této sekci se budeme zabývat interpretací mobilního zařízení v souřadném systému robota. Na obrázku 2.9 pozorujeme dva souřadné systémy. Souřadný systém $\{\mathbf{B}\}$ je systém robota (viz obrázek 2.4) ležící ve středu trojúhelníkové základny. Druhý souřadný systém s $\{\mathbf{M}\}$ má počátkem v levém dolním rohu obrazovky mobilního zařízení. Jednotlivé operace k ovládání mobilního zařízení budou vyjádřeny v souřadném systému $\{\mathbf{M}\}$. Cílem je vytvořit převodní vztah umožňující robotu interagovat s dotykovým zařízením, jež je uživatelsky jednoduše získatelné. Uživatel nakalibruje okrajové body zařízení, které umožní získání vztahu. K tomuto účelu využijeme transformační matice $[\mathbf{T}]^B_M$ umožňující převod ze souřadného systému $\{\mathbf{M}\}$ do $\{\mathbf{B}\}$. Vztah reprezentující tento převod je:

$$\vec{x}^B = \mathbf{T}^B_M \vec{x}^M, \tag{2.22}$$

kde $\vec{x}^M$ reprezentuje bod vyjádřený v souřadném systému $\{\mathbf{M}\}$ a $\vec{x}^B$ reprezentuje bod vyjádřený v souřadném systému $\{\mathbf{B}\}$. Homogenní transformační matice $[\mathbf{T}]$ je v robotice matice rozměru 4x4 jež je složena z rotační matice $[\mathbf{R}]$ (3x3) a translačního vektoru $\vec{t}$ (3x1) kombinující jak rotaci tak translaci



Obrázek 2.9: Zobrazení souřadných systémů.

mezi dvěma souřadnými systémy[21].

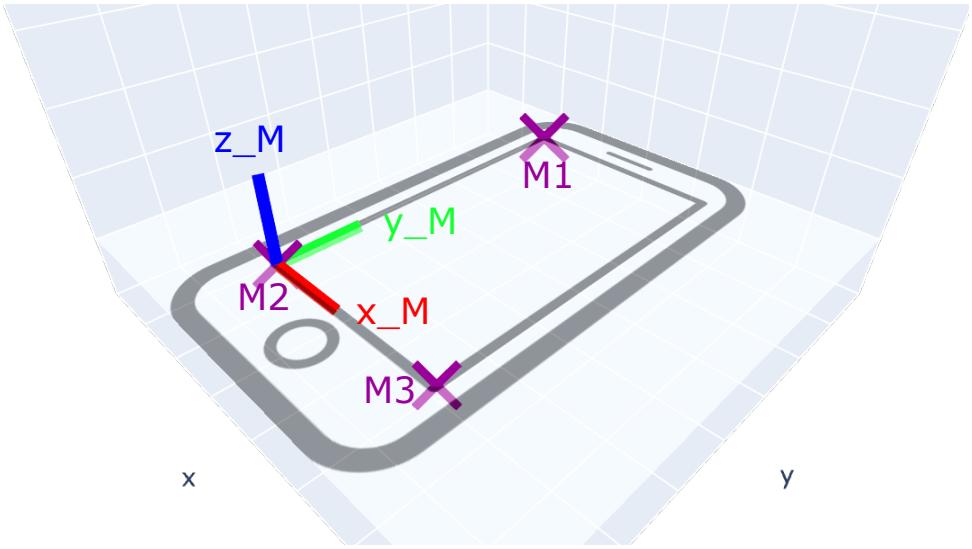
$$[\mathbf{T}] = \begin{bmatrix} & [\mathbf{R}] & & \vec{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.23}$$

Pro nalezení matice [**T**] potřebujeme zjistit rovinu obrazovky. Tu si definujeme třemi okrajovými body obrazovky viz obrázek 2.10:

- **M1** - levý horní roh obrazovky
- **M2** - levý dolní roh obrazovky
- **M3** - pravý dolní roh obrazovky

Tyto body **M** jsou vyjádřeny v souřadném systému {**B**}. Jelikož se bod **M2** shoduje s počátkem souřadného systému {**M**}, translační vektor je totožný s tímto bodem.

$$\vec{t} = \mathbf{M2} \tag{2.24}$$



Obrázek 2.10: Souřadný systém dotykového Zařízení.

Pro získání rotační matice je zvolen následující postup. Od bodů **Mi**, $i = 1, 2, 3$ odečteme translační vektor:

$$\begin{aligned} \mathbf{M1}' &= \mathbf{M1} - \vec{t} \\ \mathbf{M2}' &= \mathbf{M2} - \vec{t} \\ \mathbf{M3}' &= \mathbf{M3} - \vec{t} \end{aligned} \tag{2.25}$$

Nyní si definujeme vektory $\vec{m_1}$ a $\vec{m_2}$ jako:

$$\begin{aligned} \vec{m_1} &= \mathbf{M1}' - \mathbf{M2}' \\ \vec{m_2} &= \mathbf{M3}' - \mathbf{M2}' \end{aligned} \tag{2.26}$$

Musíme zde počítat s faktem, že vektory $\vec{m_1}$ a $\vec{m_2}$ nemusí být ortogonální vlivem nepřesnosti v uživatelském měření. Pro nalezení ortonormální báze souřadného systému $\{\mathbf{M}\}$ budeme postupovat následovně. Z principu velikosti mobilní obrazovky bude vždy vektor $\vec{m_1}$ delší než vektor $\vec{m_2}$, čímž se z něj stává vhodnější referenční vektor. Bázový vektor $\vec{y_M}$ získáme normalizací vektoru $\vec{m_1}$:

$$\vec{y_M} = \frac{\vec{m_1}}{\|\vec{m_1}\|} \tag{2.27}$$

Bázový vektor $\vec{z_M}$ je ortogonální s vektory $\vec{m_1}$ a $\vec{m_2}$ a získáme ho jako vektorový součin normalizovaného vektoru $\vec{m_2}$ a vektoru $\vec{y_M}$:

$$\vec{z_M} = \frac{\vec{m_2}}{\|\vec{m_2}\|} \times \vec{y_M} \tag{2.28}$$

Poslední bázový vektor $\vec{x_M}$ je vektorový součin vektorů $\vec{y_M}$ a $\vec{z_M}$:

$$\vec{x_M} = \vec{y_M} \times \vec{z_M} \tag{2.29}$$

Při dodržení uvedeného pořadí ve vektorových součinech získáme ortonormální bázové vektory tvořící pravotočivou bázi. Rotační matice [**R**] ze vztahu 2.23 je tvořena získanými bázovými vektory:

$$[\mathbf{R}] = \begin{bmatrix} \vec{x_M} & \vec{y_M} & \vec{z_M} \end{bmatrix} \tag{2.30}$$

Nyní již můžeme sestavit transformační matici $[\mathbf{T}]_M^B$ umožňující trasformaci ze souřadného systému {**M**} do souřadného systému {**B**}.

$$[\mathbf{T}]_M^B = \begin{bmatrix} \vec{x_M} & \vec{y_M} & \vec{z_M} & \vec{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.31}$$

2.4 Analýza pracovního prostoru Delta robota

Pracovní prostor hraje důležitou roli při návrhu designu robota, jelikož nám určuje rozsah pohybů a dosah koncového efektoru, což přímo ovlivňuje schopnosti robota interagovat s mobilním zařízením. Pracovní prostor je závislý na kinematickém modelu robota (viz sekce *Kinematická úloha Delta manipulátoru*). V této sekci se zaměříme na analýzu vhodných kinematických parametrů popisující Delta robota, jež mají vliv na pracovní prostor a mobilitu robota.

2.4.1 Základní kinematické parametry Delta robota

Celý kinematické model je ovlivněn čtyřmi základními parametry:

- **L** - Délka horní části ramene,
- **l** - Délka dolní části ramene,
- $\mathbf{s_B}$ - Délka strany rovnostranného trojúhelníku základny,
- $\mathbf{s_P}$ - Délka strany rovnostranného trojúhelníku koncového efektoru.

Tyto parametry hrají klíčovou roli v určení pracovního prostoru Delta robota, jelikož určují možné pozice ramen a koncového efektoru. Analýza těchto parametrů nám umožní pochopení a určení vhodného pracovního prostoru potřebného k obsluze mobilních zařízení.

2.4.2 Omezující podmínky

Kinematický model umožňuje umístění koncového efektoru jak nad základnou Delta robota, tak pod ní v závislosti na konfiguraci jednotlivých ramen. Avšak při naší aplikaci robota budeme uvažovat pouze polohu koncové efektoru pod základnou robota, což znamená, že souřadnice z koncového efektoru nikdy nemůže být kladná (obrázek 2.9).

Druhé omezení se týká fyzického rozsahu servo motorů, které musíme brát v potaz. Při určitých konfiguracích mohou kloubové souřadnice dosáhnout záporných hodnot. Největší záporná hodnota dosáhnutelná kloubovými souřadnicemi je přímo omezena rozsahem servo motorů. Tento parametr si definujeme jako θ_{MIN} a ve vztahu kloubových souřadnic nám určuje minimální hodnotu, jenž mohou kloubové souřadnice nabývat. Se stejnou myšlenkou si definujeme parametr θ_{MAX} určující maximální možnou velikost kloubových souřadnic. Kloubové souřadnice tedy musí splňovat následující podmínku:

$$\theta_{MIN} \leq \theta_i \leq \theta_{MAX}, \ i = 1, 2, 3. \tag{2.32}$$

2.4.3 Získání bodů tvořící pracovní prostor

Pro získání bodů pracovního prostoru si nejdříve vygenerujeme rovnoměrně rozloženou síť 3D bodů v potenciálním rozsahu robota. Pro každý z těchto bodů vypočteme inverzní kinematickou úlohu pokud tato inverzní kinematická úloha má řešení a výsledné kloubové souřadnice splňují omezující podmínkami, body si uložíme.

2.4.4 Vizualizace pracovního prostoru

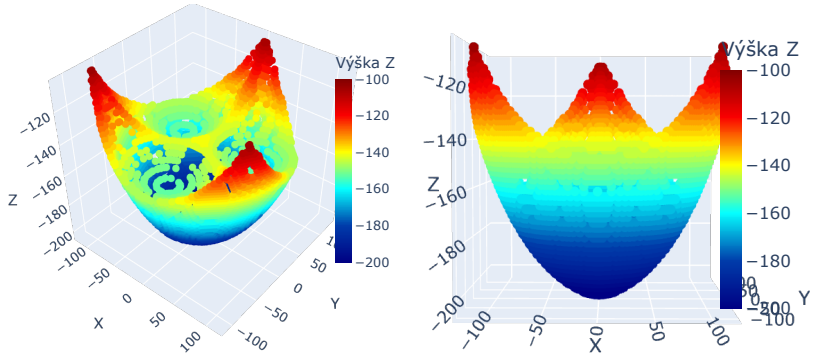
K efektivní vizualizaci pracovního prostoru Delta robota využijeme programovacího jazyka Python (více v sekci *Python*) s využitím grafické knihovny Plotly[22]. Veškeré vizualizace jsou dostupné k interakci ve formě *.html* souborů v přiložených souborech. Naše počáteční hodnoty parametrů zvolíme následující:

$$\begin{gathered} \mathbf{L} = 52 \text{ [mm]}, \quad \mathbf{l} = 153 \text{ [mm]}, \quad \mathbf{s_B} = 127 \text{ [mm]}, \quad \mathbf{s_P} = 41 \text{ [mm]}, \\ \theta_{\mathbf{MIN}} = 0°, \quad \theta_{\mathbf{MAX}} = 90°. \end{gathered} \tag{2.33}$$

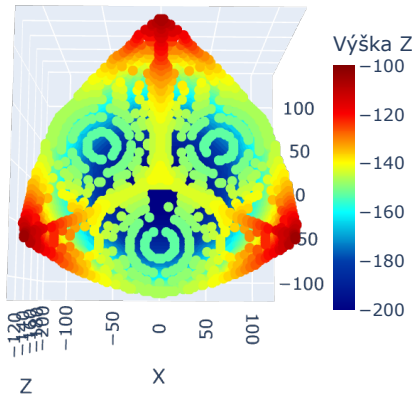
Tyto hodnoty jsou hodnoty Delta robota *tapsterbot* jež slouží později v práci jako inspirace základ CAD modelu. Na obrázku 2.11 vidíme vizualizaci pracovního prostoru. Pro přehlednější zobrazení byla použita barevná škála měnící se s se souřadnicí z bodů pracovního prostoru. Jedná se o neobvyklý útvar rozdělený na 3 totožné trisekce. Tato vizualizace nám dává základní představu o možnostech rozsahu Delta robota. Využitelná část pracovního prostoru se rozprostírá přibližně v rozmezí hodnot $-155 \leq z \leq -170$. Při hodnotách $z \geq -155$ se již robot nemůže pohybovat v celém rozsahu pracovního prostoru. Při hodnotách $z \leq -170$ se značně zužuje plocha xy v níž se se robot dokáže pohybovat. Při hledání optimálních kinematických parametrů se tedy bude zaměřovat na maximálně využitelný pracovní prostor, jež je přibližně v $\frac{1}{3}$ až $\frac{2}{3}$ v rozsahu z hodnot.

2.4.5 Vliv kinematických parametrů na pracovní prostor

Nyní se zaměříme na pozorování vlivů změny kinematických parametrů na pracovní prostor. Pro každý parametr se podíváme, jak vypadá pracovní



(a) : Rohový pohled. **(b) :** Pohled z boku.



(c) : Pohled shora.

Obrázek 2.11: Vizualizace pracovního prostoru robota *tapsterbot*[6].

prostor pro dvojnásobnou a poloviční hodnotu parametru. Jelikož pro vizualizaci není důležitý rozměr, budou uvedené parametry a grafy bezrozměrné. Pro přehlednost prostorového umístění jednotlivých pracovních prostorů jsou zde barevně rozlišeny hodnoty z jednotlivých bodů. U každého obrázku je referenční barevná škála, jež je pro všechny následující obrázky v této sekci stejná. Jako referenční využijeme parametry z předchozí sekce:

$$\mathbf{L} = 52, \quad \mathbf{l} = 153, \quad \mathbf{s_B} = 127, \quad \mathbf{s_P} = 41, \quad \theta_{\mathbf{MIN}} = 0°, \quad \theta_{\mathbf{MAX}} = 90°. \tag{2.34}$$

Vliv parametru θ_{MAX} nebudeme zkoumat, jelikož pro úhel $\theta_{MAX} \geq 90°$ by do sebe ramena fyzicky zasahovala. V sekci *Vliv parametru* θ_{MIN} jsou kinematické parametry konstantní a zkoumáme pouze vliv parametru θ_{MIN}.

