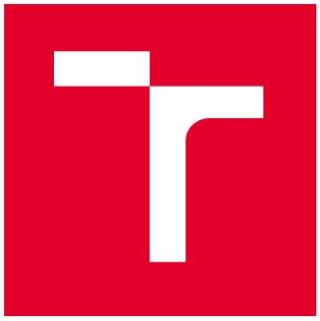


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

ŘÍZENÍ ROBOTA UR5 Z ROBOTICKÉHO OPERAČNÍHO SYSTÉMU (ROS)

CONTROL OF THE UR5 ROBOT FROM THE ROBOTIC OPERATING SYSTEM (ROS)

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Marcel Zdeněk

AUTHOR

VEDOUCÍ PRÁCE — Ing. Filip Radil

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Marcel Zdeněk**
Studijní program:	Mechatronika
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Filip Radil**
Akademický rok:	2022/23

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Řízení robota UR5 z Robotického operačního systému (ROS)

Stručná charakteristika problematiky úkolu:

Nasazení průmyslových robotů neustále roste, s čímž souvisí i jejich pronikání do stále více oblastí průmyslu. To vyžaduje také vývoj více inteligentních metod programování, které ovšem může být obtížné implementovat. Tato práce se bude zabývat řízením robota UR5e z prostředí ROS, které unifikuje programovací prostředí robotů a usnadňuje aplikaci přídavných SW nástrojů a integraci dodatečného HW.

Cíle bakalářské práce:

– Proveďte rešerši z oblasti využití Robotického operačního systému (ROS) a projektu ROS Industrial
– Popište balíčky MoveIt a Gazebo
– Zprovozněte řízení robota UR5e z balíčku MoveIt
– Vytvořte laboratorní úlohu, ve které robot uchopí objekt na základě dat z hloubkového senzoru
– Ověřte funkčnost navrženého řešení úlohy na reálném robotu

Seznam doporučené literatury:

Robot Operating System (ROS). 707. Cham: Springer International Publishing. ISBN 9783319549262. ISSN 1860-949X. Dostupné z: doi:10.1007/978-3-319-54927-9

CORKE, Peter. Robotics, Vision and Control. 118. Cham: Springer International Publishing, 2017. ISBN 9783319544120.

GREPL, Robert. Kinematika a dynamika mechatronických systémů. Brno: Akademické nakladatelství CERM, 2007, 158 s. : il. ; 26 cm. ISBN 978-80-214-3530-8.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2022/23

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

ABSTRAKT

Cílem bakalářské práce je popsat Robotický operační systém (ROS) a jeho využití. Dále vytvořit laboratorní úlohu, kde robot UR5e na základě dat získaných z hloubkové kamery, navede koncový člen manipulátoru, tzv. gripper, na uchopení předmětu v prostoru. V práci je popsán ROS a jeho rozšiřující balíčky MoveIt a Gazebo, které jsou především vhodné pro řízení a simulace manipulátorů a robotů obecně. V závěru práce je popsáno zprovoznění HW s ROS a postup při návrhu uchopovacího algoritmu a jeho testování na laboratorní úloze – nalezení a uchopení barevných kostek v prostoru a jejich následné sestavení na sebe.

Klíčová slova

ROS, MoveIt, UR5e, gripper, hloubková kamera

ABSTRACT

The aim of the bachelor's thesis is to describe the Robotic Operating System (ROS) and its use. Next, create a laboratory task where the UR5e robot, based on data obtained from the depth camera, guides the end effector of the manipulator, so-called gripper, to grasp an object in space. The thesis describes the ROS and its extension packages MoveIt and Gazebo, which are mainly suitable for controlling and simulating manipulators and robots in general. At the end of the thesis, the implementation of HW with ROS and the procedure for designing a grasping algorithm and testing it on a laboratory task – finding and grasping coloured cubes in space and their subsequent assembly on top of each other are described.

Key words

ROS, MoveIt, UR5e, gripper, depth camera

BIBLIOGRAFICKÁ CITACE

ZDENĚK, Marcel. *Řízení robota UR5 z Robotického operačního systému (ROS)*. Brno, 2023. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/149465. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. 51s. Vedoucí práce Filip Radil.

PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci na téma Řízení robota UR5 z Robotického operačního systému (ROS) vypracoval samostatně s použitím odborné literatury a pramenů, uvedených v seznamu, který tvoří přílohu této práce.

Datum

Marcel Zdeněk

PODĚKOVÁNÍ

Děkuji tímto panu Ing. Filipu Radilovi za vedení, ochotu, trpělivost, cenné připomínky a rady, které mi poskytl při vytváření této závěrečné práce.

OBSAH

1 ÚVOD

Současný trend automatizace skoro ve všech činnostech člověka se postupně dostává nejen do oblastí průmyslové výroby, laboratoří různých odvětví, ale i do vývoje prototypových řešení. Robotický operační systém (ROS) umožňuje vyvíjet vlastní řídící SW pro propojení libovolného množství zařízení a pro jeho variabilitu je ve velké míře využíván ve vývoji a ve výzkumných institucích.

Cílem této práce je popsat využití ROS a jeho rozšiřující balíčky MoveIt a Gazebo, které jsou především vhodné pro řízení a simulace manipulátorů a robotů obecně. Dále také tato práce by měla vytvořit základy pro další ROS projekty v mechatronické laboratoři VUT FSI v Brně. V neposlední řadě také popsat komunitní projekt ROS Industrial, který je nedílnou součástí při užívání ROS s automatizační a robotickou technikou.

Praktickou částí práce je zprovoznění řízení robotického manipulátoru UR5e pomocí ROS balíčku MoveIt. Toto robotické rameno od firmy Universal Robots je studentům k dispozici v mechatronické laboratoři VUT FSI v Brně. K robotickému rameni je také třeba do ROS implementovat ovládání gripperu a hloubkové kamery. Pro ovládání robotického manipulátoru je nutné vytvořit uchopovací algoritmus, který pomocí získaných dat z hloubkové kamery uchopí objekt v prostoru. Vytvořený algoritmus otestovat na navržené laboratorní úloze (nalezení a uchopení barevných kostek v prostoru a jejich následné sestavení na sebe) a zhodnotit výsledky testování.

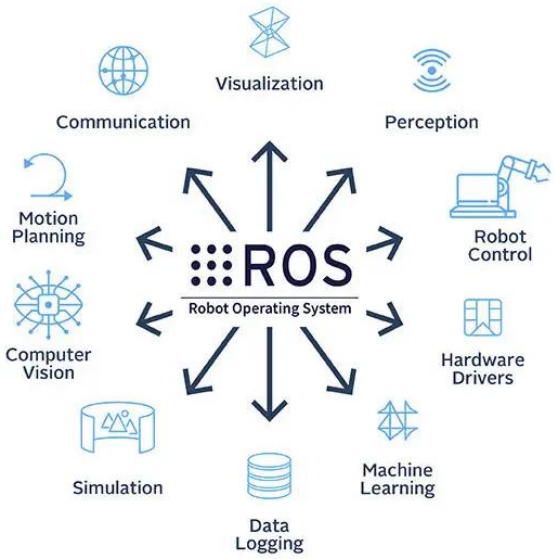
2 Robotický operační systém (ROS)

2.1 Struktura a využití ROS

ROS (*Robotic Operating System*), je *open source*, meta operační systém poskytující uživatelům nástroje a knihovny pro vývoj robotických aplikací a nástrojů pro správu robotických senzorů a akčních mechanismů. Dále umožňuje komunikaci mezi jednotlivými moduly v robotickém systému, vizualizaci dat a simulaci robotických systémů.

ROS je meta operační systém, protože umožňuje vytvořit vyšší komunikační strukturu nad skupinou zařízení, na kterých je provozován. ROS také poskytuje i řadu funkcí, které přicházejí se standardním operačním systémem, jako je abstrakce hardwaru, ovládání zařízení, implementace funkcí, předávání zpráv mezi procesy a správa balíčků. Procesy jsou znázorněny v grafové architektuře, kde ke zpracování dochází v uzlech, které mohou přijímat, odesílat a multiplexovat data senzorů, včetně řízení, stavu, plánování a dalších [1].

