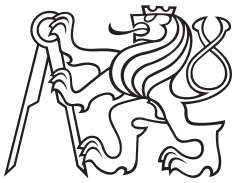
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Systém pro automatizované testování pomocí kolaborativních robotů

Minh Hoang Tran

Vedoucí práce: Ing. Vladimír Smutný, Ph.D.
Studijní program: Kybernetika a robotika
Květen 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Tran** Jméno: **Minh Hoang** Osobní číslo: **507395**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Systém pro automatizované testování pomocí kolaborativních robotů

Název bakalářské práce anglicky:

System for Automated Testing Using Collaborative Robots

Pokyny pro vypracování:

Cílem práce je vytvořit dálkově ovladatelný systém pro testování senzorů s minimální nutností interakce s manipulátorem.
1. Seznamte se s platformou Robot Operating System (ROS) pro řízení manipulátoru.
2. Seznamte se s požadavky zákazníka na testovací systém a navrhněte koncepci takového systému.
3. Navrhněte algoritmus pro efektivní otestování senzoru s využitím kolaborativního robotického manipulátoru. Cílem algoritmu je zajistit generování a vykonávání pohybů, včetně zaznamenávání výstupů senzorů o příslušné poloze nástroje.
4. Algoritmus implementujte a zdokumentujte.
5. Algoritmus otestujte a vyhodnoťte výsledky. Diskutujte možnosti dalšího vývoje a možnosti kompatibility řešení s robotickými manipulátory různých výrobců.

Seznam doporučené literatury:

[1] ROS 2 Documentation (https://docs.ros.org/en/iron/index.html)
[2] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, “Robot Operating System 2: Design, architecture, and uses in the wild,” Science Robotics vol. 7, May 2022.
[3] D. Casini, T. B. s, I. Lütkebohle, and B. B. Brandenburg, “Response-Time Analysis of ROS 2 Processing Chains Under Reservation-Based Scheduling,” In 31st Euromicro Conference on Real-Time Systems (ECRTS 2019), Dagstuhl, Germany, 2019, vol. 133, p. 6:1-6:23. doi: 10.4230/LIPIcs.ECRTS.2019.6.
[4] David Coleman; Ioan A. S, ucan; Sachin Chitta; Nikolaus Correll. „Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study“. In: Journal of Software Engineering for Robotics 5.1 (2014), p. 3–16.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Vladimír Smutný, Ph.D. robotické vnímání CIIRC

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

Datum zadání bakalářské práce: **25.01.2024** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **21.09.2025**

Ing. Vladimír Smutný, Ph.D.	prof. Dr. Ing. Jan Kybic	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 práce, Ing. Vladimíru Smutnému, Ph.D., za trpělivost a odborný dohled na mou práci. Dále bych chtěl společnosti IMA s.r.o za poskytnutou příležitost zde pracovat na této práci.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 23. května 2024

Minh Hoang Tran

Abstrakt

Tato práce se věnuje návrhu a implementaci knihovny pro automatizované testování a validaci vyvíjeného senzoru pomocí šestiosého robotického manipulátoru. Testování pomocí robotického manipulátoru umožňuje opakovatelnost testů a důkladné otestování senzorů za různých podmínek.

Cílem je navrhnout systém, který bude jednoduše integrovatelný do existujících systémů. Knihovna by měla poskytovat uživatelsky přívětivé rozhraní pro konfiguraci a programování testovacích scénářů. Spolehlivé provedení testů musí zahrnovat zpracování, synchronizaci a ukládání datových záznamů pro následnou analýzu signálů.

Použitý přístup zahrnuje návrh modulární knihovny, která umožňuje flexibilní konfiguraci. Knihovna je navržena tak, aby umožňovala snadnou adaptaci na nové senzory bez potřeby rozsáhlých úprav kódu. V rámci této práce je popsána architektura knihovny a její hlavní komponenty.

Současná knihovna splňuje požadavky zákazníka pro testování senzorů. Moduly však mohou být rozšířeny o další funkce, které zvýší robustnost a spolehlivost testů.

Klíčová slova: Automatizované testování, robotický manipulátor, senzor, knihovna, modularita

Vedoucí práce: Ing. Vladimír Smutný, Ph.D.

Abstract

This thesis focuses on the design and implementation of a library for automated testing and validation of a developing sensor using a six-axis robotic manipulator. Using a robotic manipulator allows repeatability of tests and thorough testing of sensors under various conditions.

The goal is to design a system that can be easily integrated into existing systems. The library should provide a user-friendly interface for configuring and programming test scenarios. Reliable execution of tests must include processing, synchronization, and storage of data records for subsequent signal analysis.

The approach used involves the design of a modular library that allows for flexible configuration. The library is designed to enable easy adaptation to new sensors without the need for extensive code modifications. This thesis describes the architecture of the library and its main components.

The current library meets the customer's requirements for sensor testing. However, the modules can be extended with additional features to enhance the robustness and reliability of the test.

Keywords: Automated testing, robotic manipulator, sensor, library, modularity

Title translation: System for automated testing with collaborative robots

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

1.1 Cíl práce

Cílem této práce je navrhnout a vytvořit efektivní systém pro ovládání šestiosého robotického manipulátoru, určeného pro testování a validaci vyvíjených senzorů. Systém by měl být dálkově ovladatelný s minimální nutností interakce s manipulátorem. Senzor je staticky uchycen, zatímco manipulátor bude vykonávat pohyby chapadlem v blízkosti snímaného prostoru senzoru. Tento systém bude schopen parametricky generovat pohyby manipulátoru a následně tyto pohyby vykonávat. Klíčovým úkolem systému je nejen provedení opakovatelných pohybů, ale také záznam parametrů pohybu a výstupů ze senzorů v reálném čase. To umožní detailní analýzu chování senzorů v různých podmínkách.

1.2 Motivace

Šestiosý robotický manipulátor, navržený pro testování senzorů, nabízí možnost simulace reálných scénářů a provádění rozsáhlých a časově náročných testů. Schopnost generovat parametrické pohyby a systematicky zaznamenávat výstupy umožní podrobné zkoumání chování senzorů v různých vzdálenostech a úhlech. Tato flexibilita je klíčová pro identifikaci senzorů, což je nezbytné pro jejich další vývoj a optimalizaci. Implementace takového systému přispěje k zefektivnění testovacího procesu, snížení nákladů a zvýšení kvality výsledných produktů.

Tento systém poskytne robustní rozhraní pro opakovatelné testování, který lze snadno přizpůsobit různým typům senzorů a specifickým požadavkům jednotlivých projektů.

Kapitola 2
Způsoby testování senzorů

V současné době hraje testování a validace senzorů klíčovou roli v mnoha oblastech průmyslu a výzkumu. Senzory se používají v široké škále aplikací, od automobilového průmyslu přes zdravotnictví až po robotiku. Přesnost, spolehlivost a opakovatelnost výsledků senzorů jsou zásadní pro jejich úspěšnou integraci v reálných podmínkách. Proto je nezbytné mít k dispozici efektivní, nástroj pro ověření konzistence a jejich měřících vlastností.

2.1 Manuální testování

Manuální testování senzoru pohybu je jedním z nejjednodušších a nejpřístupnějších způsobů testování senzorů. Tento proces zahrnuje přímou interakci operátorů se senzory, kde operátoři buď manipulují se samotnými senzory, nebo provádějí pohyby ve snímaném poli senzoru. Tímto způsobem jsou simulovány různé pohyby a podmínky, kterým mohou být senzory vystaveny v reálném provozu.

Tento přístup je užitečný při ověřování interakce senzorů s lidskými operátory, což umožňuje hodnotit, jak senzory reagují na skutečné lidské pohyby a chování. Jednoduchost manuálního testování spočívá v tom, že není zapotřebí žádných přidaných zařízení nebo nastavení pro provedení testů.

Nevýhodou je však neopakovatelnost měření. Přirozené lidské pohyby je obtížné opakovat. Toto vede k nekonzistentním výsledkům při opakovaných testech. Při vyskytnuté chybě je tedy obtížné identifikovat, zda se jedná o chybu v měření nebo chybu senzoru.

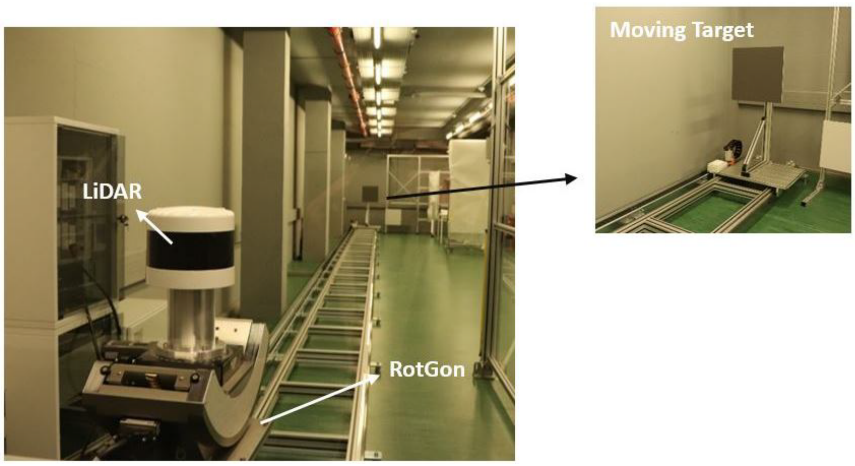
2.2 Pohyblivé pásy a kolejnice

Dalším způsobem je testování pomocí pohyblivého pásu nebo kolejnic [3], [14]. Tato metoda umožňuje provádění opakovatelných pohybů a simulaci různých podmínek. To je klíčové pro přesné a konzistentní testování senzorů, jelikož opakovatelnost pohybů zajišťuje, že každý test je prováděn za stejných podmínek. Toto výrazně zvyšuje spolehlivost a validitu získaných dat.

Výhody této metody zahrnují schopnost provádět řízené a opakovatelné pohyby, které jsou nezbytné pro důkladné testování senzorů. Pohyblivé pásy a

kolejnice mohou být nastaveny tak, aby simulovaly různé scénáře. Příkladem jsou konstantní rychlosti, zrychlení a zpomalení.

Nevýhodou použití pohyblivých pásových systémů může být jejich omezení v pohybu pouze v jedné ose. To znamená, že mohou poskytnout pouze translaci v jedné ose, což může být pro některé aplikace nedostatečné. To platí zejména pro aplikace, kde je třeba testovat senzory na schopnost detekce pohybů ve více směrech.



Obrázek 2.1: Platforma pro vyhodnocení a testování s goniometrickým rotačním systémem, LIDAR senzorem a kolejovým systémem. [3], obrázek 6.

2.3 Mobilní platformy

Mobilní platformy jsou vozidla nebo zařízení schopná pohybu po různých typech terénu. Mohou být vybaveny senzory pro detekci překážek, navigaci, měření vzdáleností a další účely. Mobilní platformy umožňují testování senzorů v různých prostředích, jako jsou lesy, pole, městské ulice, nebo dokonce interiéry budov.

Tyto platformy poskytují prostředí pro testování senzorů, které není možné replikovat v laboratorních podmínkách, naopak umožňují testování senzorů v dynamických a nepředvídatelných podmínkách. Tento typ testování také umožňuje zkoumat, jak senzory reagují na různé podmínky prostředí, jako jsou změny teplot, osvětlení nebo vlhkost vzduchu [9].