Z analýzy obrázků 2.12 - 2.21 pozorujeme následující vlastnosti. Změnou kinematických parametrů se nám pracovní prostory rozdělily na dvě skupiny nesoucí podobné charakteristiky. První skupinu pozorujeme na obrázcích 2.12, 2.15, 2.16, 2.19. Využitelná část pracovního prostoru, ve které je robot schopen manipulovat s mobilním zařízením je značně malá a nevyhovující. Okrajové části připomínající rohy jsou zbytečně velké a robot jich nikdy nevyužije. Pro vysokou hodnotu parametru $\mathbf{l}$ (obrázek 2.15) je pracovní prostor hluboko pod základnou robota, což by mělo za důsledek zbytečně dlouhé rameno a rám robota. Rozsah těchto prostorů v osách xy dosahuje rozdílných hodnot, avšak

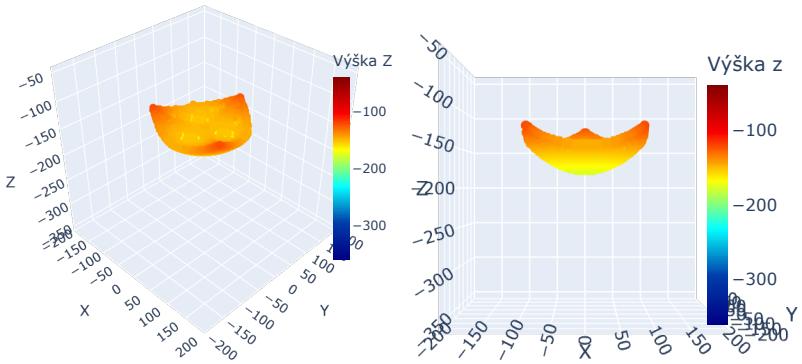
při porovnání s referenční pracovním prostorem z předchozí sekce (obrázek 2.11) jde vždy o nevyužitelné okrajové části.

Druhou skupinu pracovních prostorů pozorujeme na obrázcích 2.13, 2.14, 2.17, 2.18. Tyto pracovní prostory jsou charakteristické absencí nevyužitelných okrajových částí. Sic tyto prostory dosahují nižších rozměrů v osách xy, celý jejich objem je lépe využitelný pro naší aplikaci. Nejmarkantnější rozdíl pozorujeme při změně kinematického parametru $\mathbf{L}$ na obrázku 2.13. Tento pracovní prostor dosahuje největších rozměrů, má minimum okrajových nevyužitelných částí a dochází k jednomu z nejmenších zužování ve výšce z, kde by potenciálně mohlo být mobilní zařízení. Za zmínku též stojí změna parametru $\theta_{\mathbf{MIN}}$ pozorovatelná na obrázcích 2.20 a 2.21. Intuitivně zvětšení minimálního možné úhlu dosažitelného servo motory má za výsledek zjevné zmenšení pracovního prostoru. Avšak snížení tohoto parametru má za následek výrazné zvětšení užitelného objemu.

Z této analýzy kinematických parametrů jsme získali následující poznatky:

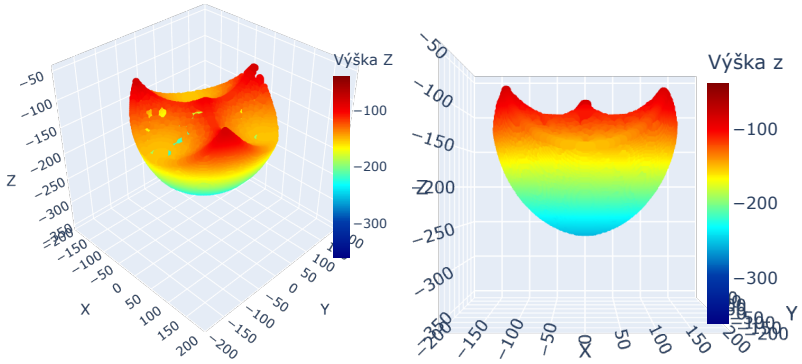
- Vhodnou volbou parametru $\mathbf{L}$ můžeme maximalizovat pracovní prostor.
- Parametr $\mathbf{l}$ ovlivňuje hloubku, jež je schopné robot dosáhnout.
- Vliv parametrů $\mathbf{s_b}$ a $\mathbf{s_b}$ má velice podobný charakter a není třeba s nimi příliš manipulovat.
- Parametr $\theta_{\mathbf{MIN}}$ je vhodné minimalizovat pro dosažení vhodného výškového rozsahu.

Vliv parametru L



(a) : Rohový pohled. **(b) :** Pohled z boku.

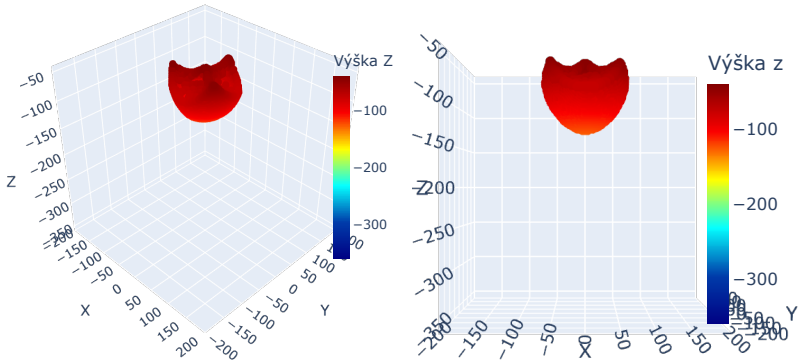
Obrázek 2.12: Parametry: $\underline{\mathbf{L} = 26}$, $\mathbf{l} = 153$, $\mathbf{s_B} = 127$, $\mathbf{s_P} = 41$.



(a) : Rohový pohled. **(b) :** Pohled z boku.

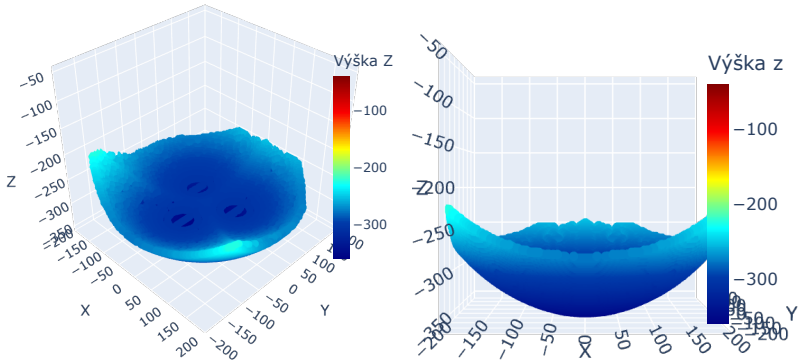
Obrázek 2.13: Parametry: $\underline{\mathbf{L} = 104}$, $\mathbf{l} = 153$, $\mathbf{s_B} = 127$, $\mathbf{s_P} = 41$.

Vliv parametru l



(a) : Rohový pohled. **(b) :** Pohled z boku.

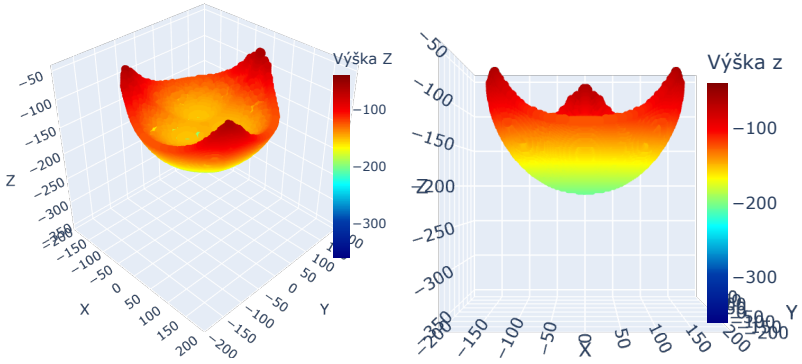
Obrázek 2.14: Parametry: $\mathbf{L} = 52$, $\underline{\mathbf{l} = 77}$, $\mathbf{s_B} = 127$, $\mathbf{s_P} = 41$.



(a) : Rohový pohled. **(b) :** Pohled z boku.

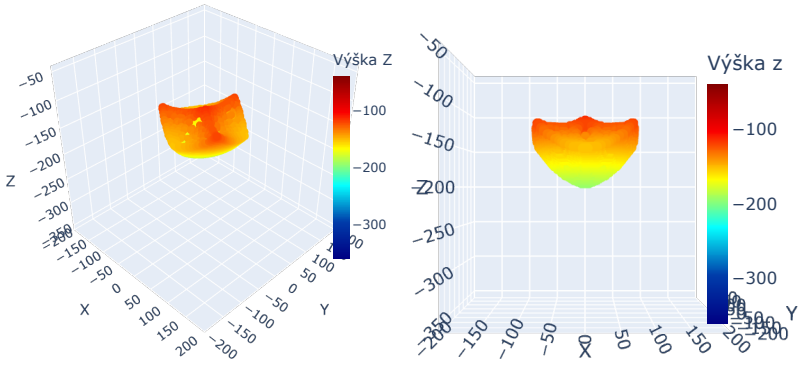
Obrázek 2.15: Parametry: $\mathbf{L} = 52$, $\underline{\mathbf{l} = 306}$, $\mathbf{s_B} = 127$, $\mathbf{s_P} = 41$.

Vliv parametru s_b



(a) : Rohový pohled. **(b) :** Pohled z boku.

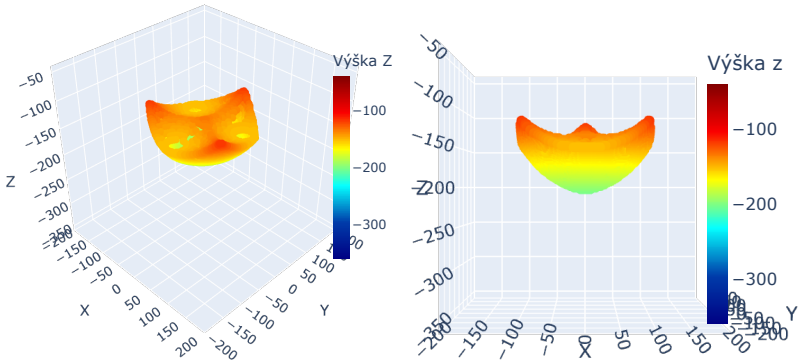
Obrázek 2.16: Parametry: $\mathbf{L} = 52$, $\mathbf{l} = 153$, $\underline{\mathbf{s_B} = 64}$, $\mathbf{s_P} = 41$.



(a) : Rohový pohled. **(b) :** Pohled z boku.

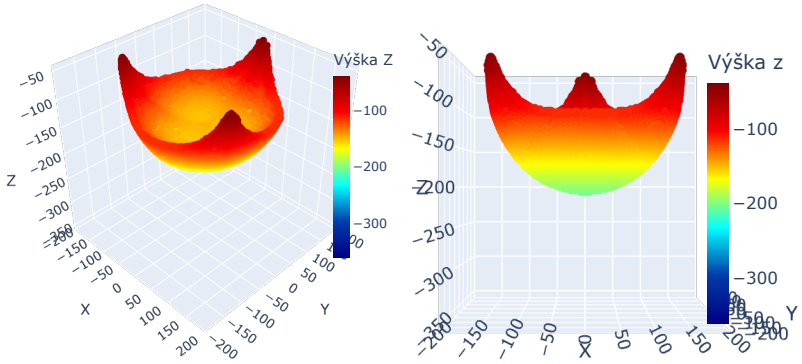
Obrázek 2.17: Parametry: $\mathbf{L} = 52$, $\mathbf{l} = 153$, $\underline{\mathbf{s_B} = 257}$, $\mathbf{s_P} = 41$.

Vliv parametru s_p



(a) : Rohový pohled. **(b) :** Pohled z boku.

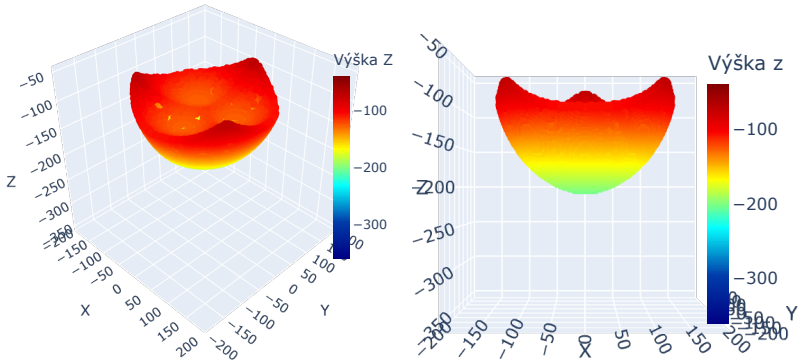
Obrázek 2.18: Parametry: $\mathbf{L} = 52$, $\mathbf{l} = 153$, $\mathbf{s_B} = 127$, $\underline{\mathbf{s_P} = 20}$.



(a) : Rohový pohled. **(b) :** Pohled z boku.

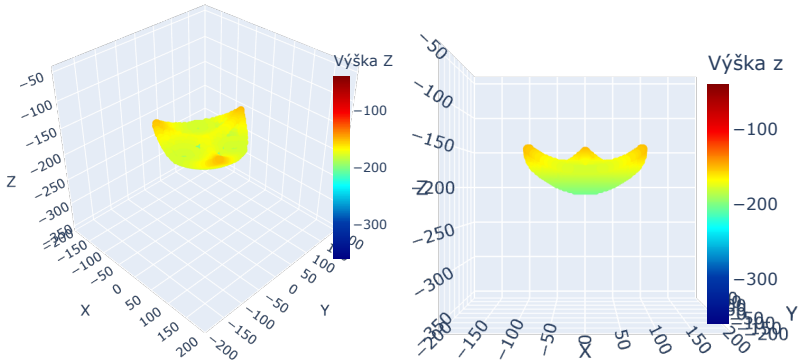
Obrázek 2.19: Parametry: $\mathbf{L} = 52$, $\mathbf{l} = 153$, $\mathbf{s_B} = 127$, $\mathbf{s_P} = 82$.

Vliv parametru θ_{MIN}



(a) : Rohový pohled. **(b) :** Pohled z boku.

Obrázek 2.20: Parametry: $\theta_{MIN} = -30°$.



(a) : Rohový pohled. **(b) :** Pohled z boku.

Obrázek 2.21: Parametry: $\theta_{MIN} = 30°$.

2.4.6 Nalezení vhodných kinematických parametrů

V této sekci se budeme zabývat nalezení vhodných kinematických parametrů pro Delta robota, jež umožní interakci s celou škálou velikostí mobilních zařízení. Zadefinujeme si hlavní kritérium podle kterého budeme určovat parametry a následně nalezneme nové kinematické parametry.