Nevýhodou testování v nepředvídatelných podmínkách je však obtížnost jejich opakování. Kvůli dynamické povaze prostředí, ve kterém se mobilní platformy pohybují, je těžké zajistit, aby každé testování proběhlo za stejných podmínek.

2.4 Robotické manipulátory

Robotické manipulátory poskytují další úroveň flexibility při testování senzorů. Tyto manipulátory umožňují simulaci různých podmínek, včetně pohybu ve více osách, různých natočení, rychlostí a zrychlení [5]. Při testování však nemusí být vždy nezbytné využití všech možných natočení, často postačuje simulace pohybu ve třech osách a aplikace různých rychlostí [1].

Robotické manipulátory mají schopnost simulovat pohyb rukou a nohou, která umožňuje provádět testování senzorů pohybů určených pro interakci s lidskými končetinami. Tato schopnost je obzvláště užitečná při vývoji senzorů, které musí přesně detekovat a reagovat na lidské pohyby. Manipulátory mohou simulovat různé scénáře, kterým mohou být senzory v reálných podmínkách vystaveny, a umožňují tak provádět komplexní testování.

Nevýhodou využití robotických manipulátorů může být jejich složitější instalace. Navíc manipulátory mohou být dražší a náročnější na údržbu než jednodušší testovací systémy.

2.5 Požadavky zákazníka

Testování senzorů pomocí robotických manipulátorů probíhá v laboratoři, kde je senzor umístěn v blízkosti manipulátoru. Při testování senzorů v laboratoři je nutné brát v úvahu, že se v tomto prostředí mohou vyskytovat a pohybovat další osoby. Z tohoto důvodu je nezbytné, aby byly zachovány kolaborativní funkce manipulátorů, které zajišťují bezpečnost a efektivní spolupráci mezi roboty a lidmi. Manipulátor je naprogramován tak, aby vykonával sérii předdefinovaných pohybů.

Během testování jsou zaznamenávány údaje ze senzoru společně s polohou robota, které budou sloužit k jeho následné analýze a ověření spolehlivosti.

Cílem práce je vytvořit knihovnu pro testování senzorů pomocí kolaborativního robota, která bude jednoduše integrovatelná do zbylých systémů zákazníka.

Důležitou součástí knihovny bude modul pro zpracování dat, který zajistí efektivní sběr naměřených hodnot ze senzoru a robota. Tento modul musí být schopen zpracovávat data ze senzoru a synchronizovat je s pohyby robota, aby bylo možné analyzovat spolehlivost senzoru.

Modul pro konfiguraci a programování testovacích scénářů by měl být uživatelsky přívětivý a umožnit rychlé a jednoduché nastavení testovacích parametrů.

Knihovna musí být navržena tak, aby byla robustní a spolehlivě vykonávala testovací scénáře.

Kapitola 3

Technické prostředky

3.1 Robotické manipulátory

Pro účely této práce byly poskytnuty kolaborativní roboti UR5, vyráběn společností Universal Robots A/S, a CRX-10iA od společnosti FANUC Corporation.

Kolaborativní roboti se oproti jiným manipulátorům liší svými vestavěnými senzory kloubových momentů. Díky těmto senzorům jsou kolaborativní roboti schopni detekovat kolize s jinými předměty v pracovním prostoru a přerušit pohyb. Tyto funkce zajišťují bezpečnost jak manipulátoru, tak i objektů v jeho dosahu. Kolaborativní roboti se využívají převážně pro práci s člověkem, montáž, balení, manipulace s materiály, testování, inspekce a podobně.

Pro tuto úlohu byl zákazníkem na základě jeho požadavků vybrán robot UR5. Robot poskytuje dostatečné klíčové vlastnosti pro požadovanou úlohu, kde není nutné dosahovat vyšších rychlostí než 1000 mm/s. Vzhledem k tomu, že v úloze není prováděna manipulace s předměty, nosnost není mezi klíčovými požadavky pro robota.

3.1.1 Universal Robots UR5

Universal Robots UR5 [13] je šestiosý kolaborativní robot vyvíjený dánskou společností Universal Robots A/S. Manipulátory UR5 jsou známy svou univerzálností a schopností provádět různé úlohy.

Klíčové parametry UR5

- Nosnost: 5 kg
- Dosah: 850 mm
- Maximální rychlost: 1000 mm/s
- Ovládací systém: Universal Robots Controller s platformou Polyscope verze 3.15
- Opakovatelnost pohybu: ±0,1 mm

Obrázek 3.1: Universal Robots UR5 [13]

3.2 Způsob řízení

Důležitou vlastností je i ovládání robota a jeho možnost vzdáleného ovládání s minimální interakcí uživatele s tabletem nebo robotem. Uživatelské knihovny pro vzdálené ovládání umožňují ovládání i za běhu programu robota. Toto umožnuje připojení extérních periférií, například kamer pro počítačové vidění a ovládání pohybu. Další výhodou uživatelských knihoven je monitorování a ukládání stavů robota, jako jsou polohy, rychlosti, teploty, síly a podobně.

3.2.1 Robot Operating System

Jendou z nejpoužívanějších metod ovládání robotů obecně je platforma Robot Operating System (ROS) [11]. Jedná se o open-source middleware platformu navrženou pro vývoj, řízení a komunikaci mezi robotickými systémy. Hlavními nástroji pro vývoj jsou jazyky C++ a Python.

Architektura ROS

- Jádro (Core/Master) je podstatná část frameworku odpovědná za správu a koordinaci běžících procesů (uzel, Node) a komunikaci mezi nimi. Funkcemi jádra je vytváření, ukončování a monitorování běžících procesů, poskytnutí mechanismů pro komunikaci mezi procesy a koordinaci akcí a událostí mezi různými komponentami.

- Komunikační systém umožňuje výměnu dat mezi různými částmi robotického systému. Základním modelem komunikace je model publish-subscribe, kde jeden proces publikuje data a ostatní procesy se mohou přihlásit k odběru těchto dat.

- Balíčky jsou autonomní jednotky systému obsahující algoritmy, ovladače nebo nástroje využívané robotem.

MoveIt

MoveIt [7] je rozsáhlá knihovna pro obě generace platformy ROS, která poskytuje robustní a flexibilní nástroje pro plánování pohybů a manipulaci pro roboty. Je navržena pro různé typy robotů, zejména pro manipulátory a mobilní roboty.

Knihovna poskytuje nástroje a algoritmy pro plánování optimálních trajektorií pohybů. Součástí MoveIt jsou flexibilní kinematické modely pro různé typy robotů, které zajišťují přesné určení polohy a orientace každého kloubu [2].

Jelikož MoveIt knihovna byla navržena pro více druhů robotů, využívá i mnoho různých plánovačů.

Za nejpoužívanější knihovnu se považuje Open Motion Planning Library (OPML), která primárně implementuje stochastické plánovací algoritmy, jako jsou Probabilistic Roadmap (RPM), Rapidly-exploring Random Tree (RRT), Expansive Space Trees (EST) a podobně [8]. MoveIt je přímo integrován s OMPL a využívá tuto knihovnu jako primární sadu plánovačů.

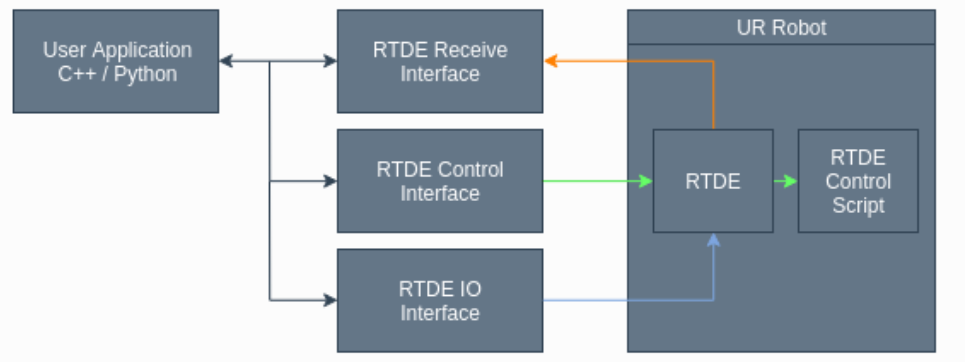
Další využívanou sadou plánovačů je Pilz Industrial Motion Planner, který je deterministický generátor pro kruhové a lineární pohyby. Plánovač umožnuje pohyby na sebe navazovat a tím vytvářet i komplexnější trajektorie.

Dále nabízí nástroje pro detekci kolizí a umožňuje vizualizaci plánů pohybů pomocí nástroje Rviz. To hraje klíčovou roli v bezpečnosti, jelikož dokáže předvídat kolize robota s okolím nebo se sebou samotným během plánování a před vykonáním pohybu.

3.2.2 UR RTDE

Real-Time Data Exchange (RTDE) [10] od Universal Robots představuje komunikační rozhraní, která umožnuje externím systémům vzdáleně ovládat a monitorovat robotický manipulátor. RTDE poskytuje rozhraní pro přenos informací v reálném čase a umožňuje řízení pohybů a současně přijímání aktuálních informací o jeho stavu. Komunikace je realizována zasíláním a přijímáním paketů pomocí TCP/IP protokolu, což zajištuje spolehlivý přenos dat.

UR RTDE [12] je knihovna vyvíjena univerzitou University of Southern Denmark. Je založena na jazyce C++ s využitím komunikačního rozhraní RTDE. Díky vazbám pro Python lze knihovnu využít i v aplikacích založených na jazyce Python. Mezi kompatibilními operačními systémy jsou Linux, Windows i MacOS.



Obrázek 3.2: Struktura knihovny UR RTDE [12]

UR RTDE poskytuje tři rozhraní.
RTDE Control slouží k primárně k ovládání pohybů a dalších funkcí, jako kontrolu dosažitelnosti poloh a podobně (zelené šipky v obrázku 3.2).
RTDE Receive slouží k přijímání informací o stavu robota, jako jsou na příklad proudy, momenty, polohy, rychlosti, konfigurace a podobně (oranžové šipky v obrázku 3.2).
RTDE IO slouží k asynchronnímu nastavení digitálních a analogových vstupních a výstupních periférií i za pohybu manipulátoru (modré šipky v obrázku 3.2) .

3.2.3 Výběr způsobu řízení

Společnost Universal Robots vyvíjí ovladače pro rozhraní ROS pro jimi vyráběné roboty. Ovladače jsou vyvíjené jak pro generaci ROS1, tak i ROS2 a obsahují integrovanou knihovnu MoveIt s využitím plánovačů OMPL.

Tyto ovladače závisí na specifických verzích firmware. Je tedy nutné prověřit kompatibilitu ovladačů s robotem a případně provést aktualizaci. Dále je doporučeno využití jádra reálného času (Real Time Kernel) s operačním systémem Ubuntu a přímého propojení počítače s robotem pomocí rozhraní Ethernet.

V úloze je robot však připojen k síti a není k němu připojen přímo žádný počítač. Toto zapříčinilo zpomalenou a nestabilní komunikaci mezi robotem a počítačem. Řešením tohoto byla úprava ovladačů a jejich následné sestavení a instalace. Zpomalená komunikace bohužel zapříčinila nestabilní pohyb robota.

V současné verzi ovladačů od vývojářů není také podporované Python API pro knihovnu MoveIt. Toto znemožňuje vývoj aplikací v jazyce Python a umožňuje ovládání robota pouze s využitím jazyka C++.

Preferencí je však jednotný systém všech vyvíjených aplikací, proto se upřednostňuje využití operačního systému Windows a programovacího jazyka Python.

Z těchto důvodů byla tedy pro současný projekt vybraná uživatelská knihovna UR RTDE.

3.3 Senzor

Vyvíjený senzor využívá čip MGC3140 pro snímání 3D gest a pohybů, založeným na technologii GestIC®, patentovanou společností Microchip Technology. MGC3140 je kapacitní dotykový kontrolér navržený tak, aby především snímal přirozené pohyby rukou a prstů na základě principů blízkého elektrického pole, s nízkou spotřebou energie. Čip je primárně využíván pro průmyslové a automobilové aplikace.

Klíčovými vlastnostmi MGC3140 jsou:

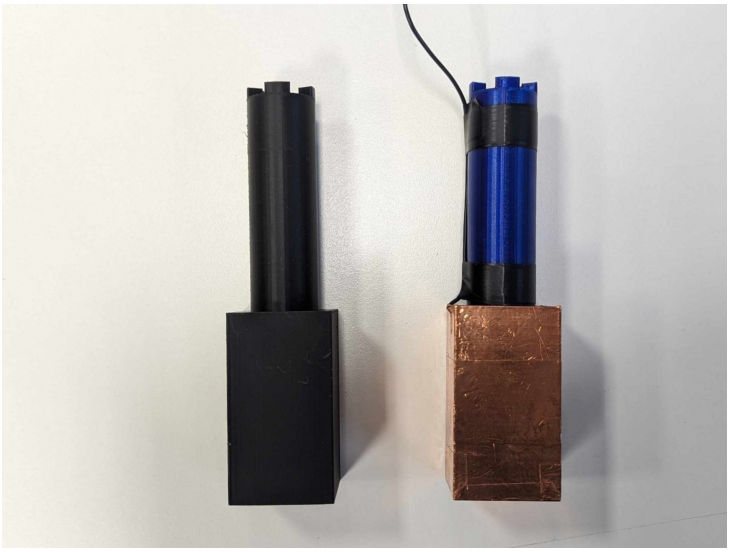
- Rozpoznávání gest rukou a vzdálenosti v osách X, Y, Z.
- Detekce vzdáleností a dotyků.
- Detekční rozsah: od 0 mm do 100 mm.
- Vyčítací frekvence: 200 vzorků za sekundu.
- Rozsah pracovních teplot: od -40 °C do 125 °C.

3.4 Chapadlo

V závislosti na konkrétním typu senzoru je klíčové vybrat chapadlo, které odpovídá požadovaným vlastnostem pro dané testování. Vlastnosti chapadla mají významný vliv na výsledky testování a spolehlivost získaných dat. V této úloze je chapadlo realizováno nehybným prodloužením ramene za posledním kloubem.

Pro testování výše zmiňovaného senzoru je zapotřebí, aby se vlastnosti chapadla blížily požadovaným vlastnostem lidské ruky. Výrobce čipu pro podobné potřeby doporučuje využití umělé ruky [4]. Tato umělá ruka je popsána jako pevná cihla z polystyrenu o rozměrech $40 \times 40 \times 70$ mm, která je obalena měděnou páskou o tloušťce 35 μm.

Umělá ruka byla zákazníkem provedena a experimentálně otestována pomocí nástavce vytištěného na FDM 3D tiskárně, z materiálu PETG, následně obaleným měděnou páskou podle návodů výrobce čipu. Uzemnění chapadla je realizované přes kostru robota.

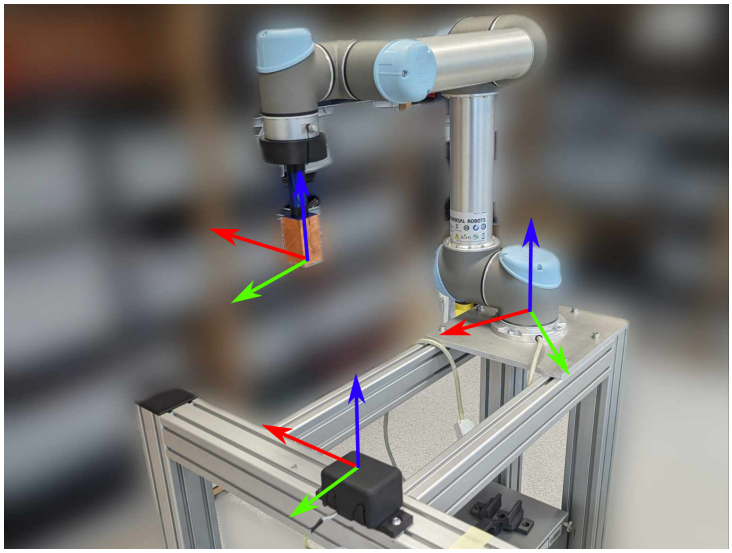


Obrázek 3.3: Použité chapadlo pro testování senzoru

Kapitola 4

Návrh řešení

Robot je upevněn na pohyblivém vozíku. Propojení robota s řídicím počítačem je realizované připojením robota do sítě. Jelikož se předpokládá častá výměna senzorů a práce v jeho okolí s aktivními kolaborativními funkcemi robota, není kolem robota namontovaná ochranná klec.

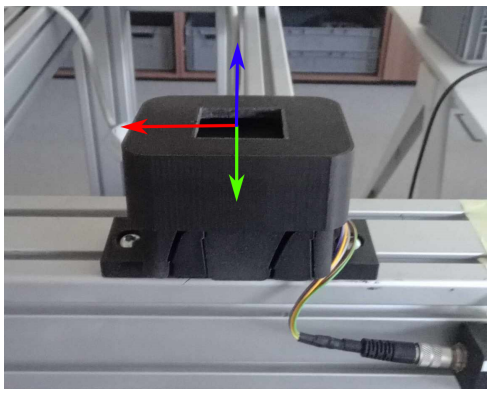


Obrázek 4.1: Pracoviště

Na obrázku 4.1 se vyskytuje vozík, na kterém je upevněný robot UR5 s testovaným senzorem. Jsou zde zároveň znázorněny souřadnicové systémy senzoru, základny robota a chapadla. Robot je ve výchozí pozici, která byla nastavena vyvíjenou knihovnou. Modrá osa znázorňuje osu z, červená osu x a zelená osu y.

Senzor je nehybně upevněn vůči základně robota v jeho dosahu tak, aby dokázal detekovat všechny požadované pohyby. Jeho montáž probíhá upevněním nástavce pro uložení senzoru. Toto zajišťuje možnost jednoduché výměny čipů se zachováním relativní polohy souřadnicové soustavy vůči základně robota a eliminuje se tak nutnost jejího nalézání při opakovaném testování a výměně senzoru.

Vzájemná poloha souřadnicového systému senzoru vůči základně robota je nalezená pomocí učícího režimu kolaborativního robota a manuálním navigováním chapadla do konkrétní pozice vůči souřadnicovému systému senzoru. K navigaci do konkrétní polohy napomáhá nástroj dočasně připevněný na pouzdře senzoru (viz obr 4.2).



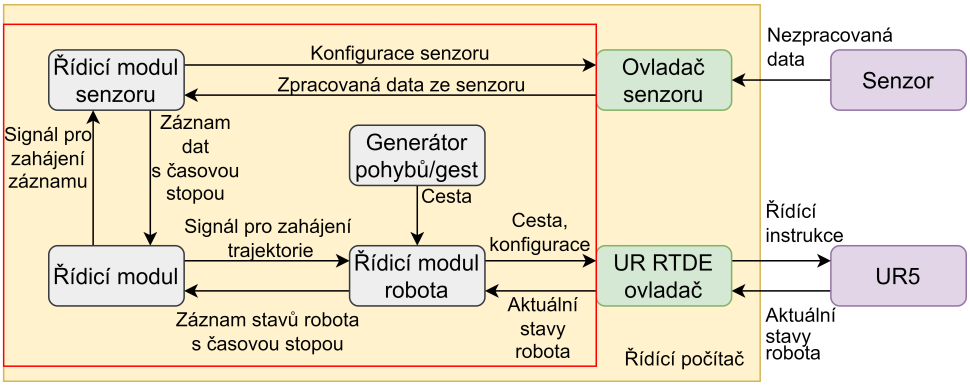
Obrázek 4.2: Nástavec pro nalezení vzájemné polohy souřadnicových systémů

Obrázek 4.2 zobrazuje souřadnicovou soustavu senzoru. Vůči této souřadnicové soustavě jsou navrhovány cesty testu. Pro nalezení vzájemné polohy souřadnicových systémů senzoru a základny robota se snažíme umístit chapadlo do nástavce.

4.1 Struktura modulů

Výsledná knihovna pro vytváření testů je rozdělena do jednotlivých modulů zajišťující určité funkcionality. Toto usnadňuje modularitu a správu celého systému.

Pro zachování jednotnosti vyvíjených aplikací v systémech zákazníka je celá knihovna psaná v jazyce Python. Knihovna pro testování senzorů pracuje s dvěma hlavními periferiemi, kterými jsou robot UR5 a testovaný senzor. Knihovna pro testování senzorů byla rozdělena do modulů podle blokového diagramu 4.3



Obrázek 4.3: Struktura vyvíjené knihovny

Blokový diagram v obrázku 4.3 znázorňuje, jak mezi sebou jednotlivé bloky komunikují. Fialové bloky znázorňují vnější periferie. Zelené bloky znázorňují poskytnuté prostředky zákazníkem či třetí stranou. V červeném rámečku se vyskytují chybějící bloky.

Generátor pohybů/gest

Úkolem tohoto bloku je vytvářet cesty v podobě datových struktur. Tyto datové struktury obsahují parametry pro robota, aby mohl bezproblémově vykonat různé pohyby. Parametry pohybu jsou nastavovány podle konkrétních požadavků uživatele. Důležitou konfigurací modulu je nalezená vzájemná poloha souřadnicových systémů senzoru a základny robota.

Součástí modulu jsou různé modifikace cest, včetně jejich translací, rotací a spojování. Hlavními druhy pohybů jsou pohyby po úsečkách a kruhových obloucích. O vytváření cest pro pohyb robota více v kapitole 4.2.

Řídicí modul senzoru

Úkolem tohoto bloku je zaznamenávání signálu ze senzoru v požadovaných momentech a následném uložení do paměti ve správné datové struktuře.

Senzor je připojený k řídicímu počítači pomocí rozhraní USB, modul tedy předává ovladači senzoru informace ohledně portu, ke kterému je senzor připojen, společně s informací, která data se mají vyčítat. Vyčítání dat probíhá asynchronně, ale neobsahuje časovou značku, která by sloužila k synchronizaci signálu s robotem. Před uložením vzorku do paměti se data doplňují o systémový čas řídicího počítače, v moment, kdy byla data zaznamenána.

Vyčítání probíhá asynchronně a pro spolehlivější záznam signálu je nezbytné nastavení priorit aplikace v reálném čase.

Řídicí modul robota

Blok ovladače robota je zodpovědný za bezpečné řízení pohybů robota, na základě předem definovaných instrukcí, a vyčítání dat ohledně polohy, orientace, rychlostí a případných dalších požadovaných stavů robota. Modul také zajišťuje nastavneí výchozí pozice robota, chapadla a priorit v reálném čase.