Definice problému

Hlavní kritérium pro nalezení parametrů pro Delta robota je schopnost interagovat s celou obrazovkou mobilního zařízení. Úhlopříčky obrazovek moderních mobilních zařízení se nejčastěji pohybují v rozmezí 4.7 palce (11.9 cm) až 6.7 palce (17.0 cm)[23]. Pokud do tohoto rozsahu zahrneme i tablety, můžeme se dostat až na úhlopříčku obrazovky 13 palců (33.0 cm). Zvolme si maximální úhlopříčku obrazovky ovládatelné Delta robotem 11 palců (27.9 cm) s rozměry stran 22.3×16.7 cm. Tato hodnota pokrývá všechny běžně dostupné mobilní zařízení s nimiž by mohl robot interagovat.

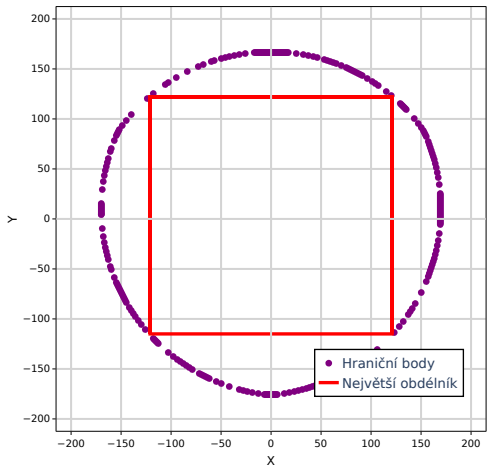
Metodiku pro hledání vhodných kinematických parametrů umožňující robotu ovládat obrazovku o úhlopříčce 11 palců si zvolíme velice jednoduchou. Budeme hledat kvádr, jež celým svým objemem leží v pracovním prostoru robota a délky jeho stran jsou minimálně 22.3×16.7 cm. Tloušťka většiny moderních mobilních zařízení je maximálně 1 cm. Prostor nad obrazovkou, ve kterém by se měl být schopen robot pohybovat si definujeme též jako 1 cm. Výsledná výška tohoto kvádru je 2 cm. Hledání optimálních parametrů budeme provádět ručně. Pro různé kinematické parametry budeme pozorovat pracovní prostor v němž na dané výšce z zobrazíme maximální možný kvádr určený hraničními body pracovního prostoru v dané výšce. Tuto operaci je možné zalgoritmizovat, avšak pro náš případ by to bylo složité a nepotřebné.

Hledání maximálního kvádru v množině bodů

Ve specifické výšce z si provedeme řez pracovním prostorem. Tímto dostaneme dvoudimenzionální množinu hraničních bodů. V této množině budeme hledat obdélník s maximálním obsahem. K tomuto účelu využijeme Python knihovny *largestinteriorrectangle*[24]. Tato knihovna umožňuje nalezení největšího možné obdélníku v polygonu. K tomuto využívá algoritmu popsaného v článku *Algorithm for finding the largest inscribed rectangle in polygon*[25]. V našem případě si hraniční body seřadíme do takového pořadí, aby za sebou vytvořené hrany ze sekvence bodů tvořily konvexní polygon (algoritmus umí pracovat i s nekonvexnímy polygony, avšak my se s tímto případem nesetkáme). Výsledek této operace pozorujeme na obrázku 2.22.

2.4.7 Vhodné kinematické parametry

Při hledání parametrů byl též kladen důraz na fyzické aspekty robota, který je popsán v následující kapitole. Jednotlivé délky ramen nemůžou být příliš

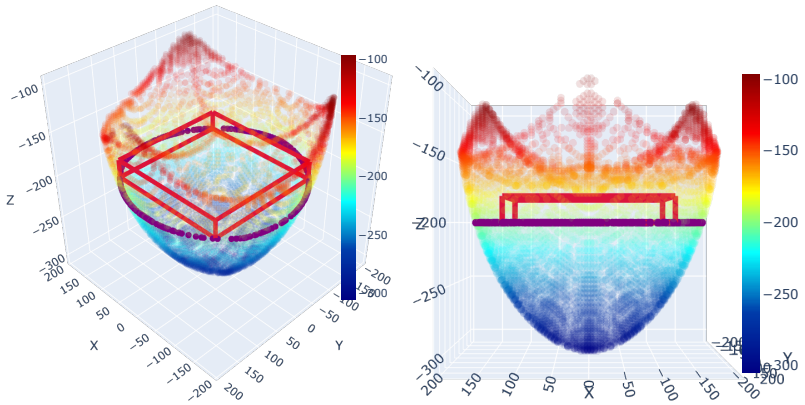


Obrázek 2.22: Největší obdélník v množině hraničních bodů.

dlouhé, jelikož by zde mohl nastat problém s nedostatečným výkonem použitých servo motorů. Délka horní trojúhelníkové základny zůstala zachována jako u referenčního *tapsterbota*, jelikož je zde vhodně vyřešené namontování motorů. Parametr $\theta_{\mathbf{MIN}}$ byl minimalizován, pro nižší hodnoty by docházelo ke kolizím se samotnými servomotory. Výsledné kinematické parametry umožňující obsluhu 11 palcové obrazovky jsou:

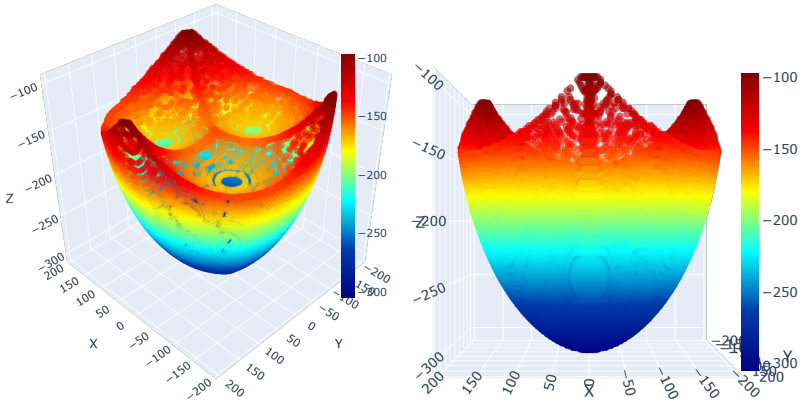
$$\begin{gathered}\mathbf{L} = 80 \text{ [mm]}, \quad \mathbf{l} = 217 \text{ [mm]}, \quad \mathbf{s_B} = 128 \text{ [mm]}, \quad \mathbf{s_P} = 30 \text{ [mm]}, \\ \theta_{\mathbf{MIN}} = -30°, \quad \theta_{\mathbf{MAX}} = 90°.\end{gathered} \tag{2.35}$$

Na obrázku 2.23 pozorujeme výsledný pracovní prostor se zobrazeným maximálním kvádrem. Kvádr má rozměry $242 \times 237 \times 20$ [mm]. Podstava kvádru leží v hloubce $z = -200$ [mm], pohled na příčný řez touto vrstvou je pozorovatelný na obrázku 2.22. Celý objem kvádru leží v pracovním prostoru, čímž reprezentuje dostatečný prostor pro umístění mobilního zařízení s prostorem potřebným pro pohyb koncového efektoru. Na obrázku 2.24 pozorujeme detail pracovního prostoru.



(a) : Rohový pohled. **(b) :** Pohled z boku.

Obrázek 2.23: Pracovní prostor s vizualizovaným kvádrem.



(a) : Rohový pohled. **(b) :** Pohled z boku.

Obrázek 2.24: Detail pracovního prostoru.

Kapitola 3
Design Delta robota

V této kapitole se seznámíme s modelem Delta robota spolu s použitou elektronikou. Postupně se zaměříme na CAD model Delta robota, použitý mikroprocesor a pohybové aktuátory.

3.1 CAD model Delta robota

CAD model nachází silnou inspiraci v open-source Delta robota *tapsterbot* od Tapster Robotics[6]. Avšak pro naší aplikaci má tento robot moc malý rozsah a jsou zde odlišnosti od kinematického modelu. S využitím kinematických parametrů z předchozí kapitoly jsem navrhl nový design Delta robota, jenž zachovává jednoduchou konstrukci referenčního modelu, avšak jí rozšiřuje o vylepšený paralelogram se schopností obsluhovat obrazovku až o velikosti 11 palců. Model je zobrazený na obrázku 3.1. Veškeré díly jsou navrhnuté k jednoduchému tisku na 3D tiskárně.

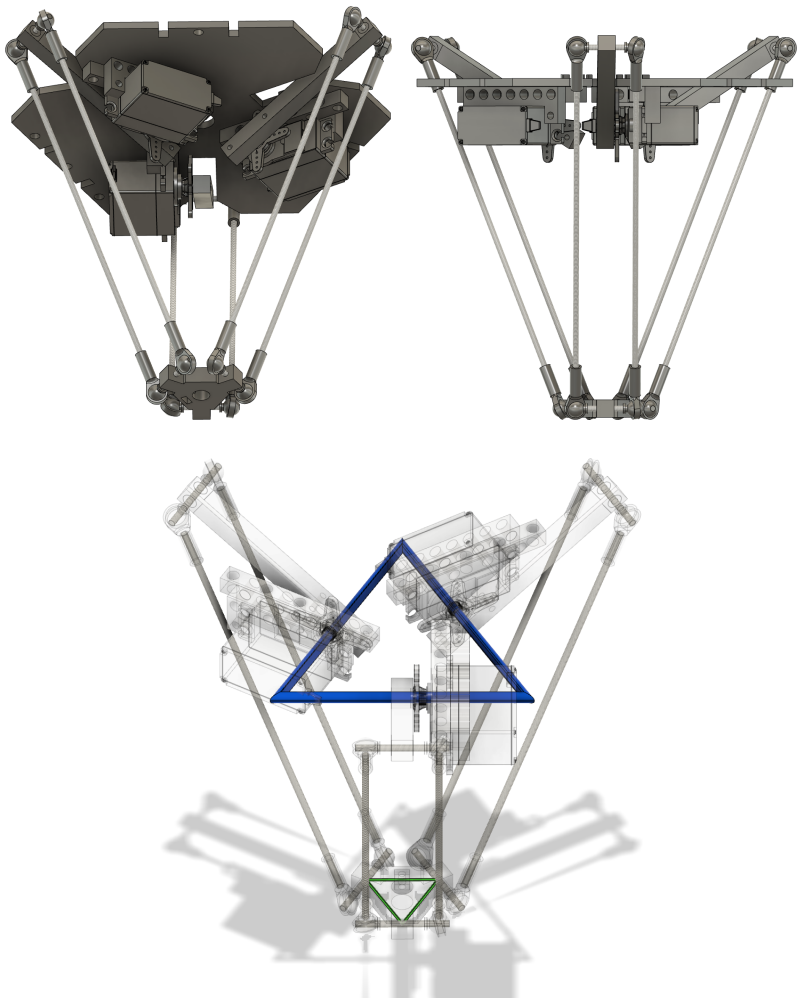
Základna a koncový efektor

Na obrázku 3.3 pozorujeme detail umístění servo motorů v základně robota spolu s detailem koncového efektoru. Jsou zde barevně zobrazené rovnostranné trojúhelníky z referenčního kinematického modelu (obrázek 2.4). Umístění servo motorů je převzaté z referenčního modelu *tapsterbot*. Serva jsou napolohována způsobem, aby osa otáčení jednotlivých ramen byla ve středu jednotlivých stran rovnostranného trojúhelníku základny. Koncový efektor disponuje otvorem na umístění dotykového pera.

Horní rameno a paralelogram

Horní rameno Delta robota je jednoduchý kvádr s otvory umožňující připevnění k servo motoru a k rovnoběžníkovému mechanismu. K dosažení maximálního rozsahu teoretického sférického kloubu byl použit kulový čep používaný v RC modelech. Plastové oko pro táhlo disponuje závitem M3 (3 mm), díra v mosazném čepu je pro šrouby rozměru velikosti M3[26]. Zbytek paralelogramu je složen ze závitových tyčí též velikosti M3. Mechanismus

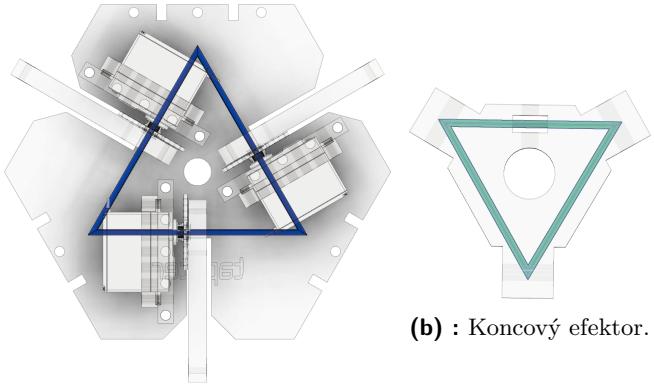
umožňuje dosáhnutí požadovaných pozic koncového efektoru, přičemž zachovává dostatečnou robustnost díky použitým kulovým čepům v kombinaci se závitovými tyčemi.



(a) : Pohled shora bez základny s detailem na referenční trojúhelníky.

Obrázek 3.1: CAD model Delta robota.

Obrázek 3.2: Kulový čep.



(a) : Základna se servomotory.

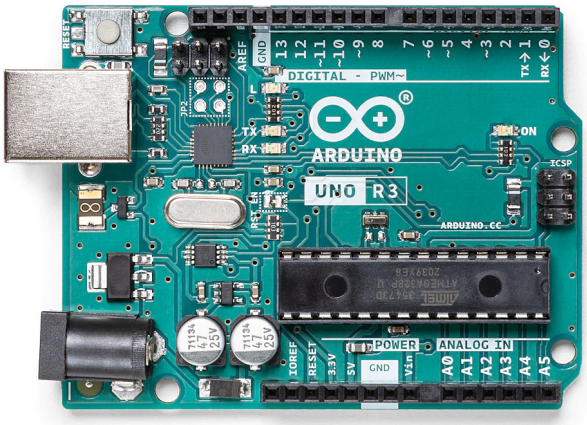
(b) : Koncový efektor.

Obrázek 3.3: Základna a koncový efektor.

3.2 Použitá elektronika

V této sekci se budeme zabývat základními elektronickými prvky, jež jsou využité v této práci. Pro jednoduchost designu bylo k ovládání servo motorů použito Arduino UNO, které je vhodné jednak díky své dostupnosti, tak kvůli hojné uživatelské základně.

3.2.1 Arduino UNO



Obrázek 3.4: Arduino UNO[7].

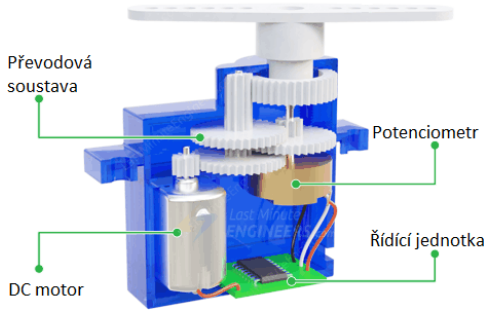
Arduino UNO je populární mikroprocesorová vývojová deska platformy Arduino založená na ATmega328P. Tento mikroprocesor běží na 8-bit AVR architektuře, která často nabízí balanc mezi výpočetním výkonem a nárokem na nízkou spotřebu. Základní parametry Arduino UNO jsou[7]:

- Pracovní napětí: 5 V.