Před vykonáním pohybů má modul za úkol, pomocí ovladače robota, zkontrolovat dosahy a zda inverzní kinematická úloha pro požadovanou polohu má řešení. Vykonávání pohybu probíhá asynchronně společně s vyčítáním požadovaných stavů robota. Tato data se ukládají do paměti s časovou značkou.

Účel tohoto modulu je zjednodušení uživatelského rozhraní ovladače pro řízení robota.

Řídicí modul

Blok řídicího modulu vysílá signál pro zahájení požadovaného testu a pro vyčítání snímaných dat po dobu pohybu. Po vykonání pohybu se přijaté signály z robota a senzoru synchronizují. Signály se průběžně ukládají do jednotlivých souborů ve formátu json nebo csv. O synchronizaci v kapitole 4.3.

4.2 Návrh cesty

4.2.1 Reprezentace souřadnicového systému

Při vytváření testovacích scénářů a cest, jak se má robot pohybovat, je pro uživatele jednodušší se orientovat v souřadnicovému systému senzoru. Pro pohyby robotem je však nezbytné nalézt souřadnicové systémy vůči základně robota. Pro tento účel se často využívá homogenních transformačních matic, které mají tvar:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}, \tag{4.1}$$

kde $\mathbf{R}$ je rotační matice o rozměrech 3×3, znázorňující natočení souřadnicových systémů vůči sobě, vektor $\mathbf{t}$ znázorňuje posun mezi souřadnicovými systémy.

Požadované transformační matice se získají maticovým násobením transformační matice souřadnicového systému ze základny robota do senzoru a transformační matice souřadnicového systému ze senzoru do cíle.

$$\mathbf{T}_{1\to3} = \mathbf{T}_{1\to2}\mathbf{T}_{2\to3} \tag{4.2}$$

Osa-úhel reprezentace rotace

Natočení souřadnicových systémů vůči sobě lze reprezentovat i reprezentací osa-úhel, kde jednotkový vektor $\mathbf{k}$ označuje směr osy, kolem které se souřadnicové systémy natáčí, a úhel θ popisující velikost a směr rotace.
Výhodou reprezentace osa-úhel je parametrizace rotace ve třírozměrném prostoru podle každé osy pomocí dvou hodnot.
Pro nalezení rotační matice z reprezentace osa-úhel lze využít Rodriguesova vzorce ([6], rovnice 3.51):

$$\mathbf{R}(\mathbf{k},\theta) = \mathbf{I} + (\sin\theta)\mathbf{K} + (1-\cos\theta)\mathbf{K}^2, \tag{4.3}$$

kde $\mathbf{K}$ je antisymetrická matice vycházející z vektorového součinu, obsahující prvky vektoru $\mathbf{k}$ ve tvaru

$$\mathbf{K} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix}. \tag{4.4}$$

Orientace chapadla

U robotů řady UR se nastavuje orientace chapadla pomocí exponenciálních souřadnic. Jelikož počítač pracuje s rotačními maticemi, je nutné rotační matici převést na exponenciální souřadnice ve tvaru rotačního vektoru.
Vztah mezi rotačním vektorem ω a reprezentací osa-úhel je $\omega = \theta\mathbf{k}$

Úhel θ se vypočítá podle [6], rovnice 3.54

$$\theta = \arccos\left(\frac{1}{2}(\mathrm{tr}\mathbf{R} - 1)\right) \tag{4.5}$$

a antisymetrická matice $\mathbf{K}$ podle [6], rovnice 3.53 jako

$$\mathbf{K} = \frac{1}{2\sin\theta}(\mathbf{R} - \mathbf{R}^T), \tag{4.6}$$

kde $\mathbf{R}$ je rotační matice.

Kromě samotného chapadla můžou se senzorem interagovat i ramena manipulátoru. Toto může zapříčinit chybné měření. Pro minimalizaci interakcí ramen se senzorem je nutné nastavení správné orientace chapadla. V navržené knihovně je zajištěno, že osy z souřadnicových systémů chapadla a senzoru jsou udržovány rovnoběžné se shodnou orientací (viz obr. 4.1). Chapadlo se navíc otáčí kolem své osy z v závislosti na směru pohybu, tak aby se prováděla translace souřadnicového systému chapadla v ose x.

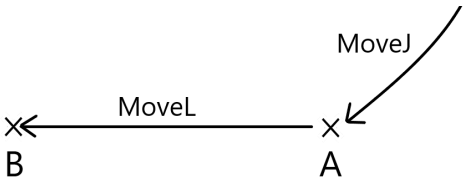
4.2.2 Pohyb po úsečce

Ovladač UR RTDE [12] v současné verzi poskytuje dva hlavní typy pohybu označené *MoveJ* a *MoveL* [13]. Tyto pohyby lze zadávat jak pozicí a orientací v prostoru, tak i konfigurací kloubů.

MoveJ provádí pohyby, které jsou vypočítávány v kloubovém prostoru. Klouby jsou řízeny tak, aby dosáhly požadované polohy současně. Tento typ pohybu má za následek trasu souřadnicového systému chapadla ve tvaru křivky. Tento typ pohybu je vhodný pro co nejrychlejší pohyb bez ohledu na trasu souřadnicového systému chapadla.

MoveL provádí translaci do koncového bodu. Klouby tedy provádí složitější pohyb.

Výsledná cesta pro pohyb po úsečce se skládá ze dvou pohybů. Prvním pohybem dosáhne souřadnicová soustava chapadla počátečního bodu úsečky. Druhým pohybem se vykoná translace souřadnicového systému chapadla mezi počátečním a koncovým bodem úsečky. Úsečka se v knihovně definuje těmito dvěma body.



Obrázek 4.4: Pohyb po úsečce

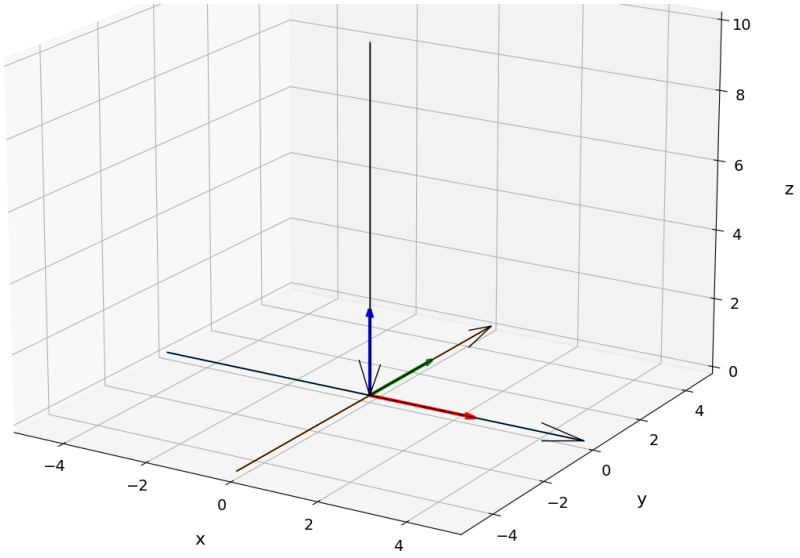
Obrázek 4.4 znázorňuje pohyb po úsečce definované vyvíjenou knihovnou.

Definování pomocí parametrů

Druhou možností definice úsečky v knihovně je definice pomocí parametrů. Tato metoda byla přidána pro rychlejší a jednodušší vytváření cest. Její nevýhodou je však možnost vytvoření pouze tří základních cest. Tyto tři základní cesty jsou úsečky ve směrech osy x, y a z. Parametry pro úsečkou jsou v tomto případě délka a zvolená osa. Tento přístup umožňuje uživateli specifikovat požadovanou úsečku intuitivněji, což urychluje proces u jednodušších pohybů.

V případech výběru osy x či y se nachází střed úsečky v počátku souřadnicového systému senzoru, znázorněným v obrázku 4.2. V případě osy z se z důvodu bezpečnosti nachází koncový bod v počátku souřadnicového systému senzoru.

Obrázek 4.5 znázorňuje, jak vypadají jednotlivé pohyby po úsečkách definovaných pomocí délky úsečky a zvolených os.



Obrázek 4.5: Pohyb po úsečce pomocí parametrů

4.2.3 Pohyb po kruhovém oblouku

Jelikož ovladač UR RTDE v současné verzi nepodporuje pohyby po kruhových obloucích, je zapotřebí řídit pohyb po kruhovém oblouku pomocí sekvence lineárních pohybů pro jeho aproximaci. Pro získání středu a poloměru kružnice je zapotřebí znát tři body, které se nacházejí na kružnici.

Pro libovolné, různé tři body A, B, C a rovinu xy se postupuje následovně.

Převedení do dvourozměrného prostoru

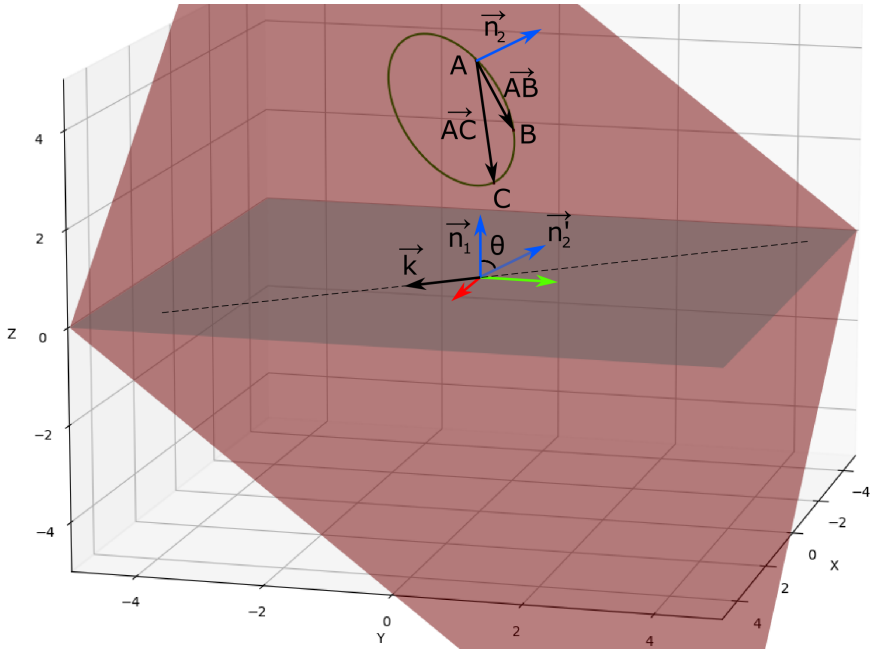
Pro identifikaci je zapotřebí transformovat body kruhového oblouku do roviny xy, což identifikaci ulehčí. Matice rotace se nalezne pomocí reprezentace osa-úhel a Rodriguesova vzorce 4.3.

Úhel θ nalezne vypočítáním úhlu, které svírají normálové vektory roviny xy a roviny, ve kterém se kruhový oblouk nachází. V tomto případě je normálový vektor roviny xy roven vektoru $(0, 0, 1)$. Normálový vektor roviny, ve kterém se kruhový oblouk nachází, se nalezne vektorovým součinem vektorů vzniklých ze tří bodů (obr. 4.6). Jsou-li tyto vektory lineárně závislé, pak jedná o přímku.

$$\theta = \arccos\left(\frac{\mathbf{n_1} \cdot \mathbf{n_2}}{|\mathbf{n_1}| \cdot |\mathbf{n_2}|}\right) = \arccos\left(\frac{(0,0,1) \cdot (\vec{AB} \times \vec{AC})}{|(0,0,1)| \cdot |\vec{AB} \times \vec{AC}|}\right). \tag{4.7}$$

Vektor $\mathbf{k}$, označující směr osy, kolem které se souřadnicové systémy natáčí, v rovnici 4.3 se nalezne vektorovým součinem normálových vektorů rovin.

$$\mathbf{k} = \frac{\mathbf{n_1} \times \mathbf{n_2}}{|\mathbf{n_1} \times \mathbf{n_2}|} = \frac{(0,0,1) \times (\vec{AB} \times \vec{AC})}{|(0,0,1) \times (\vec{AB} \times \vec{AC})|}. \tag{4.8}$$



Obrázek 4.6: Parametry osa-úhel reprezentace

Obrázek 4.6 je ilustrativní obrázek, který znázorňuje nalezení parametrů rotace v reprezentaci osa-úhel. Výsledná reprezentace osa-úhel slouží k transformaci bodů A, B, C do roviny xy.

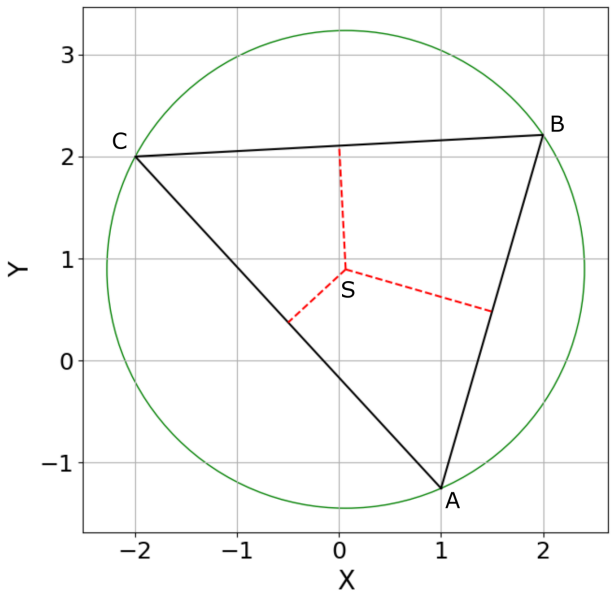
Nalezení středu a poloměru kružnice

Pro nalezení středu kružnice se vypočítají středy úseček vytvořených ze zadaných bodů. Každým středem úsečky je proložena přímka ve směru normálového vektoru dané úsečky. Střed kružnice se nachází v průsečíku těchto přímek a je dopočítán jako řešení soustavy lineárních rovnic (obr 4.7).

$$\begin{aligned} S &= S_{AB} + \mathrm{t} n_{AB} \\ S &= S_{AC} + \mathrm{t} n_{AC}, \end{aligned} \tag{4.9}$$

Rovnice 4.9 znázorňuje soustavu lineárních rovnic. Každá rovnice reprezentuje parametrický tvar přímky. S_{AB} a S_{AC} jsou středy úseček AB a AC. Vektory n_{AB} a n_{AC} jsou normálové vektory úseček. S je střed kružnice.
Poloměr je vypočítán jako vzdálenost středu od libovolného bodu.

Obrázek 4.7 je ilustrativní obrázek, znázorňující nalezení středu kružnice ze zadaných bodů A, B a C.



Obrázek 4.7: Nalezení středu kružnice

Počet diskrétních pohybů kruhového oblouku

Počet diskrétních pohybů, kterými robot aproximuje pohyb po kruhovém oblouku lze nastavit uživatelem. Tato hodnota ovlivňuje míru zkreslení kruhového oblouku. Obecně platí, že čím větší počet diskrétních pohybů, tím méně zkreslená je výsledná cesta pohybu robota.

Příliš velký počet diskrétních kroků může vést ke zpomalení pohybu robota z důvodu výpočetní náročnosti. Při příliš vysokém počtu kroků se snižuje délka jednotlivých pohybů. To může způsobit, že robot, kvůli nedokonalostem senzorů, chybně identifikuje další cílové polohy jako již dosažené. Maximální a minimální počet kroků pro aproximaci kruhového oblouku se proto počítá pro každý oblouk podle jeho délky a minimální a maximální délky jednotlivého kroku, které lze v knihovně nakonfigurovat.

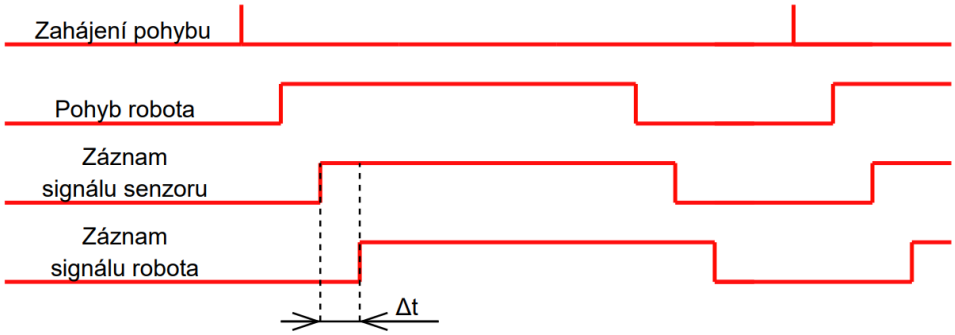
4.2.4 Plánování cesty

Hodnoty ze senzoru jsou zaznamenávány po celou dobu vykonávání cesty, včetně pohybu do počátečních bodů. Tyto pohyby mohou být zaznamenány senzorem a komplikovat tím následnou analýzu signálu. U některých senzorů se může vyskytovat i hystereze signálu a měření tím více zkreslovat. Abychom se vyvarovali těmto signálům a chybné interpretaci dat, lze cestu rozšířit o translace souřadnicového systému chapadla v ose z na začátku a na konci cesty.

Toto zajistí pohyb souřadnicové soustavy chapadla do počátečního bodu mimo dosah senzoru. Toto minimalizuje chybu měření během pohybu souřadnicové soustavy chapadla do počátečního bodu cesty.

4.3 Synchronizace dat

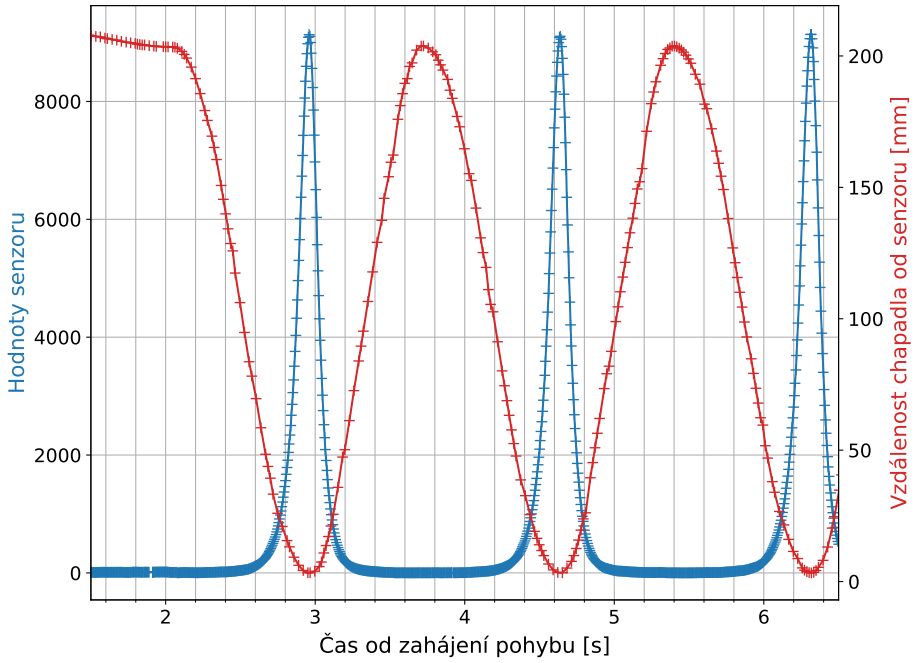
Řídicí modul senzoru a robota poskytuje záznam signálu, který byl zaznamenán během vykonávání cesty. Časové údaje přiřazené k jednotlivým vzorkům odpovídají systémovému času řídicího počítače v okamžiku, kdy byl vzorek zaznamenán. Hodnoty zaznamenané robotem v daném čase však neodpovídají hodnotám měřenými senzorem. Toto je zapříčiněno dopravním zpožděním v komunikaci mezi počítačem a zařízením. Tyto odchylky vedou k nesprávné interpretaci dat a mají vliv na kvalitu výsledků jejich následných analýz.



Obrázek 4.8: Časový diagram

Z tohoto důvodu je nutné nalézt odchylku signálů v čase a minimalizovat tak chybu naměřených dat. Jelikož senzor zaznamenává pohyby a vzdálenosti v osách x, y, z (kapitola 3.3), lze měření realizovat opakovanou translací souřadnicového systému chapadla ve směru osy z, v dosahu senzoru (obrázek 4.9). Senzor zaznamenává největší signál v momentě, kdy je chapadlo nejblíže senzoru. Výsledné hodnoty ze senzoru a robota s časovými značkami lze pak analyzovat a porovnat tak rozdíl v čase mezi extrémy obou signálů.

Pro každou periodu se provede nalezení lokálního extrému a následně se porovnají rozdíly v čase konkrétních extrémů obou signálů. Nakonec se výsledné odchylky v časech mezi extrémy průměrují a u následných měření lze provádět korekce časových značek.



Obrázek 4.9: Měření odezvy senzoru během translace chapadla v ose z

Kapitola 5
Implementace

V této kapitole bude poskytnut popis implementace vyvíjené knihovny. Nejprve budou objasněny závislosti vyvíjené knihovny, které zahrnují potřebné verze softwaru, balíčky a knihovny nezbytné pro její správnou funkčnost. Následně bude popsána vytvořená datová struktura, kterou vyvíjená knihovna využívá k manipulaci s daty. Poslední částí této kapitoly bude popis jednotlivých tříd, funkcí a metod, které tvoří jádro vyvíjené knihovny.

5.1 Závislosti

Knihovna využívá řadu závislostí, které jsou klíčové pro správné fungování a efektivitu vývoje. Závislosti představují extérní balíčky nebo knihovny a specifické verze softwaru, které projekt využívá k implementaci specifických funkcionalit bez nutnosti vytvářet vlastní řešení od základu.

- Projekt je navržen ve verzi jazyku **Python 3.10** [1]. Knihovna UR RTDE v aktuální verzi není kompatibilní s novějšími verzemi Pythonu. V současné verzi projektu není vyvíjená knihovna kompatibilní ani se staršími verzemi Pythonu. Důvodem je využití specifických funkcí dostupný od verze jazyku Python 3.10.
- Knihovna **Numpy**[2] poskytuje projektu především efektivní manipulaci s daty a různé matematické operace, včetně výpočtů transformačních matic.
- Knihovna **Pandas**[3] poskytuje projektu manipulaci se zaznamenanými daty ze senzoru a robota. Nabízí rozsáhlé možnosti pro manipulaci, včetně filtrace, selekce a podobně.
- Knihovna **Scipy**[4] poskytuje nástroje pro výpočetní matematiku, optimalizaci, statistiku a další. V projektu je využita pro analýzu zaznamenaných dat. Využívá se hlavně pro nalezení lokálních extrémů zaznamenaných signálů.
- **UR RTDE** [12] (kapitola 3.2.2) především umožňuje projektu efektivní ovládání a monitorování stavů robota.