- Vstupní napětí (doporučeno): 7-12 V.
- Vstupní napětí (limity): 6-20 V.
- Maximální proudový odběr: 200 mA.
- Digitální I/O piny: 14 (z nichž 6 poskytuje výstup s šířkovou modulací (PWM)).
- Analogové vstupní piny: 6.
- Stejnosměrný proud na I/O pin: 20 mA.
- Stejnosměrný proud pro 3,3V pin: 50 mA.
- Paměť Flash: 32 KB (ATmega328P), z nichž 0,5 KB je použito bootloaderem.
- SRAM: 2 KB (ATmega328P).
- EEPROM: 1 KB (ATmega328P).
- Krystal: 16 MHz.

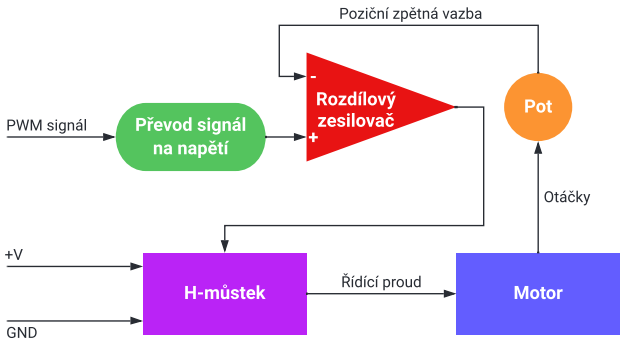
Arduino UNO umožňuje práci s vstupními analogovými piny, vstupně-výstupními digitálními piny, PWM výstupem, seriovou (UART), SPI a I2C komunikací. Tyto vlastnosti umožňují použití vysoké škály senzorů, aktuátorů, serv a dalších elektronických komponent. Nejznámější vývojové prostředí pro tuto platformu je Arduino Integrated Development Environment (IDE)[27], jež je založené na programovacím jazyku C++ s velkým množstvím rozšiřujících knihoven. IDE umožňuje uživatelsky jednoduché vytváření programů, kompilování a následné nahrání do mikroprocesoru. IDE též obsahuje mnoho uživateli vytvořených knihoven, jež usnadňují práci se senzory, komunikačními protokoly a mnoho dalšími, čímž nadále rozšiřuje možnosti aplikace platformy Arduino. V případě Delta robota bude sloužit ke zpracování příkazů řídícího systému a následnému ovládání serv.

3.2.2 Servo motor



Obrázek 3.5: Struktura modelářského servo motoru [8].

V této sekci se budeme zabývat modelářskými rotačními servo motory. Jde o malé, mikroprocesorem jednoduše ovladatelné zařízení, jež má spolu s Arduinem mnoho uplatnění v robotice, DIY projektech či automatizačních aplikacích. Skládají se z malého elektrického motorku, převodního systému složeného z ozubených kol, zabudované řídící jednotky a potenciometru. Potenciometr je propojen s výstupem převodového systému a zajišťuje poziční zpětnou vazbu do řídící jednotky. Schéma řídící jednotky pozorujeme na obrázku 3.6. Řídící jednotka přijímá referenční signál, jež převede na napětí. Toto referenční napětí reprezentuje referenční pozici servo motoru. Referenční pozice je v záporné zpětné vazbě porovnána s aktuální pozicí servo motoru a koriguje řídící proud potřebný pro dosažení referenční pozice. Jedná se o uzavřený zpětnovazební řídící systém, který využívá negativní zpětnou vazbu k řízení směru otáčení motoru k dosažení zadané polohy.



Obrázek 3.6: Schéma řídící jednotky [8].

Servo je ovládáno řídícím PWM signálem a očekává pulz každých 20 ms, neboli frekvence řídícího signálu je 50 Hz.

- Šířka pulsu 1 ms a kratší otočí servo na pozici 0°.
- Šířka pulsu 1.5 ms otočí servo na pozici 90°.
- Šířka pulsu 2 ms a delší otočí servo na pozici 180°.

Puls o šířce, jež je v rozsahu 1 ms až 2 ms, je proporciální k pozičnímu rozsahu serva. Je nutné podotknout, že každý výrobce může mít tento vztah mezi šířkou pulsu a pozicí odlišný, avšak výše uvedené je nečastější, se kterými se lze běžně setkat.

Servo motor MG996R

Pro Delta robota budou využity 3 servo motory MG996R. Parametry servo motorů[28]:

- Napájecí napětí: 4,8 - 7,2 V.
- Rychlost: 0,17 s/60°při 4,8 V, 0,14 s/60°při 6 V.

- Točivý moment: 9,4 kg-cm při 4,8 V, 11 kg-cm při 6 V.
- Operační proud: 500 - 900 mA při 6 V.
- Maximální proudový odběr: 2,5 A při 6 V.
- Operační rozsah: 180°.

3.2.3 Napájecí zdroj

Pro napájení použité elektroniky poslouží jakýkoliv spínaný, či laboratorní zdroj s výstupním napětím 4,8 - 7,2 V, který zvládne poskytnout dostatečný proud (maximální proudový odběr elektroniky při 6 V je 7700 mA).

Kapitola 4

Implementace aplikace

V této kapitole se budeme zabývat řídící aplikací. Na základě požadavků navrhneme architekturu. Aplikace se skládá z jednotlivých komponent - Klient, Server a Robot. Komponenty se skládají z modulů, jejichž funkcionality se detailně popíšeme.

4.1 Architektura řídící aplikace

V této sekci si popíšeme motivaci za architekturou aplikace, nastíníme si princip využitých konfiguračních souborů a zjednodušeně si uvedeme rozdělení do komponent.

4.1.1 Motivace

Architektura aplikace musí brát ohled na integrování do simulační metody HiL. K docílení tohoto požadavku a oddělení potřebných funkcionalit si rozdělíme celou aplikaci do tří dílčích komponent - Klient, Server a Robot. Klientská komponenta bude zpracovávat veškeré uživatelské požadavky a ve formě příkazů pro robota je bude odesílat na server. Tato klientská komponenta bude též sloužit jako rozhraní mezi prostředím HiL a robotem, kdy HiL vytvoří instanci klienta a bude komunikovat se serverem. Rozdělení na server a klient je především kvůli nasazení robota mimo simulační počítač, kdy se tyto dvě zařízení mohou nacházet ve větších vzdálenostech. Přenos dat přes ethernet je v takovém případě ideálním řešením. Komponenta serveru bude obsahovat hlavní řídící modul Delta robota. Tento modul bude zpracovávat klientské požadavky a provádět veškeré kinematické výpočty pro jejich interpretaci robotem. Komponenta Robot bude využívat fyzického mikroprocesorového počítače Arduino pouze jako prostředek pro převod teoretických kloubových souřadnic na fyzický pohyb servo motorů.

Cílem je vytvořit základ pro aplikaci, jež na základě vstupních konfiguračních souborů bude schopna pomocí fyzického modelu Delta robota interagovat s mobilním zařízením a podávat uživateli zpětnou vazbu o provedených operací. Konfigurační soubory budou popisovat operace potřebné k obsluze mobilního telefonu, fyzické parametry Delta robota a převodní vztah umožňující robotu interagovat s mobilním telefonem. Programovací jazyk pro klientskou

a serverovou komponentu bude Python. Komponenta Robota, která obsahuje Arduino je napsána v přidruženém Arduino IDE. Aplikace tedy nebude vázaná na specifický model mobilního telefonu a Delta robota, ale umožňuje obecné nasazení.

4.1.2 Konfigurační soubory

Při spuštění aplikace je třeba konfiguračních souborů obsahující potřebná inicializační data pro jednotlivé komponenty. Konfigurační soubory jsou ve formátu JSON. Pro aplikaci jsou vytvořeny dva typy konfiguračních souborů. První popisují fyzické parametry robota, konfiguraci mobilního telefonu v souřadném systému robota a komunikační port. Toto jsou minimální potřebné soubory pro chod aplikace.

Druhý typ jsou soubory popisující operace potřebné k ovládání mobilního telefonu, což je reprezentace tlačítek, klávesnic, gest a sekvencí. Každá z těchto operací dodržuje svůj specifický formát nesoucí všechny potřebné informace k jejímu vykonání. Příklad reprezentace všech těchto operací v JSON tvaru jsou k nalezení v příloze B.

4.1.3 Požadované funkcionality

Hlavním cílem robota je schopnost ovládat mobilní telefon a zvládat jednoduché operace, které se s tím pojí. Takové operace mohou být klikání na určité aplikace, imitace přejetí prstem po obrazovce, či psaní na klávesnici. K interpretaci těchto operací si vytvoříme datové struktury, které jsou intuitivní a zároveň strojově lehce interpretovalené ve formě konfiguračních souborů.

Tlačítka

Jako tlačítko budeme považovat určitý bod na obrazovce, který je možno stisknout. Reprezentovat jej budeme jménem a souřadnicemi středu daného tlačítka.

Klávesnice

Na obrázku 4.1 pozorujeme příklad, jak může vypadat klávesnice mobilního zařízení. Jednotlivé klávesy lze definovat pozicí jejich středu x a y od dolního levého rohu mobilního zařízení (viz sekce Interpretace dotykového zařízení). Klávesnice nadále obsahuje jednotlivé vrstvy, mezi kterými lze proklikávat. Například z počáteční vrstvy klávesnice (4.1a) se lze dostat do vrstvy se speciálními znaky (4.1b) stisknutím levého dolního tlačítka *123* a naopak stisknutím tlačítka *ABC*. Celou klávesnici si budeme reprezentovat jako struktura obsahující všechny vrstvy klávesnice. Každá vrstva obsahuje pozice jednotlivých kláves a pozice kláves umožňující přechod do sousedních vrstev klávesnice.

Jednotlivé vrstvy si následně intepretujeme jako vrcholy grafu a pozici kláves umožňující přechod do jiné vrstvy jako hrany grafu. Tímto dostaneme

způsob, jak z jednoduchého uživatelského vstupu skládajícího se z posloupnosti znaků a znalosti počáteční vrstvy vytvořit sekvenci příkazů pro řídící modul robota umožňující napsání všech znaků s přechody mezi jednotlivými vrstvami.

(a) : Počáteční vrstva klávesnice. **(b) :** Vrstva se speciálními znaky.

Obrázek 4.1: Různé vrstvy klávesnice mobilního zařízení.

Gesta

Gesta slouží reprezentaci operacích simulující užívání mobilního zařízení. Mezi gesta řadíme kliknutí na tlačítko, kliknutí na pozici xy, přejetí prstem po obrazovce mezi dvěma body a napsání posloupnosti znaků na klávesnici. Pro každý typ gesta je vytvořena vlastní struktura nesoucí potřebné informace k vykonání daného gesta.

Sekvence gest

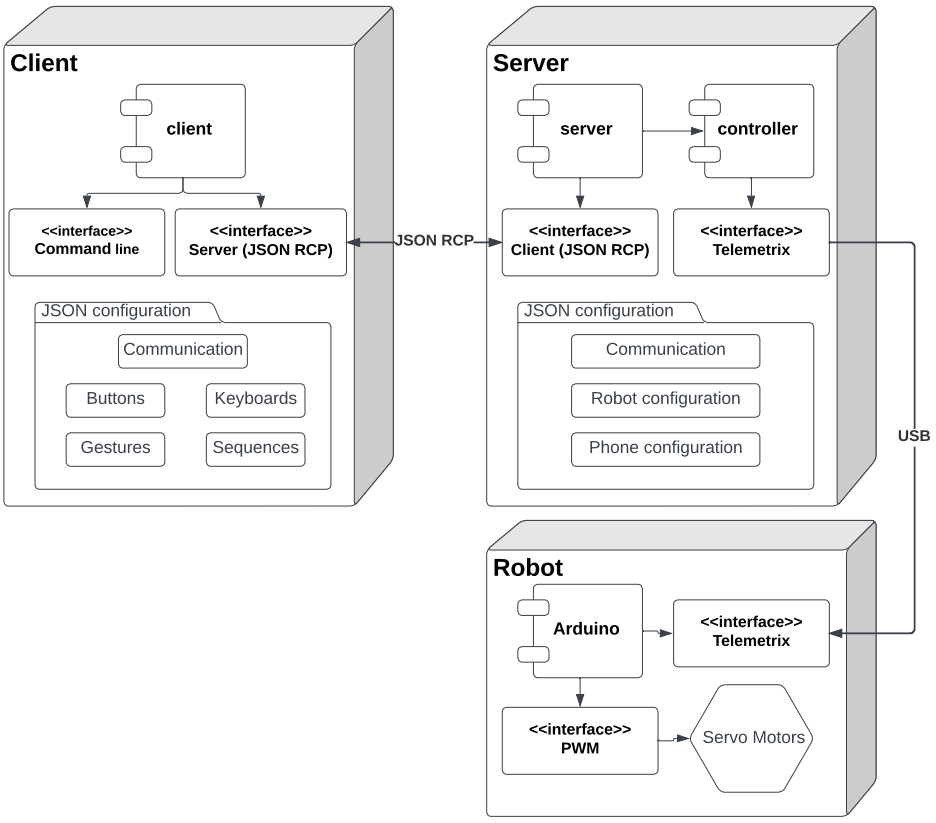
Jako sekvence budeme reprezentovat posloupnosti jednotlivých gest poskytují uživateli možnost tvoření složitějších operacích a jejich uložení jako jednotný celek. Sekvence jsou reprezentovány celkem čtyřmi parametry - typem, jménem, čekajícím časem po dokončení sekvence a počtem jejich opakování. Typy jsou dvojího druhu. První umožňuje využití již vytvořených gest. Druhý typ umožňuje vytvoření lokálního gesta pouze pro danou sekvence přímo při inicializaci. Obě tyto reprezentace v jedné sekvenci je možno pozorovat v příloze B.5.

4.1.4 Rozdělení komponent

Jelikož je celá aplikace psaná v anglickém jazyce, jednotlivé obrázky se schémata v této kapitole budou též v anglickém jazyce kvůli zachování přehlednosti a soudržnosti se zdrojovým kódem. Překlad do češtiny by byl v tomto případě mohl být matoucí a nepřesný.

Na obrázku 4.2 zjednodušené schéma popisující architekturu aplikace. Celá aplikace se skládá ze tří klíčových komponent specifikující se své funkcionality. Komponenta klienta zajišťuje načtení konfiguračních souborů, interpretaci vstupu od uživatele přes příkazovou řádku, zaslání požadavku na server a intepretaci odpovědi serveru. Komunikace mezi klientem a serverem probíhá přes komunikační protokol JSON-RPC. Komponenta serveru obsahuje hlavní řídící modul Delta robota. Server na základě klientských požadavků spouští dostupné funkce řídícího modulu a jejich návratové hodnoty vrací klientovy.