[1] `https://www.python.org/`
[2] `https://numpy.org/`
[3] `https://pandas.pydata.org/`
[4] `https://scipy.org/`

- Knihovna **pywin32** [5] v projektu slouží k nastavení priorit aplikace v reálném čase.

5.2 Datová struktura cest a pohybů

Pro popis cest testovacích scénářů byla vytvořena třída **MotionSequence**. Pro popis jednotlivých pohybů cesty byly vytvořeny datové třídy (*dataclass*) **Linear(Motion)** a **Joint(Motion)**, které obsahují požadované parametry pro vykonání pohybu.

Třída MotionSequence

Bloky v digramu 4.3 si mezi sebou předávají informace pro vykonání cesty pomocí datové třídy (*dataclass*) **MotionSequence**.
Atributem této třídy je:

- `path`... seznam obsahující objekty **Motion**. Tyto objekty obsahující parametry jednotlivých pohybů.

Na třídě je definovaná metoda sčítání, která slouží ke spojování cest.

Třída Motion

Pro parametry jednotlivých pohybů se využívá datové třídy (*dataclass*) **Motion**.
Atributy třídy jsou:

- `position`... vektor *(x, y, z)* v jednotkách mm.
- `orientation`... rotační vektor (kapitola 4.2.1).

Z datové třídy (*dataclass*) **Motion** dědí atributy datové třídy (*dataclass*) **Linear(Motion)** a **Joint(Motion)**, které slouží k rozpoznání druhu pohybu (kapitola 4.2.2).
Atributy datové třídy (*dataclass*) **Linear(Motion)**:

- `v`... rychlost v jednotkách mm/s
- `a`... zrychlení v jednotkách mm/s^2.

Atributy datové třídy (*dataclass*) **Joint(Motion)** jsou:

- `v`... úhlová rychlost v jednotkách rad/s
- `a`... úhlové zrychlení v jednotkách rad/s^2.

[5] `https://pypi.org/project/pywin32/`

5.3 Řídicí modul senzoru

Řídicí modul senzoru z diagramu 4.3 je implementován třídou **Sensor**.

Konstruktor

Argumentem konstruktoru je:

- `port`... textový řetězec specifikující port, na kterému je senzor připojen.

Atributy třídy jsou:

- `serial`... port, na kterém je senzor připojen
- `t_read`... vlákno sloužící k přijímání dat.
- `data_frames`... seznam zaznamenaných dat.
- `read_flag`... indikátor, zda se mají zaznamenaná data ukládat.

V rámci této metody dochází k inicializaci sériového portu a dalších proměnných. Dále je za pomocí ovladače senzoru vytvořeno čtecí vlákno. Toto vlákno při záznamu dat ze senzoru volá metodu '_callback'.

Metoda '_callback'

Jedná se o privátní metodu, která je volána čtecím vláknem pokaždé, když jsou přijata nová data ze senzoru. Tuto metodu je nutné modifikovat v případě výměny senzoru a ovladače senzoru. Je-li `read_flag` nastaven na hodnotu `True`, data jsou zpracována, doplněna o systémový čas řídicího počítače a uložena do paměti. Zaznamenaná data se ukládají ve formě datové struktury **DataFrame** z knihovny Pandas.

Metoda 'start' a 'stop'

Metoda 'start' aktivuje sběr dat nastavením atributu `read_flag` na hodnotu `True` a resetuje záznam `data_frames`.
Metoda 'stop' deaktivuje sběr dat nastavením atributu `read_flag` na hodnotu `False`.
Výstupem metody 'stop' je:

- `final_df`... Objekt **DataFrame** obsahující záznam signálu ze senzoru.

Metoda 'terminate'

Metoda slouží pro zastavení čtecího vlákna a uvolnění zdrojů. Toto je klíčové pro správné ukončení aplikace.

Integrace a práce s daty

Po inicializace třídy **Sensor** lze pomocí metod 'start' a 'stop' ovládat sběr dat. Toto umožňuje dynamicky zahajovat a ukončovat záznam dat v závislosti na potřebách uživatele. Výsledná datová struktura **DataFrame** obsahuje sloupec `data` se zaznamenanými hodnotami ze senzoru a sloupec `timestamp` obsahující čas v moment, kdy byla data zaznamenána řídicím počítačem.

5.4 Řídicí modul robota

Řídicí modul senzoru z diagramu 4.3 je implementován třídou **UR**.

Konstruktor

Argumenty konstruktoru jsou:

- `addr`... textový řetězec specifikující IP adresu, ke které je robot připojen.
- `real_time`... logická hodnota určující, zda má být robot ovládán s prioritami v reálném čase.

Atributy třídy jsou:

- `io`... instance třídy **RTDEIO** z knihovny UR RTDE sloužící k ovládání vstupních a výstupních periférií robota (kapitola 3.2.2).
- `rec`... instance třídy **RTDEReceive** z knihovny UR RTDE sloužící k přijímání informací o stavu robota (kapitola 3.2.2).
- `ctrl`... instance třídy **RTDEControl** z knihovny UR RTDE sloužící řízení robota (kapitola 3.2.2).
- `period`... perioda čtení dat z robota v jednotkách sekund.
- `home_q`... konfigurace kloubů výchozí pozice robota v jednotkách rad.

V rámci metody dochází k připojení k robotu a inicializaci jeho ovládacích prvků. Následně se nastaví pomocí metody 'set_home' výchozí pozice robota a pomocí metody 'set_tcp' souřadnicová soustava chapadla, zobrazena na obrázku 4.1. Po té se nastaví výchozí perioda vyčítání hodnot z robota pomocí metody 'set_read_period'.

Metoda 'set_read_period'

Metoda slouží k nastavení periody čtení dat z robota. Minimální perioda je 0,015 sekund. Tato perioda byla určena experimentálně. Při nižších hodnotách dochází k chybnému měření signálů.
Argumentem metody je:

- `period`... perioda v jednotkách sekund. Výchozí hodnota je 0,015 sekund.

Metoda 'set_home'

Metoda slouží ke změně výchozí pozice robota.
Argumentem metody je:

- `joint_q`... konfigurace kloubů v jednotkách rad.

Metoda 'home'

Metoda zajišťuje pohyb robota do výchozí pozice. Tato metoda slouží ke změně výchozí pozice robota.
Argumentem metody je:

- `safe_home`... volitelný parametr určující, zda se má použít bezpečný režim návratu do výchozí pozice.

Metoda 'set_tcp'

Metoda pro nastavení souřadnicového systému chapadla.
Argumenty metody jsou:

- `translation`... translační vektor *(x, y, z)* souřadnicovému systému chapadla vůči souřadnicovému systému posledního kloubu. Translační vektor se uvádí v jednotkách mm.
- `rotation`... rotační vektor určující orientaci souřadnicového systému chapadla.

Metoda 'set_payload'

Metoda pro nastavení zátěže a těžiště nástroje na konci ramene robota.
Argumenty metody jsou:

- `mass`... hmotnost chapadla v jednotkách kg.
- `cog`... volitelný translační vektor označující těžiště nástroje vůči souřadnicovému systému posledního kloubu. Uveden v jednotkách mm.

Metoda 'teach_mode' a 'end_teach_mode'

Metody pro aktivaci a deaktivaci učícího režimu robota.

Metoda 'move_path'

Metoda pro vykonání pohybu po definované cestě. Cesta je specifikovaná objektem **MotionSequence** obsahující seznam pohybů, které má robot vykonat. V rámci metody se před vykonáním jakéhokoliv pohybu provede validace cesty pomocí privátních metod '_valid_path' a '_valid_position'.

Během vykonávání pohybu probíhá asynchronně čtení dat voláním privátní metody '_read_data'. V aktuální verzi knihovny obsahuje výsledná datová struktura **DataFrame** sloupec `position`, `orientation`, `v` a `timestamp`.
Argumentem metody je:

- `path`... objekt **MotionSequence** specifikující cestu, která se má vykonat.

Výstupem metody:

- `final_df`... objekt **DataFrame** obsahující záznam stavů robota během vykonávání cesty.

Metoda '_read_data'

Metoda ke čtení aktuálních stavů robota. Metodu je zapotřebí modifikovat podle požadavků uživatele na to, které stavy se mají vyčítat. Možnými volbami stavů pro vyčítání jsou vzájemná poloha, konfigurace kloubů, rychlost, zrychlení, teploty, momenty a další [12].

V aktuální verzi knihovny se vyčítá rychlost chapadla a vzájemná poloha souřadnicových systémů chapadla a základny robota.
Návratová hodnota:

- `df`... objekt **DataFrame** obsahující vzorek zaznamenaného signálu robota.

Metoda 'stop_script'

Robot se přesune do výchozí pozice a metoda ukončí běžící skript na ovládacím tabletu robota.

5.5 Generátor pohybů

Třída **MotionGenerator** slouží k vytváření cest třídy **MotionSequence** pomocí pohybových příkazů **Motion**. Pohyby se vytváří na základě souřadnicového systému senzoru.

Konstruktor

Argumenty kostruktoru:

- `robot2target`... translační vektor souřadnicové soustavy senzoru vůči základně robota. Uváděn v jednotkách mm.
- `robot2target_orientation`... orientace souřadnicové soustavy senzoru vůči základně robota. Očekávaný vstup je rotační matice nebo rotační vektor.
- `v`... volitelná výchozí lineární rychlost v jednotkách mm/s. Výchozí hodnota je 300 mm/s.
- `a`... volitelné výchozí lineární zrychlení v jednotkách mm/s^2. Výchozí hodnota je 500 mm/s^2.
- `omega`... volitelná výchozí úhlová rychlost v jednotkách rad/s. Výchozí hodnota je 0.5π rad/s.

- `alpha`... volitelné výchozí úhlové zrychlení v jednotkách rad/s^2. Výchozí hodnota je π rad/s^2.
- `neutral_distance`... maximální translace ve směru osy z na začátku a konci cesty (kapitola 4.2.4). Uváděno v jednotkách mm.

Atributy třídy:

- `T`... transformace souřadnicového systému senzoru ze základny robota. Objekt **SE3** s argumenty 'translation' a 'rotation'.
- `inv_T`... transformace souřadnicového systému základny robota ze senzoru.
- `v`... výchozí lineární rychlost.
- `a`... výchozí lineární zrychlení.
- `omega`... výchozí úhlová rychlost.
- `alpha`... výchozí úhlové zrychlení.
- `neutral_distance`... maximální vzdálenost translace ve směru osy z na začátku a konci cesty.

V rámci této metody se ze zadaných argumentů vypočítají všechny potřebné transformační matice pro následné vytváření pohybových instrukcí pro požadované cesty.

Metoda 'add_neutral_points'

Metoda přidává translaci ve směru osy z na začátek a konec zadané cesty `path`. Nové počáteční a koncové body cesty budou v minimální vzdálenosti `neutral_distance` v ose z od souřadnicového systému senzoru.
Argument metody:

- `path`... objekt **MotionSequence**. Cesta která se rozšíří o pohybové příkazy obsahující translaci v ose z.