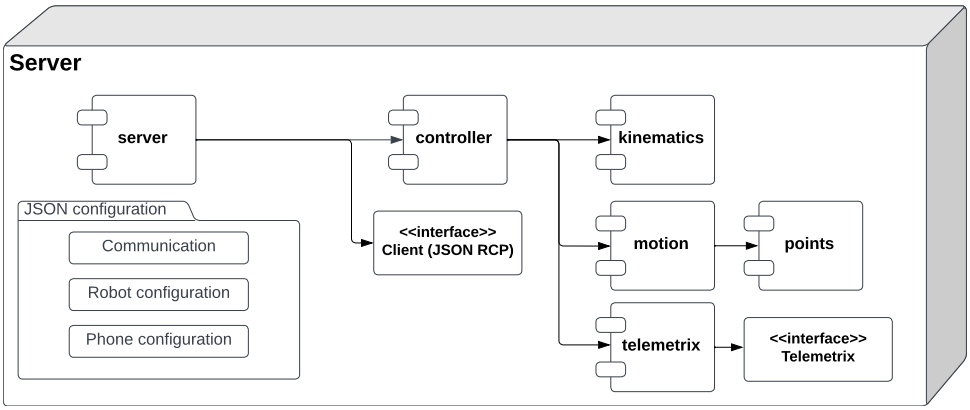
Toto rozdělení je voleno především kvůli budoucímu zakomponování do simulační metody HiL. HiL si vždy vytvoří klientské spojení se serverem a bude klást požadavky pro pohyb robota. Tyto požadavky mohou být jednotlivá gesta, či sekvence. Po splnění požadavku klient zanikne spolu se spojením. Avšak komponenta serveru ovládající robota zůstane aktivní a bude vyčkávat na další spojení. Komponenta Robota reprezentuje samotný fyzický model řízený Arduinem. Spojení mezi řídícím modulem a Arduinem probíhá s využitím komunikační knihovny Telemetrix. Veškeré kinematické výpočty probíhají v Serverové komponentě v řídícím modulu, který následně zasílá požadované kloubové souřadnice Arduinu. Arduino následně převádí kloubové souřadnice na PWM signál, který posílá servo motorům, které pohnou rameny na požadované pozice. Po vykonání požadavku vygeneruje řídící modul zpětnou zprávu pro klienta, kterou server odešle jako návratovou hodnotu.



Obrázek 4.2: Zjednodušené schéma architektury programu.

4.2 Komponenta Server

Na obrázku 4.3 je schéma komponenty **Server**. Komponenta se skládá z dílčích modulů specifikující se na jednotlivé úlohy. Komponenta disponuje dvěma komunikačními rozhraními (interface). První z nich obsahuje modul ***server*** ke komunikaci s klientem přes JSON-RPC protokol. Druhé komunikační rozhraní



Obrázek 4.3: Celé schéma komponenty **Server**.

je zabudováno v modulu ***telemetrix***, které obsluhuje řídící modul ***controller***. Dála jsou zde moduly ***kinematics***, ***motion*** a ***points*** poskytující potřebné funkcionality řídícímu modulu. Funkčnost Modulů si popíšeme v následujících sekcích. Posledním důležitým prvek komponenty jsou konfigurační soubory popisující komunikaci, konfiguraci robota a konfiguraci mobilního zařízení v souřadném systému robota.

4.2.1 Modul *points*

Tento modul obsahuje třídy reprezentující body v prostoru a nimi tvořené sekvence bodů. Bod si reprezentujeme jako třídu *Point*, jejíž atributy jsou souřadnice x, y a z. Sekvenci bodů si reprezentujeme jako třídu *Points*, která je zkonstruována z bodů a její atributy poskytují potřebné funkcionality pro práci se sekvencemi bodů.

4.2.2 Modul *motion*

Modul ***motion*** poskytuje funkcionality pro vytvoření sekvence trojrozměrných bodů. Pro tyto funkcionality obsahuje modul dvě třídy *EasingFunctions* a *Motion*. Třída *EasingFunctions* poskytuje takzvané *"Easing functions"*, což jsou funkce určující míru rychlosti změny parametru v čase[29]. Tyto funkce si můžeme rozdělit do následujících skupin:

- *linear*: Lineární proporce mezi časem a změnou.
- *easeInQuad*, *easeInCubic*, *easeInQuart*, *easeInQuint*: Zrychlení od nulové rychlosti s různými mírami zrychlení.
- *easeOutQuad*, *easeOutCubic*, *easeOutQuart*, *easeOutQuint*: Zpomalení k nulové rychlosti s různými mírami zpomalení.
- *easeInOutQuad*, *easeInOutCubic*, *easeInOutQuart*, *easeInOutQuint*: Zrychlení do poloviny, pak zpomalení, s různými mírami.

Třída *Motion* poskytuje metody potřebné k vytvoření sekvence bodů v prostoru. Body a jejich sekvence jsou reprezentovány třídami modulu ***Point***. Důležité metody třídy:

4.2.3 Modul *kinematics*

Modul obsahuje třídu *Kinematics*, která je zkonstruována se vstupními kinematickými parametry Delta robota, na jejichž základě poskytuje funkce pro výpočet inverzní kinematické úlohy. Algoritmický výpočet dopředné kinematické úlohy nebyl pro robota potřebný, avšak třída je uzpůsobena na případné rozšíření.

4.2.4 Modul *telemetrix*

Tento modul obsahuje třídu *Telemetrix* umožňující komunikaci mezi vývojovým prostředím Python a Arduinem za užití USB. Tato třída využívá stejnojmenného komunikačního protokolu. Z této třídy využíváme pouze část implementovaných metod, které slouží k inicializaci komunikace s Arduinem a následnému ovládání servo motorů.

4.2.5 Modul *controller*

Hlavní řízení robota zajišťuje třída *Tapster* řídícího modulu ***controller***. Na základě vstupních konfiguračních souboru obsahující potřebné informace o Delta robotu a konfiguraci mobilního zařízení v souřadném systému robota je tato inicializována třída. Při konstruování třída naváže spojení s Arduinem, inicializuje servo motory a přesune koncový efektor do domovské pozice. Po této inicializaci nastává zpracovávání klientských příkazů.

Metody této třídy si rozdělíme do dvou kategorií - interní a veřejné. Interní metody slouží ke kinematickým výpočtům, transformací mezi souřadnými systémy, komunikaci s Arduinem a vhodné interpretaci veřejných metod. Veřejné metody řídící modul poskytuje serveru, které jsou volatelné klientem. Tyto metody slouží k usnadnění ovládání robota a jeho interakce s mobilním zařízením.

Metoda *_move_through_points*

Tato metoda slouží k hlavnímu ovládání samotného robota. Jejím vstupem je posloupnost bodů, jimiž má projít koncový efektor robota. Pro posloupnost bodů je vypočítána inverzní kinematická úloha. Pokud všechny body leží v pracovním prostoru a tedy pro všechny body má inverzní kinematická úloha řešení, je získána posloupnost kloubových souřadnic θ_i. Tyto souřadnice jsou následně zaslány mikrokontroléru Arduino, s využitím třídy *Telemetrix*. Návratová hodnota metody je finální pozice koncového efektoru. Pokud nějaký ze vstupních bodů ležel mimo pracovní prostor, je vrácena návratová hodnota reprezentující chybovou hlášku.

Veřejné metody

Hlavní veřejné metody jsou lze charakterizovat do dvou skupin. Skupiny se rozlišují souřadným systémem, ve kterém jsou vstupní body reprezentovány. Metody intepretující body v souřadném systému mobilního zařízení obsahují prefix *"phone"*. Přehled všech interních a veřejných metod i s jejich kompletním předpisem je dostupný v příloze C.1. Zde si uvedeme nejdůležitější veřejné metody, jejichž vstupem jsou body reprezentované v souřadném systému robota:

- *go* - metoda sloužící pro pohyb koncového efektoru na požadovanou pozici xyz.
- *tap* - metoda sloužící pro kliknutí na požadované pozici xyz.
- *swipe* - metoda sloužící pro přejetí koncovým efektorem mezi dvěma.
- *arc* - metoda sloužící pro přejetí koncového efektoru po kruhové trajektorii.

Pro zmíněné metody existují jejich protějšky přijímající body vyjádřené v souřadném systému mobilního zařízení. Tyto metody využijou transformační matice pro transformaci vstupních bodů do souřadného systému robota.

4.2.6 Kalibrace mobilního zařízení

Pro kalibraci mobilního zařízení je využito speciální třídy umožňující užití řídícího modulu pro manuální ovládání polohy koncového efektoru s využitím klávesových zkratek. Při inicializaci je zadáno jméno kalibrovaného zařízení a následně jsou nalezeny rohové body obrazovky, tím že se jich dotkne dotykové pero. Výsledek této operace je uložen ve formě konfiguračního souboru.

Interpretace trajektorie

Pro účely testovacího modelu je zde trajektorie pohybu reprezentována velice jednoduchým způsobem. Ač by bylo možné implementovat složité řídící algoritmy pro plánování pohybu v prostoru, v tomto případě to nebylo nutné a nebudeme se touto problematikou zabývat. Trajektorii pro pohyb koncového efektoru budeme reprezentovat posloupností lineárně rozdělených bodů v prostoru. Jelikož použité servo motory neposkytují zpětnou vazbu o jejich reálné poloze není zde příliš prostoru pro návrh řídících algoritmů.

4.2.7 Modul *server*

Tento modul obsahuje třídu *Server* obstaravající chod na bázi JSON-RPC serveru. Toho je dosaženo využitím Python knihovy *ajsonrpc*[30]. Tato knihovna implementuje odlehčenou verzi asynchronního serveru užívající komunikačního protokolu JSON-RPC 2.0. Ač tento princip umožňuje asynchronní chování a komunikaci s více klienty zároveň, v našem případě se vždy bude jednat

o jednoho připojeného klienta, kde komunikace bude mít sériový charakter. Server je inicializován s konfiguračními soubory na jejichž základě vytvoří objekt řídícího modulu. Řídící modul poskytuje serveru metody potřebné řízení robota. K této implementaci serveru jsou potřebné dva důležité objekty knihovny *ajsonrpc* - *Dispatcher* a *ResponseManager*.

Dispatcher

Dispatcher je objekt typu *Dictionary*[31] (český termín může být slovník, či hashovací tabulka), který mapuje jména jednotlivých metod a jejich aliasy. Jako alias budeme chápat vytvoření odkazu na danou metodu. *Dispatcher* může obsahovat třídy, objekty, funkce i metody. V našem případě bude obsahovat veřejné metody třídy *Tapster* řídícího modulu a doplňující metody třídy *Server*. Pro odlišení metod jednotlivých tříd doplňuje *Dispatcher* jména jednotlivých metod prefixem, který je jméno třídy, ze které metoda pochází.

ResponseManager

ResponseManager má za hlavní cíl generovat odpověď na požadavek klienta. Tato úloha obnáší parsování požadavku, ošetření chybových situací a starání se o paralelní exekuci paralelních požadavků. K exekuci požadavků využívá metody poskytnuté objektem *Dispatcher*. Pokud přijde požadavek se jménem metody, kterou disponuje *Dispatcher*, je metoda zavolána se vstupními parametry klientského požadavku. Pokud je zavolání metody úspěšné *ResponseManager* vytvoří odpověď serveru na základě návratové hodnoty volané metody a tuto odpověď odešle zpět klientovi. Pokud se tato operace nevydaří, je vygenerována příslušná chybová hláška, která je odeslána klientovi.

4.3 Princip komunikace užitím protokolu JSON-RPC

V sekci popisující modul ***client*** se budeme zabývat správnou tvorbou klientských příkazů, které se následně zasílají na server. K tomuto tématu si nyní ukážeme, jaké parametry musí komunikační zprávy splňovat. Komunikace probíhá zasíláním příkazů formátu JSON dodržující specifickou strukturu. Průběh komunikace si ukážeme na specifickém příkladu, kdy klient bude chtít zavolat metodu $go(x, y, z, speed)$ se vstupními parametry $(x = 10, y = 10, z = -100, speed = 20)$. Jak již bylo řečeno v předchozí sekci popisující modul ***server***, metodu zprostředkovává *Dispatcher*, který na základě jména metody poskytne alias na samotnou metodu. V příloze B.1 lze vidět obecný předpis formátu klientského požadavku a odpovědi serveru.

Pro náš konkrétní případ je klientský požadavek zobrazený na obrázku 4.4 (všimněme si jména samotné metody doplněného o prefix třídy, která jej obsahuje). Takovýto požadavek přijme *ResponeManager* a pokud *Dispatcher* poskytuje metodu s příslušným názvem, poskytne alias na metodu, která je zavolána se vstupními parametry. *ResponseManager* se stará a správné

přiřazení vstupních parametrů, popřípadě o vytvoření příslušné chybové hlášky, pokud dojde k nějaké chybě. Z tohoto uvedené lze usoudit, že klient musí mít znalost o názvu dostupných metod spolu s jejími parametry. Způsob vyřešení této problematiky si popíšeme v následující sekci zaměřené na popis modulu ***client***. Po zavolání je vytvořena odpověď serveru splňující příslušný formát (příloha B.2), který je zaslán zpět klientovi.

```
{
    "jsonrpc" : "2.0"
    "method": "tapster.go"
    "params" : {
        "x" : 10,
        "y" : 10,
        "z" : -100,
        "speed" : 20}
    "id" : 1
}
```

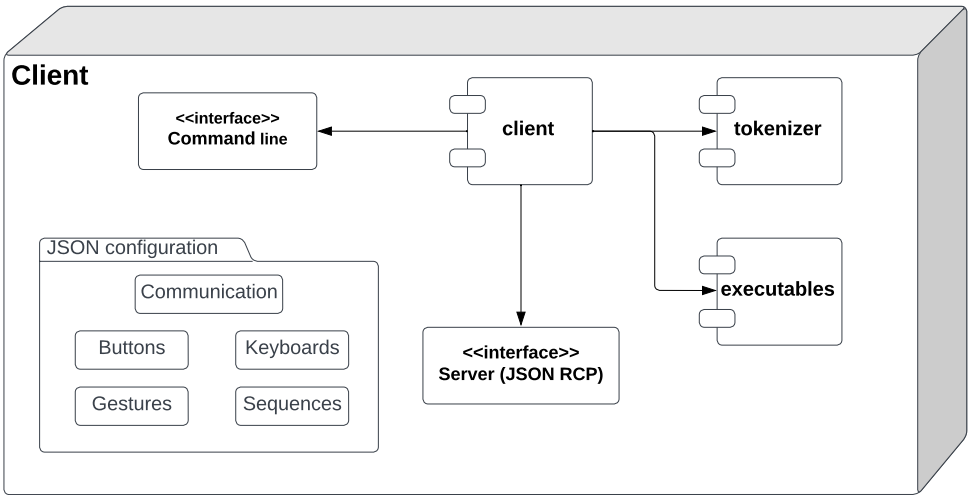
Obrázek 4.4: Příklad klientského požadavku.