Návratová hodnota:

- Objekt **MotionSequence**. Cesta rozšířena o pohybové příkazy obsahující translaci v ose z.

Metoda 'get_line'

Metoda vytvářející příkazy pro lineární pohyb mezi dvěma body (kapitola 4.2.2).
Argumenty metody:

- `A`... počáteční bod úsečky v souřadnicovém systému senzoru. Uváděn v jednotkách mm.
- `B`... koncový bod úsečky v souřadnicovém systému senzoru. Uváděn v jednotkách mm.
- `v`... volitelná lineární rychlost v jednotkách mm/s. Výchozí hodnota podle atributu třídy.

- `a`... volitelné lineární zrychlení v jednotkách mm/s^2. Výchozí hodnota podle atributu třídy.
- `neutral_points`... indikátor, zda se do cesty mají zahrnout translace ve směru osy z. Výchozí hodnota `True`.

Návratová hodnota:

- Objekt **MotionSequence**.

Metoda 'get_gesture'

Metoda vytvářející příkazy pro lineární pohyb pomocí parametrů (kapitola 4.2.2).
Argumenty metody:

- `length`... délka úsečky v jednotkách mm.
- `axis`... osa, podél které bude úsečka vytvořena ("x", "y" nebo "z"). Výchozí hodnota je "x".
- `v`... volitelná lineární rychlost v jednotkách mm/s. Výchozí hodnota podle atributu třídy.
- `a`... volitelné lineární zrychlení v jednotkách mm/s^2. Výchozí hodnota podle atributu třídy.
- `neutral_points`... indikátor, zda se do cesty mají zahrnout translace ve směru osy z. Výchozí hodnota `True`.

Návratová hodnota:

- Objekt **MotionSequence**.

Metoda 'get_circular_motion'

Metoda vytvářející příkazy pro pohyb po kruhovém oblouku (kapitola 4.2.3). Maximální rychlost pohybu po kružnici byla omezena na 350 mm/s. Při vyšších rychlostech nebyla rychlost chapadla konzistentí.
Argumenty metody:

- `points`... seznam bodů definující kruhový oblouk.
- `amount`... počet diskrétních pohybů, které budou aproximovat pohyb po kruhovém oblouku.
- `v`... volitelná lineární rychlost v jednotkách mm/s. Výchozí hodnota podle atributu třídy.
- `neutral_points`... indikátor, zda se do cesty mají zahrnout translace ve směru osy z. Výchozí hodnota `False`.

Návratová hodnota:

- Objekt **MotionSequence**.

Metoda 'get_circular_gesture'

Metoda vytvářející příkazy pro pohyb po kruhovém oblouku zadanými parametry.
Argumenty metody:

- `width`... šířka oblouku v jednotkách mm.
- `height`... výška oblouku v jednotkách mm.
- `axis`... osa, podél které bude kruhový oblouk vytvořen("x"nebo "y"). Výchozí hodnota je "x".
- `amount`... počet diskrétních pohybů, které budou aproximovat pohyb po kruhovém oblouku.
- `v`... volitelná lineární rychlost v jednotkách mm/s. Výchozí hodnota podle atributu třídy.
- `neutral_points`... indikátor, zda se do cesty mají zahrnout translace ve směru osy z. Výchozí hodnota `False`.

Návratová hodnota:

- Objekt **MotionSequence**.

Metoda 'rotate_paths_axis'

Metoda provede rotaci cesty nebo seznamu cest kolem specifické osy x, (y) nebo z, souřadnicového systému senzoru.
Argumenty metody:

- `paths`... seznam cest obsahující objekty **MotionSequence** nebo samostatná cesta **MotionSequence**.
- `angle`... úhel rotace v radiánech.
- `axis`... osa, kolem které bude rotace prováděna ("x", "y" nebo "z"). Výchozí hodnota je "z".

Návratová hodnota:

- Seznam obsahující objekty **MotionSequence** nebo samostatný objekt **MotionSequence**. Struktura návratové hodnoty závisí na struktuře argumentu `paths`.

Metoda 'translate_paths'

Metoda provede translaci cesty nebo seznam cest.
Argumenty metody:

- `paths`... seznam cest obsahující objekty **MotionSequence** nebo samostatná cesta **MotionSequence**.
- `t`... translační vektor uváděn v jednotkách mm.

Návratová hodnota:

- Seznam obsahující objekty **MotionSequence** nebo samostatný objekt **MotionSequence**. Struktura návratové hodnoty závisí na struktuře argumentu `paths`.

Metoda 'rotate_paths_range'

Metoda v aktuální verzi provede vícenásobnou rotaci cesty, nebo seznamu cest, kolem osy z souřadnicového systému senzoru.
Argumenty metody:

- `paths`... seznam cest obsahující objekty **MotionSequence** nebo samostatná cesta **MotionSequence**.
- `angle_range`... seznam úhlů definující rozsah rotací. Uváděno v jednotkách rad.
- `amount`... počet, kolikrát se provede rotace pohybu.

Návratová hodnota:

- Seznam obsahující objekty **MotionSequence**.

Metoda 'translate_paths_range'

Metoda provede vícenásobnou translaci cesty nebo seznamu cest.
Argumenty metody:

- `paths`... seznam cest obsahující objekty **MotionSequence** nebo samostatná cesta **MotionSequence**.
- `t`... translační vektor uváděn v jendotkách mm.
- `amount`... počet, kolikrát se provede translace pohybu.

Návratová hodnota:

- Seznam obsahující objekty **MotionSequence**.

5.6 Řídicí modul testu

Řídicí modul je implementován třídou **SensorTest**.

Konstruktor

Argumentem konstruktoru je:

- `robot`... instance třídy **UR** sloužící k ovládání robota.
- `sensor`... instance třídy **Sensor** sloužící k ovládání senzoru.
- `mg`... instance třídy **MotionGenerator** sloužící k transformaci souřadnicových systémů.
- `time_offset` ... volitelná odchylka signálů v čase. Výchozí hodnota 0 s.

Atributy třídy jsou:

- `robot`... instance třídy **UR** sloužící k ovládání robota.
- `sensor`... instance třídy **Sensor** sloužící k ovládání senzoru.
- `mg`... instance třídy **MotionGenerator** sloužící k transformaci souřadnicových systémů.
- `time_offset` ... odchylka signálů v čase v sekundách.

Metoda 'execute_path'

Metoda slouží k zahájení záznamu ze senzoru a vykonání cest požadovaného testu. Po dokončení cesty se data zpracují, synchronizují a uloží v požadovaném formátu. Uložené pozice souřadnicového systému chapadla jsou vůči souřadnicovému systému senzoru.
Argumenty metody:

- `paths`... seznam cest obsahující objekty **MotionSequence** nebo samostatná cesta **MotionSequence**.
- `label`... popisek v názvu souboru, do kterého se mají zaznamenaná data uložit. Výchozí hodnota je aktuální datum a čas v moment zavolání metody.

Metoda 'measure_offset'

Metoda slouží ke změření odchylky signálů v čase. Pomocí instance třídy **MotionGenerator** se vytvoří sekvence pohybů pro změření odchylky. Po vykonání cesty jsou signály z robota a senzoru jsou zaznamenány a analyzovány pomocí knihovny **Scipy** (kapitola 4.3). Nalezená odchylka v čase se ukládá do atributu `time_offset`.
Návratová hodnota:

- Logická hodnota `True`, pokud měření proběho úspěšně.

Funkce 'set_rt_priority'

Součástí modulu je funkce 'set_rt_priority', která nastaví běžící aplikaci priority v reálném čase.

Integrace

Pro správnou inicializaci třídy **SensorTest** je nutné nejprve vytvořit instanci třídy **UR**, **Sensor** a **MotionGenerator**. Tyto instance tříd umožňují třídě **SensorTest** správné řízení testu a záznam dat. Není-li nastaven atribut `time_offset` nebo zavolána metoda 'measure_offset', neprovádí se korekce časových značek signálu.

Kapitola 6

Experimentální výsledky

6.1 Synchronizace času

Experiment byl proveden podle kapitoly 4.3. Robot prováděl opakovanou translaci ve směru osy z. Každá perioda obsahovala translaci ve směru osy $-z$ a $+z$ o délce 200 mm o rychlosti 600 mm/s (obrázek 4.9).

Perioda [-]	1	2	3	4	5	6	7	8	9	10
Odchylka v čase [ms]	52	29	43	45	43	53	49	51	41	27
Průměrná hodnota [ms]	43,40									

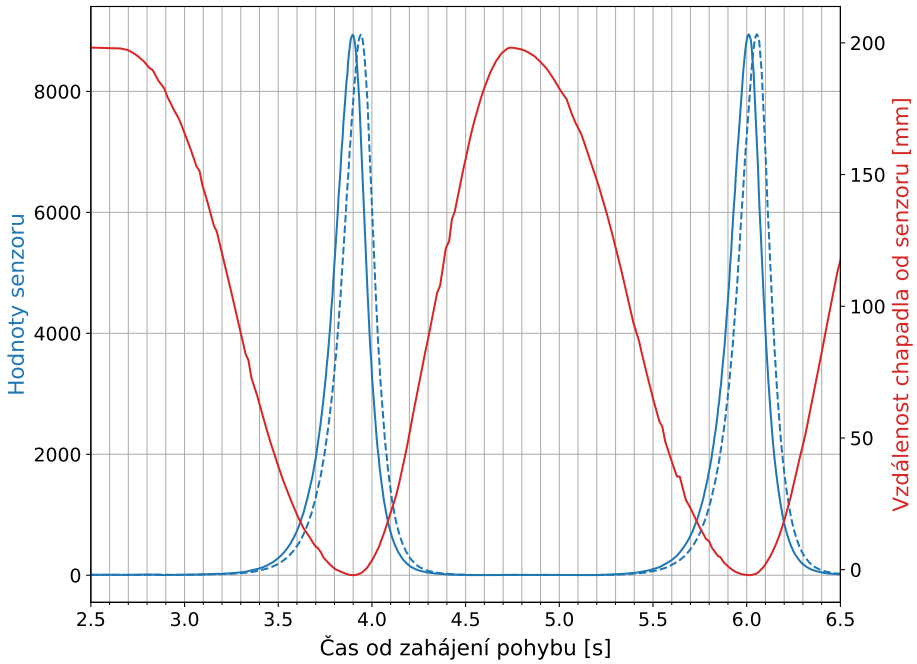
Tabulka 6.1: Měření časové odchylky signálů

Pro ověření změřené hodnoty bylo provedeno nové měření. Rychlost translace byla změněna na 300 mm/s.

Perioda [-]	**Odchylka s korekcí [ms]**	**Odchylka bez korekce [ms]**
1	1,60	45,00
2	-5,40	38,00
3	-12,40	31,00
4	0,60	44,00
5	-13,40	30,00
6	-0,40	43,00
7	-15,40	28,00
8	1,60	45,00
9	-2,40	41,00
10	4,60	48,00
Průměrná hodnota [ms]	-4,10	39,30

Tabulka 6.2: Porovnání kvality synchronizace

V obrázku 6.1 je znázorněn průběh hodnot ze senzoru a vzdálenosti souřadnicového systému chapadla od souřadnicového systému senzoru v čase. Graf obsahuje dva průběhy signálu senzoru. Přerušovaná čára reprezentuje původní naměřený signál. Plná čára reprezentuje signál, na kterém byla provedena korekce času pomocí změřené hodnoty.