4.4 Komponenta Client

Na obrázku 4.6 je schéma komponenty **Client**. Komponenta se též skládá z dílčích modulů specifikujících se na jednotlivé úlohy. Modul ***client*** disponuje dvěma komunikačními rozhraními (interface). První slouží k zpracování uživatelských vstupnů, druhé pro komunikaci se serverem užitím JSON-RPC protokolu. Pro správnou interpretaci konfiguračních souborů popisující interakci robota s mobilním zařízením slouží modul ***executables***. Modul ***tokenizer*** poskytuje funkcionality ke správnému zpracovaní uživatelských vstupů z příkazové řádky. V následujících sekcích si popíšeme detailní funkčnost modulů tvořící komponentu **Client**.

4.4.1 Modul *tokenizer*

Modul ***tokenizer*** obsahuje třídu *Token* a *Tokenizer*. Třída *Token* slouží k intepretaci takzvaných tokenů. Token reprezentuje jednotlivé prvky textového řetězce, jako jsou čísla, speciální znaky, či písmena. Třída *Tokenizer* poskytuje metody k ověření, jestli dva textové řetězce mají stejnou charakteristiku. Jeden textový řetězec reprezentuje uživatelský vstup z příkazové řádky, druhý samotný předpis metody (název spolu s parametry). Tokenizací textového řetězce rozumíme jeho rozdělení na dílčí elementy odlišného typu. Tento princip slouží k porovnání, zda uživatelský vstup (textový řetězec) koreluje s existující metodou poskytnutou serverem a zda je zde dodržen stejný počet parametrů. Pokud jsou tyto podmínky splněny, třída navrací jméno metody spolu s jejími parametry ve formátu, ze kterého lze rovnou sestrojit klientský



Obrázek 4.5: Celé schéma komponenty **Client**.

požadavek. Příklad zpracování uživatelského vstupu spolu s předpisem funkce je k nalezení v příloze C.2.

4.4.2 Modul *executables*

Tento modul slouží ke zpracování konfiguračních souborů reprezentující tlačítka, klavésnice, gesta a sekvence. Tlačítka a klávesnice jsou uloženy pouze jako datové struktury pro další zpracování. Pro gesta a sekvence jsou vytvořeny sady příkazů pro pohyb robota, které reprezentují danou operaci. Tyto příkazy, které jsou reprezentovány ve formě klientských JSON-RPC požadavků, jsou složené z veřejných metod třídy *Tapster* řídícího modulu ***controller***. Jelikož jsou tyto metody jasně definované a neměnné, pro vytvoření příkazů není potřeba vzájemné interakce tříd modulů. Návratové hodnoty tříd modulu ***executables*** jsou sady JSON-RPC příkazů.

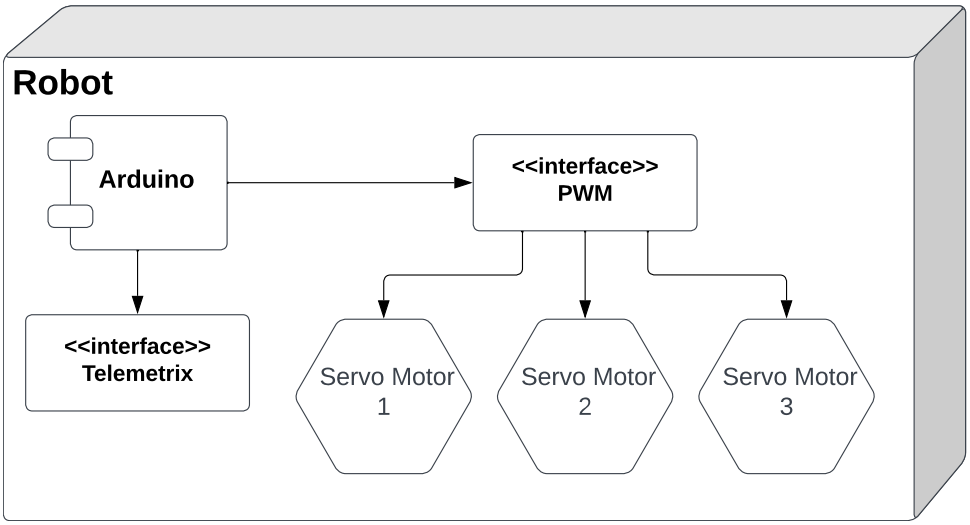
4.4.3 Modul *client*

Klientský modul ***client*** se skládá ze třídy *Client* a třídy *ParserToClient*. Třída *Client* využívá Python knihovny *jsonrpcclient*[32] k vytvoření klientského spojení se serverem. Toto spojení funguje na principu technologie *websocket*[33]. Při inicializaci využije *Client* funcionalit modulu ***executables*** a pro konfigurační soubory popisující operace mobilního telefonu vytvoří příslušné JSON-RPC příkazy. Tyto příkazy jsou uloženy ve formě atributů třídy pro pozdější využití. Pro správné parsování uživatelských vstupů potřebuje klient znalost dostupných funkcí serveru spolu se znalostí jejich předpisu. Při inicializaci klient zavolá metodu serveru poskytující tento seznam předpisu dostupných funkcí (neboli funkcí, jimiž disponuje *Dispatcher*). Tyto předpisy si též uloží jako atributy k dalšímu zpracování. Po inicializaci je *Client* připraven podávat požadavky serveru. Tato třída bude základem pro integraci do simulační metody HiL.

Pro samotnou obsluhu třídy *Client* slouží třída *ParserToClient*. Tato třída se stará o správné interpretovaní uživatelského vstupu přes příkazovou řádku a s využitím metod třídy *Client* je zasílá ve formě požadavků na server. Tento vstup může být dvojího charakteru. Uživat může přímo volat metody dostupné na serveru, či exekuovat uložené gesta a sekvence. K rozpoznání, zda je volaná funkce ve správném tvaru slouží modul ***tokenizer***. Pokud se jedná o přímé volání metody serveru, z uživatelského vstupu je vytvořený JSON-RPC požadavek a je třídou *Client* odeslán na server. V případě exekuování uloženého gesta, či sekvence jsou JSON-RPC požadavky již vytvořeny a jsou v řadě za sebou posílány na server. Při této operaci se vždy čeká na dokončení probíhajícího požadavku.

4.5 Komponenta Robot

Komponenta Robot obsahuje pouze modul ***Arduino***, který obstarává obsluhu fyzických serv za pomocí mikrokontroléru Ardino Uno. Podotkněme zde, že modul máme zadefinovaný jakožto název pro soubory s koncovkou ".py"(programovací jazyk Python), avšak program Arduina je v programovacím jazyce C++. Technicky vzato tedy nejde o modul, avšak principiálně zde není rozdíl oproti jiným modulům a kvůli zachování přehlednosti budeme ***Arduino*** nazývat modulem. Podstatnou část celé programu Arduina tvoří knihovna *Telemetrix4Arduino*. Tato knihovna umožňuje Arduinu serverové chování, neboli na základě požadavků ze strany Pythonu Arduino volá své metody a funkce. Řídící signál pro servo motory je poskytnut digitálními PWM piny Arduina. Komunikace přes USB s Pythonem je zajištěna komunikačním protokolem Telemetrix.



Obrázek 4.6: Celé schéma komponenty **Robot**.

Ovládání servo motorů

K ovládání servo motorů je využita knihobna *VarSpeedServo*[34] poskytující rozšiřující funkcionality, než základní knihovna pro ovládání serv. Užité servo motory, popsané v kapitole *Servo motor*, mají svojí vlastní řídící jednotku přijímající referenční vstupní signál. Tento princip má své výhody i nevýhody. Výhoda takto tvořených servo motorů je snadnost jejich využití, kde po poskytnutí referenčního PWM signálu se servo motory sami dostanou do požadované pozice. Toto je vhodné pro vývojové zařízení, kdy je cílem otestovat určitý princip a ne se zabývat vývojem řídící jednotky pro motor. S touto jednoduchostí se pojí nevýhody těchto servo motorů - absence znalosti aktuální polohy a nemožnost ovlivnit jejich rychlost pohybu. Pokud tedy dochází k výchylce, řídící jednotka servo motoru se jí snaží zkorigovat, avšak pokud servo motor je stále lehce vychýlen, Arduino nemá možnost tento fakt zjistit. Toto může být stěžejní důvod nepřesnosti složitých robotických systémů.

Kvůli výše zmíněným problémům využíváme knihovny *VarSpeedServo*, která poskytuje "pseudo" vyřešení problému s nemožností ovládat rychlost servo motorů. Slovo "pseudo" je využito, jelikož nemožnost ovládat rychlost servo motorů zůstala, avšak knihovna s využitím interuptů rozdělí dráhu mezi dvěma polohami na malé časové okamžiky, se kterými postupně mění referenční PWM signál. Tímto způsobe je možno dosáhnout variabilní rychlosti servo motoru, jejíž princip spoléhá, že se servo motor ihned dostává do požadovaných poloh. Tato knihovna též poskytuje možnost asynchroní pohybu více servo motorů najednou, což je stěžejní v našem případě paralelního robota.

Kapitola 5

Experiment a výsledky

Tato kapitola je zaměřena na experimentální ověření námi navržené aplikace sloužící pro reprezentaci lidské interakce s mobilním zařízením s využitím Delta robota. Cílem je ověření navržených principů, které reprezentují a usnadňují fyzickou interakci se zařízením. Mezi tyto principy náleží gesta a sekvence, jenž byly navrhnuty v předchozí kapitole. K fyzické interakci s mobilním zařízením bude využito fyzického modelu navrženého Delta robota. Přestože hlavní úlohou robota je přesné reprezentování požadovaných úloh, nebudeme se zaměřovat podrobnou analýzou jeho přesnosti. Vzhledem ke konstrukci robota a jeho použité elektronice nejsou očekávány dokonalé výsledky, avšak přesnost zařízení by měla být dostačující pro ověření celého konceptu, zda lze automatizovat ovládání mobilního zařízení při systémových testech v automotive. V rámci experimentální úlohy jsme provedli jednotlivé fyzické testy, které budeme analyzovat. Video záznamy těchto testů jsou v souborech přiložených k této práci.

Struktura kapitoly je rozdělena do několika částí. Nejdříve si popíšeme jednotlivé testy a nastíníme si jejich průběh. Následně se zaměříme na výsledky těchto testů a provedeme jejich analýzu. Další část je věnována diskuzi o samotném robotu, jeho přesnosti a možných zlepšeních. Nakonec shrneme dosažené výsledky celého experimentu.

5.1 Uspořádání experimentu

K fyzickému testování zde bude sloužit aparatura skládající se ze samotného Delta robota a mobilního zařízení. Delta robot je fyzické sestrojení námi navrženého CAD modelu, kde paralelogram je složený z modelářských kulových čepů v kombinaci se závitovými tyčemi. Základna, horní ramena a držáky servo motorů jsou vytisknuté na 3D tiskárně z materiálu PLA. Koncový efektor je osazen dotykovým perem, které zprostředkovává interakci. Mobilní zařízení je zasazeno v nehybném podstavci pod základnou Delta robota. Při osazení dotykovým perem se poloha bodu dotyku pera a obrazovky nachází pod středem rovnostranného trojúhelníku koncového efektoru. Avšak inverzní kinematická úloha je vztažena ke středu tohoto trojúhelníku. Jelikož tento rozdíl je pouze ve výškové souřadnici z těchto dvou bodů, nedochází zde narušení funkčnosti kinematických principů. Důsledek je, že fyzické umístění

obrazovky mobilního zařízení je o něco níže, než je vyjádřeno v souřadném systému robota, což nemá vliv na hlavní cíl testování. Nesmíme zapomenout, že při reálném nasazení robota je potřeba zakomponovat tuto skutečnost do kinematiky.

Při experimentu budeme simulovat obecné nasazení robota, kdy je mobilní zařízení vloženo do pracovního prostoru robota, jsou nalezeny okrajové body obrazovky, na jejichž základě je vytvořena transformační matice pro převod souřadnic ze souřadného systému obrazovky, do souřadného systému robota a následně robot je připraven na uživatelské příkazy. Teoreticky by poloha a natočení obrazovky mobilního zařízení měla být flexibilní vůči souřadnému systému robota, tudíž budeme testovat tento scénář, kdy nejsou dány přesné vzdálenosti určující umístění. Jediné nutné kritérium je, aby obrazovka celou svou plochou ležela v pracovním prostoru robota.

5.1.1 Kalibrace umístění mobilního zařízení

Pro získání transformačního vztahu mezi obrazovkou mobilního zařízení a samotným robotem je potřeba zjistit umístění rohových souřadnic obrazovky. Tohoto je docíleno s využitím funkcionalit kalibračního modulu umožňující manuální ovládání polohy koncového efektoru osazeného dotykovým perem za pomocí klávesových zkratek. Před spuštěním celé aplikace složené z klientské a serverové komponenty, je třeba provést tuto manuální kalibraci. Konfigurační soubor je následně načten řídícím modulem, který vytvoří transformační matici.

5.2 Testovací úlohy

Pro přehlednost v testovacích úlohách si zavedeme značení **T***i*, kde *i* reprezentuje pořadí úlohy. Budeme testovat schopnost robota splnit gesta a následné sekvence tvořené gesty. Následně ověříme, zda rozsah robota dosahuje teoretických hodnot a je schopen ovládat obrazovku o rozměru 11 palců.

- **T1 - Gesto psaní na klávesnici - lehké**
  Cílem této úlohy je otestovat jednoduché gesto reprezentující napsání jednoduché slova *"hello"*, kde všechny jednotlivé znaky leží v jedné vrstvě klávesnice. Test **T1** lze pozorovat na videu s názvem *"T1-write-single.MOV*.

- **T2 - Gesto psaní na klávesnici - složité**
  Tato úloha má za cíl otestovat správnou interpretaci psaní znaků na klávesnici u nichž je třeba přepínat mezi jednotlivými vrstvami, které obsahují daný znak. Jako vstup této úlohy budeme považovat textový řetězec *"[h2e$llo]"*. Na základě znalosti počáteční vrstvy a konfigurace klávesnice jsou vytvořeny příkazy pro správné napsání těchto znaků. Video záznam testu **T2** je soubor s názvem *"T2-write-advanced.MOV"*.

- **T3 - Sekvence - jednoduchá**
  Testovaná sekvence se skládá ze tří za sebou jdoucích gest. Gesta jsou kliknutí na aplikaci poznámky, návrat na domovskou obrazovku a následné přejíždění mezi domovskými obrazovkami. Video záznam testu **T3** je soubor s názvem *"T3-seq-simple.MOV"*.

- **T4 - Sekvence - složitá**
  Poslední testovaná sekvence se skládá ze z kombinace předchozích testů dohromady. Cílem je tedy úspěšně otevřít aplikaci poznámky, napsat textový řetězec *"[(hel$lo)]"*, vrátit se na domovskou obrazovku a následně se přesunout mezi domovskými obrazovkami. Video záznam testu **T4** je soubor s názvem *"T4-seq-advanced.MOV"*.