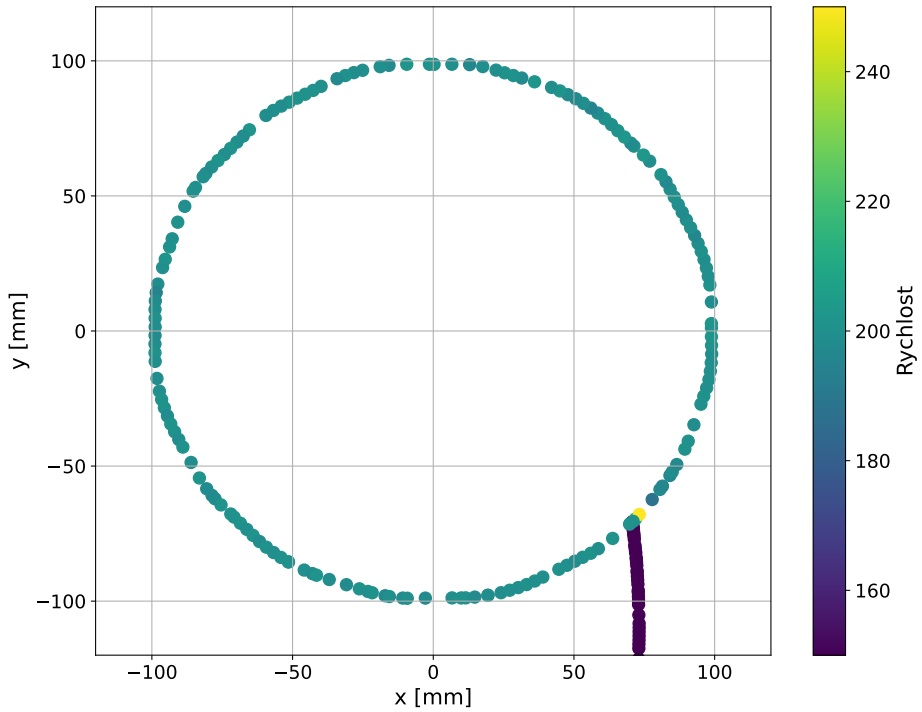
Obrázek 6.1: Synchronizovaný signál

Průměrná odchylka v čase signálů z robota a senzoru je po korekci 4,10 ms. Tato hodnota je výrazně nižší než průměrná hodnota před korekcí. Jelikož je vyčítací perioda senzoru 5 ms, lze synchronizaci signálů považovat za úspěšnou.

6.2 Pohyb po kruhovém oblouku

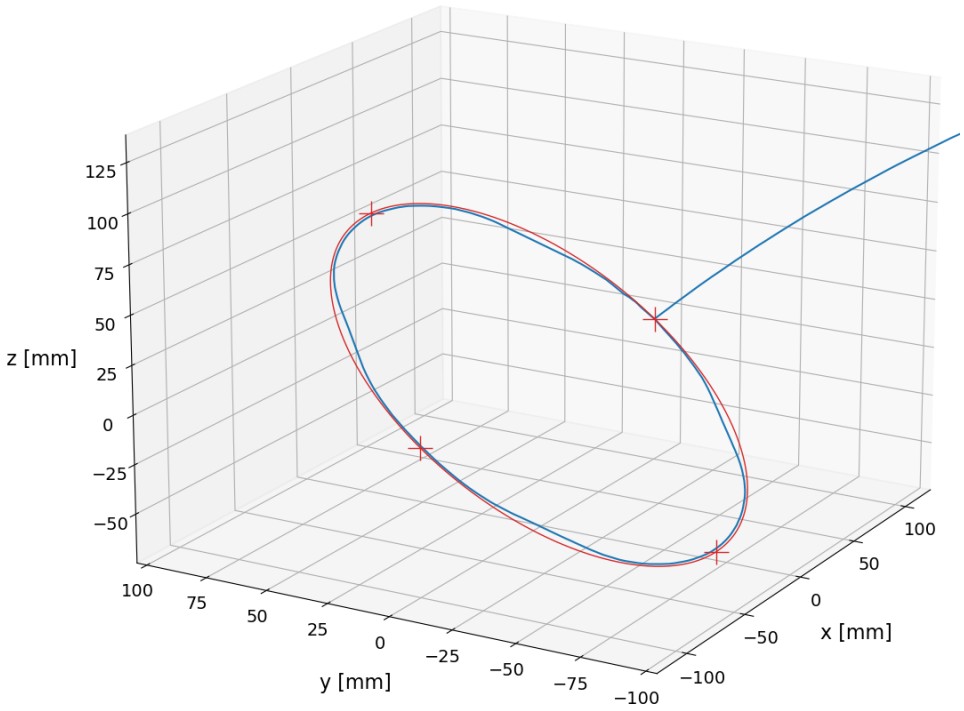
Pro měření po kruhovém oblouku byla vytvořena kružnice pomocí dvou oblouků. Body definující první kruhový oblouk jsou (100, 0, 0), (0, 100, 0) a (-100, 0, 0). Body definující druhý oblouk jsou (-100, 0, 0), (0, -100, 0) a (100, 0, 0). Jejich spojení tedy vytváří kružnici v rovině xy, nacházící se v počátku souřadnicového systému s poloměrem 100 mm. Nastavená rychlost byla 200 mm/s.

Z obrázku 6.2 můžeme vidět, průběh pohybu a konzistenci rychlosti chapadla během jeho vykonávání.



Obrázek 6.2: Rychlosti během pohybu po kruhovém oblouku

Obrázek 6.3 ukazuje, jak moc se robot přibližuje požadovanému pohybu po kruhovém oblouku. Pohyb se skládá ze dvou kruhových oblouků. První oblouk je zadán body (100, 0, 0), (0, 100·cos($\frac{\pi}{4}$), 100·sin$\frac{\pi}{4}$), (-100, 0, 0). Druhý oblouk je zadán body (-100, 0, 0), (0, -100·cos($\frac{\pi}{4}$), -100·sin($\frac{\pi}{4}$)), (100, 0, 0). Spojením těchto oblouků vzniká kružnice v rovině xy o poloměru 100 mm, která byla rotována kolem osy x o $\frac{\pi}{4}$ rad. Pohyb po jednotlivém kruhovém oblouku se skládal z minimálního počtu diskrétních pohybů, v tomto případě se jednalo o 10.



Obrázek 6.3: Pohyb po kruhovém oblouku 3D

Kapitola 7

Závěr

Tato práce se zaměřila na vývoj knihovny, která slouží jako nástroj pro spolehlivé testování různých typů senzorů. Cílem bylo vytvořit flexibilní a snadno modifikovatelný nástroj, který by umožnil efektivní testování senzorů s minimálními závislostmi.

Vytvořená knihovna umožňuje automatizované testování vyvíjeného senzoru pomocí robota UR5.

Během vývoje knihovny bylo rozhodnuto využít existující uživatelskou knihovny UR RTDE, která poskytuje dostatečné funkce pro ovládání robota.

Vývoj knihovny zahrnoval vytvoření datových struktur, rozdělení do modulů a vytvoření požadovaných funkcí pro testovací scénáře. Vytvořené datové struktury umožňují efektivní manipulace s daty a rozdělení do modulů výrazně zlepšuje modularitu a flexibilitu vyvíjené knihovny, což usnadňuje její adaptaci na nové senzory bez potřeby rozsáhlých úprav kódu.

Knihovna je navržena tak, aby byla jednoduše integrovatelná do existujících systémů a poskytovala uživatelsky přívětivé rozhraní pro konfiguraci a programování testovacích scénářů. Toto zahrnuje vytváření požadovaných cest různými způsoby, jejich úpravy, spojování a transformace.

Spolehlivé provedení testů zahrnuje kontrolu dosahů robota, vykonání cesty a zpracování dat, synchronizaci a ukládání datových záznamů.

Současná verze řešení splňuje požadavky specifikované zákazníkem a obsahuje funkce pro spolehlivé vykonání testovacích scénářů.

Pro rozsáhlejší testovací scénáře se zde však stále nachází prostor pro vylepšení a rozšíření knihovny. Pro časově náročné testy je vhodná implementace dalších bezpečnostních funkcí, které by zajišťovaly spolehlivé provedení testů. Dále by bylo vhodné implementovat více druhů pohybů, které by zahrnovaly různé orientace chapadla, simulace šumu a komplexnější křivky, čímž by se zvýšila robustnost testovacích scénářů.

Bibliografie

[1] A. Bonen et al. „A Novel Electrooptical proximity sensor for robotics: calibration and active sensing“. In: *IEEE Transactions on Robotics and Automation* 13.3 (1997), s. 377–386. DOI: 10.1109/70.585900.

[2] David Coleman; Ioan A. Șucan; Sachin Chitta; Nikolaus Correll. „Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study“. In: *Journal of Software Engineering for Robotics* 5.1 (květ. 2014), s. 3–16.

[3] Tiago Gomes et al. „Evaluation and Testing System for Automotive LiDAR Sensors“. In: *Applied Sciences* 12.24 (2022). ISSN: 2076-3417. DOI: 10.3390/app122413003. URL: https://www.mdpi.com/2076-3417/12/24/13003.

[4] Microchip Technology Inc. „GestIC® Design Guide“. In: (2013-2015). URL: https://docs.rs-online.com/6f15/0900766b81431793.pdf.

[5] J. Konstantinova et al. „Fingertip Proximity Sensor with realtime visual-based calibration“. In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, s. 170–175. DOI: 10.1109/IROS.2016.7759051.

[6] Kevin M. Lynch a Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017. ISBN: 9781107156302.

[7] *MoveIt Documentation*. URL: https://moveit.ros.org/.

[8] *OMPL Documentation*. URL: https://ompl.kavrakilab.org/.

[9] Jewoo Park et al. „An Automotive LiDAR Performance Test Method in Dynamic Driving Conditions“. In: *Sensors* 23.8 (2023). URL: https://www.mdpi.com/1424-8220/23/8/3892.

[10] *Real-Time Data Exchange(RTDE) guide*. Universal Robots. URL: https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/.

[11] *ROS Wiki*. URL: https://wiki.ros.org/.

[12] *UR RTDE Documentation*. URL: https://sdurobotics.gitlab.io/ur_rtde/.

[13] *UR5 datasheet*. Universal Robots. URL: https://s3-eu-west-1.amazonaws.com/ur-support-site/105364/99230_UR5_User_Manual_cs_Global.pdf.

[14] Guozhou Xiao a Zezhou Yang. „The Electromagnetic Nondestructive Testing Device of the wirerope-core transmission belt“. In: *2012 Power Engineering and Automation Conference*. 2012, s. 1–4. DOI: 10.1109/PEAM.2012.6612506.

Příloha A

Prohlášení o použití umělé inteligence

V souladu s *Metodickým pokynem 05/2023*[1], bylo při vypracování této práce použito následujících nástrojů.

- ChatGPT (OpenAI)[2]: pro úpravu textu a reformulace.

[1] https://intranet.fel.cvut.cz/cz/rozvoj/MP-pouzivani-ui.pdf
[2] https://chatgpt.com/