- **T5 - Manipulovatelnost s obrazovkou o rozměru 11"**
  Cílem toho testu je ověřit, že robot je schopen dosáhnout všech rohů obrazovky o rozměrech 11 palců. Jako referenční mobilní zařízení bude sloužit tablet s obrazovkou o rozměrech stran 22.3×16.7 cm. Video záznam testu **T5** je soubor s názvem *"T5-11-inch-screen.MOV"*.

5.2.1 Výsledky testovacích úloh

První testovací úloha **T1** měla za cíl otestovat gesto psaní jednoduchého slova na klávesnici. Na základě exekuce gesta s využitím klientského modulu byly správně vytvoření JSON-RPC požadavky reprezentující jednotlivé operace, které byly následně zpracovány řídícím modulem. Tato operace byla úspěšná a robot byl schopen napsat slovo na klávesnici.

Cílem úlohy **T2** bylo ověřit funkčnost robota automaticky přecházet mezi jednotlivými vrstvami klávesnice. Modul ***executables*** na základě znalosti počáteční vrstvy a vlastností klávesnice (poloha kláves v souřadném systému mobilního zařízení) vytvořil sadu příkazů potřebné pro napsání textového řetězce *"[h2e$llo]"*. První vrstva obsahovala znaky "*h*", "*e*", "*l*", "*o*", druhá vrstva obsahovala znak "2" a poslední vrstva obsahovala znaky "[", "]" a "$". Přechody mezi jednotlivými vrstvami byly uskutečněny pomocí znalosti pozice kláves pro přechod do vedlejších vrstev. Příkazy byly úspěšně vytvořeny a robot napsal tuto sekvenci znaků.

Testovací úloha **T3** měla za cíl otestovat intepretaci sekvence složené z jednoduchých gest. Použitá gesta jsou otevření aplikace, vrácení se na domovskou obrazovkou a přejížděními mezi jednotlivými obrazovkami. Tyto gesta simulují běžné obsluhování mobilního zařízení, které byly úspěšně splněny.

Úloha **T4** se zaměřuje na zkombinování všech předešlých gest dohromady. Tímto jsme schopni nasimulovat reálné využití mobilního zařízení. Jednotlivá gesta jsou otevření poznámkové aplikace, napsaní složitého textového řetězce *"[(hel$lo)]"*, opuštění poznámkové aplikace a přejíždění mezi domovskými obrazovkami. Při otevření poznámkové aplikace je počáteční vrstva klávesnice se znaky s velkým písmem, se kterými textový řetězec nezačíná. Modul zpracovávající vytvoření příkazů pro robota tento problém správně zaznamenal

a robot napíše správný textový řetězec spolu vykonáním ostatních gest.

V testovací úloze **T5** byl zkoumán rozsah robota s obrazovkou o velikosti 11 palců. Robot se byl schopen dostat do všech bodů obrazovky, tudíž splnil předpokládaný rozsah. Ale bylo zde patrné, že u krajních bodů obrazovky nedosahoval robot požadovaných přesností. Více o přednosti robota si zmíníme v následující sekci.

5.3 Fyzický model Delta robota

Při vývoji robotických zařízení se zřídka kdy povede navrhnout model, který bude dokonale fungovat od prvního nasazení. Již při zprovozňování fyzického modelu se vyskytly nedostatky, které měly za důsledek zhoršení přesnosti samotného robota. Je nutno podotknout, že i přes tyto nedostatky robot posloužil k úspěšnému otestování principů aplikace.

5.3.1 Elektronické nedostatky

Mikrokontrolér Arduino úspěšně posloužil při nasazení v této úloze. Komunikace s řídícím modulem byla bezproblémová a Arduino úspěšně ovládalo servo motory. Avšak volba servo motorů je kritickým nedostatkem tohoto modelu Delta robota. Tyto hobby servo motory se díky svým vlastnostem hodí pro jednoduché aplikace, kde není vysoký nárok na přesnost, či není potřeba složitého řízení motoru. Avšak v případě použití k ovládání ramen robota zde vyšly na povrch nedostatky ohledně přesnosti dodržení referenční polohy. I malá výchylka servo motoru může být zdrojem nedosažení požadované pozice koncového efektoru. A přesně tento případ nastal, ač servo motory disponují kovovým převodovým systémem, vůle není dostatečná a motor disponuje vždy malou odchylkou. Tato odchylka experimentálním určením dosahuje do jednoho stupně úhlové odchylky v záležitosti na konkrétním servomotoru.

Druhý nedostatek servo motorů byla absence měření skutečné polohy motoru. Servo motor se vždy snaží dosáhnout referenční polohy reprezentované PWM signálem v co nejkratším čase užitím integrované řídící jednotky. Avšak kdy a s jakou přesností je tato poloha dosažena není řídícímu modulu ***controller*** známo. A tento fakt činí obrovský nedostatek, jelikož ze není možno implementovat složitější řídící algoritmy, či ověřovat reálnou polohu motoru.

5.3.2 Konstrukční nedostatky

Konstrukce paralelogramu tvořená závitovými tyčemi spolu s kulovými čepy splnila svůj účel, který byl zamezení rotačního pohybu koncového efektoru. Při dosahování krajních pozic pracovního prostoru začaly rozsah limitovat fyzická konstrukce kulových čepů, které ač mohou ve velké míře reprezentovat sférický kloub, pro krajní pozice dochází k fyzickému omezení rozsahu samotného čepu.

Avšak největší konstrukční nedostatek souvisel se servo motory. Rameno, které je přidělané k motoru, má náchylnost na nežádoucí rotaci, kvůli pouze

jednomu styčnému bodu s motorem. Základna spolu se způsobem uchycení motoru není dostatečně robustní a dochází zde ke nechtěným prohýbáním.

5.3.3 Přesnost sestrojeného modelu

Výše zmíněné nedostatky měly nezanedbatelný vliv na přesnost samotného robota. Robot splnil svůj původní záměr reprezentovat lidského operátora při ovládání mobilního zařízení. Jelikož jsou jednotlivé ovládací prvky moderních zařízení konstruované na interakci s lidským prstem, je zde prostor pro nepřesnosti. Jako příklad si můžeme uvést nejmenší prvek, který měl robot ovládat, a to klávesu na klávesnici mobilního zařízení, která má typický rozměr 5×7 mm. Je zde poměrně velká plocha pro stisk, do které se robot musí trefit. Jak již bylo řečeno v úvodu kapitoly, hlavním cílem experimentu nebylo udělat kompletní rozbor vlastností robota. Při testování vyplynulo, že by ani takovýto rozbor neměl smysl, neboť majoritním důvodem nepřesností byly konstrukční a elektronické nedostatky. Z tohoto důvodu nebyla prováděna žádná metoda kalibrace, jelikož její vliv by zmíněné nedostatky nenahradil. Účel robota byl pouze jako prvotní testovací zařízení, které slouží k ověření konceptu. Přesnost byla dostačující pro ovládání obrazovky zařízení, ale je zde prostor pro zlepšení.

Experimentálním ověřením jsme zjistili, že odchylka od požadovaných prostorových souřadnic koncového efektoru se zvyšovala při přiblížení k okrajové části pracovního prostoru. V těchto polohách byl pohyb koncového efektoru silně nekonzistentní. Při středu pracovního prostoru byl manipulátor dostatečně přesný pro ovládání mobilního zařízení, avšak i zde byla pozorována nekonzistentnost pohybu.

5.4 Shrnutí výsledků testů

Experimentální úlohy nám ověřily zkoumaný koncept. Interakce lidského operátora mobilního zařízení lze nahradit Delta robotem. Pro ovládání robota byla vyvinuta aplikace, která byla rozdělena do dílčích funcionálních komponent. Princip tvorby gest a sekvencí pomocí konfiguračních souborů se osvědčil a fungoval dle očekávání. Gesta a sekvence byly úspěšně exekuovány s využitím fyzického modelu navrhnutého Delta robota. Fyzický model úspěšně posloužil ke svému účelu, bohužel jeho přesnost nebyla dostačující pro budoucí nasazení. Nejmarkantnější vliv na nepřesnost robota měly použité servo motory, které nedosahovaly dostatečné přesnosti a neumožňovaly zpětnou vazbu ohledně aktuální polohy. Avšak tento vývojový krok přinesl cennou zkušenost při návrhu robotického zařízení, která bude zužitkována v dalších vývojových etapách.

Kapitola 6

Závěr

Úvodem práce bylo představení problematiky týkající se systémového testování v automobilovém průmyslu. Byla představena simulační metoda HiL a s ní role mobilních zařízení a jejich využití s automobily. Pro automatizaci interakce s mobilním zařízením bude využit Delta robot. Robot zde slouží k nahrazení lidského operátora při systémového testování.

V teoretické části je seznámeno kinematickým modelem Delta robota a jeho důležitými parametry. Model slouží k nalezení kinematických rovnic. Na základě těchto rovnic je sestavena inverzní a dopředná kinematická úloha. Pro interakci robota s mobilním zařízením je potřeba vytvořit vztah charakterizující vzájemné umístění. Finální teoretický úkon je analýza vlivu kinematických parametrů na pracovní prostor robota. Tato analýza poskytne potřebné parametry k sestrojení robota, který bude dostatečný rozsah pro manipulaci s řadou zařízení o rozdílných velikostech. Je zde rozvedeno jaká metodologie vedla k nalezení vhodných kinematických parametrů.

Po nalezení parametrů je vytvořen CAD model, na jehož základě je vytvořena fyzická verze robota. V této kapitole je popsán samotný model spolu s popisem jednotlivých prvků robota. Robot je osazen mikropočítačem Arduino, který se stará o ovládání servomotorů, přičemž oba tyto elektronické díly jsou náležitě popsány.

Důležitou součástí pro ovládání robota je jeho řídící aplikace. Cílem je navrhnout a vytvořit aplikaci v programovacím jazyce Python, u jejíž architektury bude brán ohled na budoucí integraci do simulační metody HiL. Aplikace je rozdělena do dílčích komponent - Klient, Server a Robot. Každá komponenta je složena z modulů, které jsou charakteristické svými specifickými funkcionalitami. Základní operace umožňující ovládání mobilního zařízení si reprezentujeme ve formě jednoduchých a přehledných konfiguračních souborů. Tento postup byl zvolen pro diverzifikaci použití a jednoduchému strojovému tvoření operací. Účelem komponenty Klient je intepretace těchto souborů spolu s uživatelským vstupem z příkazové řádky. Komponenta server obsahuje moduly s potřebnými funkcionalitami pro řízení robota, mezi které patří kinematické výpočty, komunikace s mikroprocesorem, či správná interpretace klientských příkazů. Komponenta Robot obsahuje mikroprocesor Arduino spolu se servo motory zprostředkovávajíc fyzický pohyb robota.

V závěrečné kapitole bude experimentálně ověřena funkčnost aplikace a

navrhnutého robota. Testy úspěšně ověří celý koncept ovládání mobilního zařízení s využitím Delta robota. Principy aplikace fungují, avšak u navrhnutého robota bylo naraženo na jeho limity. Analýzou fyzického modelu robota byly nalezeny příčiny nedostatečné přesnosti robota, mezi které patří nevhodné servo motory spolu s konstrukčními nedokonalostmi.

Tato práce přináší obohacení v kinematickém základu Delta robota spolu s vizualizací jeho pracovního prostoru. Vizualizace umožňuje pochopení vlivů jednotlivých kinematických parametrů na rozsahové možnosti robota. Vytvořená aplikace byla úspěšně otestována a bude sloužit pro integraci se simulátorem HiL. Navrhnutá architektura aplikace umožňuje obecné nasazení, kdy veškeré proměnné prvky, mezi které patří samotný robot, užité mobilní zařízení, či parametry gest a sekvencí jsou reprezentovány ve formě konfiguračních souborů. Lze tedy vytvořit oddělené projekty s různými harwarovými revizemi robotů, které mohou být nezávisle na sobě využity v simulátoru HiL. Navržený robot poskytl cenné zkušenosti, které budou zhodnoceny při dalším vývoji.

6.1 Budoucí rozšíření

Hlavní aspekt budoucnosti tohoto projektu sestavení nové generace fyzického modelu, která nebude disponovat momentálními nedostatky. Majoritní vliv na přesnost robota má volba použitého servo motoru. Nový motor musí disponovat vysokou přesností při zátěži spolu s poskytnutím zpětné vazby o aktuální poloze do řídícího modulu. Přesnost motoru zaručí dosažení referenční pozice, znalost aktuální polohy motoru umožní tvorbu pokročilých řídících systémů, které pro reálné nasazení robota budou potřeba.

Pro ověření funkčnosti mobilní aplikace je potřeba zajistit zpětnou vazbu simulačnímu prostředí HiL. Cílem bude rozšířit robota o kamerový systém, který bude sloužit k validaci mobilní aplikace. Kamerový systém bude ověřovat, zda po vykonání dané operace nastane kýžený výsledek a mobilní aplikace se bude chovat podle očekávání. Tento systém bude integrován do komponenty Server, která bude tuto validaci zpracovávat.

Literatura

[1] CCB, "V-model." https://www.ccb.cz/images_aqua/2013/zari/09-Deloitte-02x.jpg, 2023. [Online; 21.05.2023].

[2] M. K. A. Yeshmukhametov, "Design and kinematics of serial-parallel hybrid mechanism with intersecting axes of rotation," 2021. [Online; accessed 27.04.2023].

[3] R. Clavel, "Device for the movement and positioning of an element in space." US Patent 4,976,582, 1990.

[4] J. Toquica, P. Oliveira, J. Motta, and D. Borges, "A proposal to solve the inverse kinematics problem of a parallel robot configuration with neural networks," 2018.

[5] P. Robert L. Williams II, "The delta parallel robot: Kinematics solutions." https://www.ohio.edu/mechanical-faculty/williams/html/PDF/DeltaKin.pdf, 2016.

[6] Tapsterbot, "Tapsterbot github repository." https://github.com/tapsterbot/tapsterbot, n.d. [Online; accessed 12.05.2023].

[7] Arduino, "Arduino uno rev3." https://store.arduino.cc/usa/arduino-uno-rev3, n.d. [Online; accessed 23.04.2023].

[8] L. M. Engineers, "Servo motor control with arduino." https://lastminuteengineers.com/servo-motor-arduino-tutorial/?utm_content=cmp-true, n.d. [Online; accessed 25.04.2023].

[9] R. Boot, J. Richert, H. Schutte, and A. Rukgauer, "Automated test of ecus in a hardware-in-the-loop simulation environment," in *Proceedings of the 1999 IEEE International Symposium on Computer Aided Control System Design (Cat. No.99TH8404)*, pp. 587–594, 1999.

[10] T. Acharya, "Black boxes and their intrusion." https://towardsdatascience.com/black-boxes-and-their-intrusion-620aa3c4c56b, 2023. [Online; 15.05.2023].

[11] "Python essays: The python programming language." https://www.python.org/doc/essays/blurb/, n.d. [Online; accessed 30.04.2023].

[12] Firmata, "Firmata protocol." https://github.com/firmata/protocol, n.d. [Online; accessed 3.05.2023].

[13] Mryslab, "Telemetrix: A python non-blocking event driven firmata client." https://mryslab.github.io/telemetrix/, n.d. [Online; accessed 6.5.2023].

[14] MrYsLab, "Telemetrix4arduino: An arduino firmware implementation of the telemetrix protocol." https://github.com/MrYsLab/Telemetrix4Arduino, n.d. [Online; accessed 6.05.2023].

[15] MrYsLab, "Telemetrix: A universal remote control protocol." https://htmlpreview.github.io/?https://github.com/MrYsLab/telemetrix/blob/master/html/telemetrix/index.html, n.d. [Online; accessed 6.05.2023].

[16] "Json: Javascript object notation." https://www.json.org/json-en.html, n.d. [Online; accessed 2.05.2023].

[17] "Json-rpc specification." https://www.jsonrpc.org/specification, n.d. [Online; accessed 2.05.2023].

[18] J. Merlet, *Parallel Robots*. Kluwer Academic Publisher, 2000.

[19] A. Pagala and H. A. Alizadeh, "Dynamic analysis of clavel's delta parallel robot," *ResearchGate*, 2012. [Online; accessed 27.04.2023].

[20] J. Bečka, "Konstrukce mechanismů." http://users.fs.cvut.cz/~beckajan/predn_design_KON_I_11.pdf. [Online; 16.05.2023].

[21] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Upper Saddle River, NJ: Pearson Prentice Hall, 3 ed., 2005.

[22] Plotly, "Plotly python graphing library." https://plotly.com/python/, n.d. [Online; accessed 13.05.2023].

[23] Coolblue, "Smartphone screens." https://www.coolblue.be/en/advice/smartphone-screens.html, n.d. [Online; accessed 13.05.2023].

[24] OpenStitching, "Largest interior rectangle." https://github.com/OpenStitching/lir, n.d. [Online; accessed 14.05.2023].

[25] Z. Marzeh, M. Tahmasbi, and N. Mirehi, "Algorithm for finding the largest inscribed rectangle in polygon," *Journal of Algorithms and Computation*, vol. 51, no. 1, pp. 29–41, 2019.

[26] P. Modelář, "Kulový čep v1 m3/3 krátký." https://www.peckamodel.cz/602456-kul-cep-m3-imbus-dl, 2023. [Online; accessed 14.05.2023].

[27] Arduino, "Arduino ide v2." https://docs.arduino.cc/software/ide-v2, n.d. [Online; accessed 23.04.2023].

[28] T. Pro, "Hs-311 servo datasheet." http://users.ece.utexas.edu/~valvano/Datasheets/ServoHS311.pdf. [Online; accessed 15.05.2023].

[29] Easings.net, "Easings functions cheat sheet." https://easings.net/, 2023. [Online; 17.05.2023].

[30] A. Pavlov, "ajsonrpc," 2023. [Online; 18.05.2023].

[31] Python, "Python tutorial: Data structures." https://docs.python.org/3/tutorial/datastructures.html, 2023. [Online; 18.05.2023].

[32] "jsonrpcclient." https://pypi.org/project/jsonrpcclient/, 2023. [Online; 18.05.2023].

[33] "The websocket api." https://websockets.spec.whatwg.org/, 2023. [Online; 18.05.2023.

[34] Netlabtoolkit, "Varspeedservo." https://github.com/netlabtoolkit/VarSpeedServo, 2023. [Online; 20.05.2023].

Příloha A

Obsah přiloženého FLASH disku

Přiložené soubory obsahují důležité soubory vytvořené pro potřeby této práce. Veškeré 3D grafy zobrazené v práci jsou uloženy ve formě interaktivních *.html* souborů. Dále jsou zde zdrojové kódy Python aplikace spolu se soubory tvořící 3D grafy. Zbývající část tvoří zdrojový latex kód práce spolu s videi zachycující provedené experimenty.

```
Tapster_Bachelor
├─ readme.txt - Popis obsahu FLASH disku.
├─ interactive_graphs - Složka s interaktivními grafy.
│  ├─ python_graphs - Složka s Python soubory generující interaktivními grafy.
│  └─ *.html - Interaktivní grafy.
├─ source_code - Složka se zdrojovými kódy aplikace.
│  ├─ config_files - Složka s JSON konfiguračními soubory.
│  └─ src - Zdrojové kódy aplikace.
├─ text - Složka s textem práce.
│  ├─ Tapster_Bachelor.pdf - Bakalářská práce v PDF formátu.
│  └─ tex - Zdrojový latex kód bakalářské práce
└─ videa - Složka s videi experimentu.
   └─ *.mov - Videa experimentu.
```

Příloha B

Konfigurační soubory

B.1 JSON-RCP

Struktura požadavku a odpovědi:

```
{
    "jsonrpc" : String specifikující verzi protokolu
    "method": String se jménem požadované metody
    "params" : Struktura obsahující parametry metody
    "id" : Identifikátor
}
```

Obrázek B.1: Struktura klientského požadavku.

```
{
    "jsonrpc" : String specifikující verzi protokolu
    "result": Návratová struktura úspěšného provedení metody
    "error" : Návratová struktura pokud nastane chyba
    "id" : Identifikátor
}
```

Obrázek B.2: Struktura odpovědi serveru.

B.2 Klávesnice

```
{
    "type": "keyboard",
    "name": "qwertz",
    "layers": {
        "qwertz": {
            "neighbor_layers": {
                "numbers": {
                    "x": 8,
                    "y": 17},
                "uppers" : {
                    "x": 3,
                    "y": 27}},
            "keys": {
                "q": {
                    "x": 0,
                    "y": 45},
                "w": {
                    "x": 8,
                    "y": 45 }}},
        "numbers": {
            "neighbor_layers": {
                "qwertz": {
                    "x": 8,
                    "y": 17}},
            "keys": {
                "1": {
                    "x": 0,
                    "y": 45},
                "2": {
                    "x": 8,
                    "y": 45}}},
        "uppers": {
            "neighbor_layers": {
                "qwertz": {
                    "x": 2,
                    "y": 28}},
            "keys": {
                "spec_0": {
                    "x": 0,
                    "y": 10},
                "spec_1": {
                    "x": 2,
                    "y": 10}}}}
}
```

Obrázek B.3: Příklad reprezentace klávesnice.

B.3 Tlačítka

```
{
    "type" : "button",
    "name": "home_button",
    "position" : {
        "x" : 22,
        "y" : 126}
}
```

Obrázek B.4: Příklad reprezentace tlačítka.

B.4 Gesta

```
{
    "type" : "gesture",
    "subtype" : "tap_button",
    "name" : "button_tap",
    "button_name" : "test_button",
    "number_of_taps": 1,
    "tap_duration" : 2,
    "wait_between_taps" : 2
}
```

Obrázek B.5: Příklad gesta zmáčknutí tlačítka.

```
{
    "type" : "gesture",
    "subtype" : "keyboard_write",
    "name" : "write_test",

    "keyboard_name": "qwertz",
    "starting_layer" : "qwertz",
    "to_write" : "[hello] {world}"
}
```

Obrázek B.6: Příklad gesta psaní na klávesnici.

```
{
    "type" : "gesture",
    "subtype" : "swipe",
    "name" : "middle_swipe",
    "duration" : 2,
    "parameters" : {
        "x" : 30,
        "y" : 30,
        "X" : 50,
        "Y" : 30,
        "swipe_speed": 40
    }
}
```

Obrázek B.7: Příklad gesta přejetí po obrazovce.

```
{
    "type" : "gesture",
    "subtype" : "tap_xy",
    "name" : "tap_XY",
    "number_of_taps": 1,
    "tap_duration" : 1,
    "wait_between_taps" : 1,
    "parameters": {
        "x" : 30,
        "y" : 40
    }
}
```

Obrázek B.8: Příklad gesta zmáčknutí pozice xy.

B.5 Sekvence

```
{
    "type" : "sequence",
    "name" : "test_seq",
    "execute": {
        "home" : {
            "type" : "execute_gesture",
            "name" : "home_screen_swipe",
            "wait" : 1,
            "repetition": 0},
        "write_test":{
            "type" : "create_gesture",
            "gesture_data":{
                "type" : "gesture",
                "subtype" : "keyboard_write",
                "name" : "write_test",
                "keyboard_name": "qwertz",
                "starting_layer" : "qwertz",
                "to_write" : "testing123"
            },
            "wait" : 1,
            "repetition": 0}},
    "sequence_order": ["write_tapster"]
}
```

Obrázek B.9: Příklad sekvence.

B.6 Konfigurační soubor robota

```
{
    "name": "Tapster mk1",
    "version": "1.0",
    "dimension": {
        "end_effector": 30,
        "top_triangle" : 128.65870756,
        "parallelogram_joint" : 217,
        "upper_joint" : 80 },
    "tap_depth" : 10,
    "servos": {
        "servo1": {
            "pin": 3,
            "servo_range_min" : 0,
            "servo_range_max": 150,
            "servo_offset": 50},
        "servo2": {
            "pin": 5,
            "servo_range_min" : 0,
            "servo_range_max": 150,
            "servo_offset": 50},
        "servo3": {
            "pin": 9,
            "servo_range_min" : 0,
            "servo_range_max": 150,
            "servo_offset": 51
            } }
    }
```

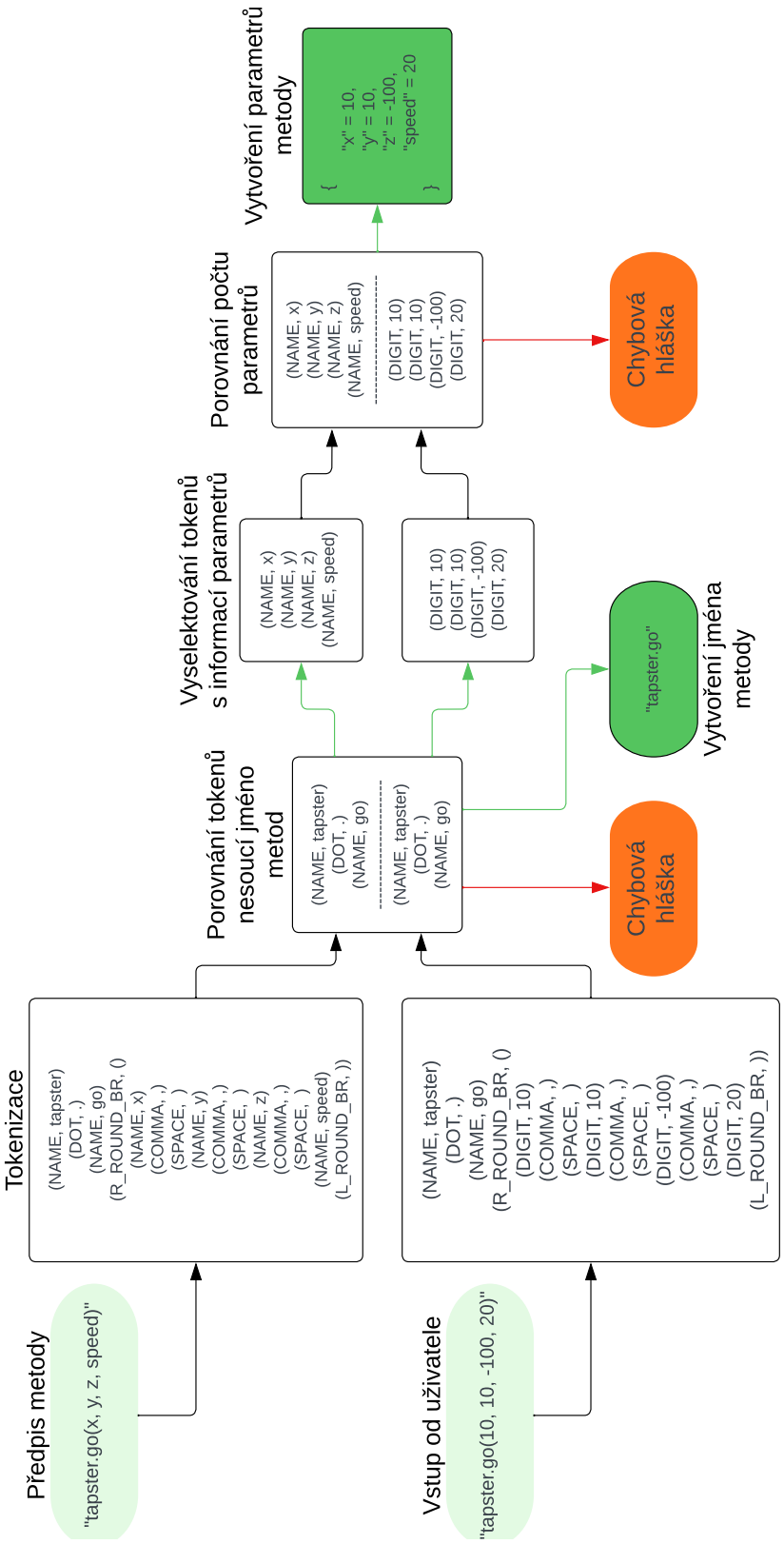
Obrázek B.10: Konfigurační soubor robota.

Příloha C

Softwarové reprezetnace

Tapster
self.config : DeltaRobotConfig self.tap_depth : int self.kinematics : Kinematics self.motion : Motion self.home_point : Point self.current_point : Point self.mov_speed : int self.offline_testing : bool self.servos : list[DeltaRobotConfig.servos] self.phone_configured : Bool self.T_matrix : np.darray self.inverse_T_matrix_operation : None
def get_method_names_signatures(cls) def get_method_names(cls) def start(self) def end(self) def home(self) def go(self, x,y,z, speed) def n_tap(self, x, y, z, n = 1, tap_duration = 0., wait_after_tap = 0.3, return_home = True) def tap(self, x, y, z) def swipe(self, x, y, z, X, Y, Z, slide_speed) def arc(self, center_x, center_y, center_z, radius, start_angle, end_angle, movement_speed) def load_ph_conf(self, phone_data : PhoneCalibData) def phone_go(self, x, y) def phone_tap(self, x, y) def phone_n_tap(self, x, y, n = 1, tap_duration = 0.1, wait_after_tap = 0.3, return_home = True) def phone_swipe(self, x, y, X, Y, swipe_speed) def phone_arc(self, center_x, center_y, radius, start_angle, end_angle, movement_speed) def phone_home(self) def get_current_position(self) def _check_and_offset_angles(self, angles: np.ndarray) def _move_through_points(self, points: Points, speed: int = 255,sleep_time :float = 0) def _calib_move_to(self, x, y, z) def _transform_coordinates(self, x, y, z) def _servos_init(self)

Obrázek C.1: Třída *Tapster* se svými atributy a metodami.



Obrázek C.2: Příklad zpracování požadavku třídou *Tokenizer*.