ROS byl původně vyvinut v roce 2007 na Stanfordově univerzitě pro výzkumné účely a později byl uvolněn jako *open source* projekt pod licencí BSD (*Berkeley Software Distribution*). ROS se provozuje primárně na systémech založených na platformě Unix (např. Linux nebo Mac OS X). ROS je těmto platformám velmi podobný, co se týče distribucí, protože ROS distribuce jsou také verzované sady balíčků [2], [3].



Obr. 1: Příklady využití ROS [4]

ROS je modulární sada balíčků, avšak jako celek ho lze rozdělit do 3 základních vrstev: souborový systém, výpočetní a grafická vrstva a komunita ROS [5].

2.1.1 Vrstva souborového systému

Souborový systém primárně spravuje části ROS nacházející se na disku:

- **Balíčky** jsou úplným základem souborové vrstvy v ROS. V balíčku můžou být uloženy ROS *runtime* procesy, což jsou také uzly, knihovny ROS, sady dat, konfigurační soubory nebo cokoli jiného, co je užitečně organizováno společně. Balíčky jsou nejelementárnější položkou na sestavení a následné vydání v ROS.

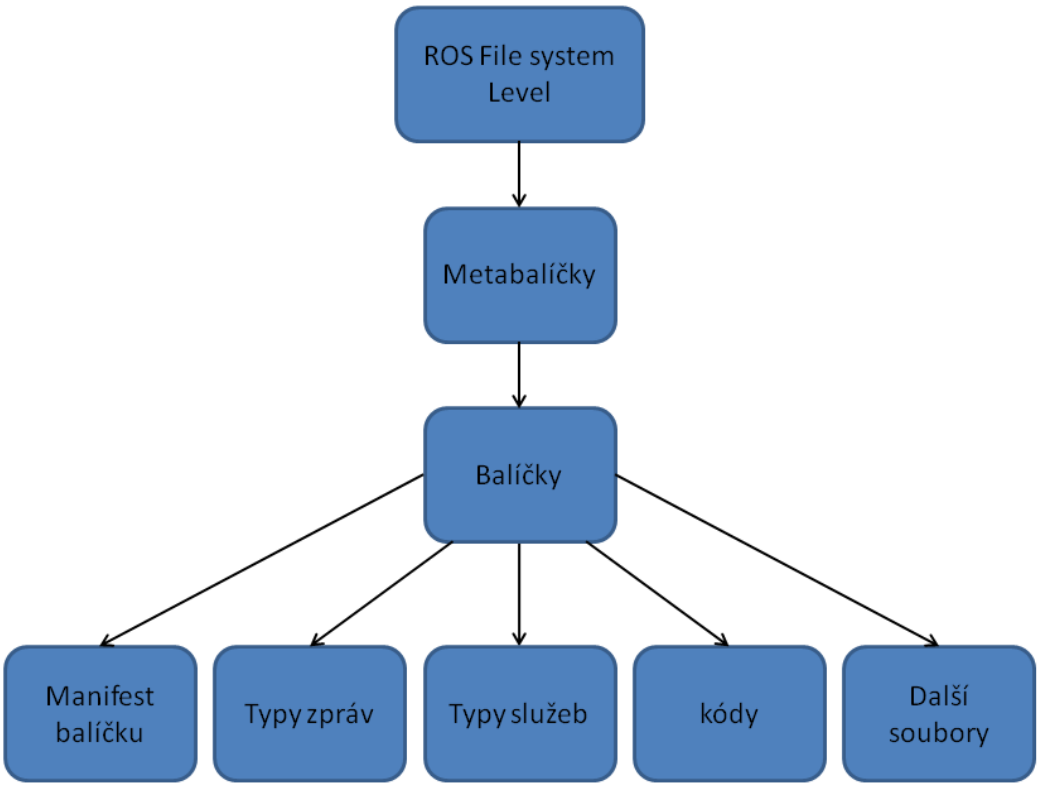
- **Metabalíčky** jsou specializované objemnější balíčky, které slouží pouze k reprezentaci skupiny dalších souvisejících balíčků.

- **Manifest balíčku** je soubor ve tvaru *názevbalíčku.xml*, poskytující základní data (*metadata*) o balíčku, včetně jeho názvu, verze, popisu, autora a licenčních informací. Dále také obsahuje potřebná data a informace pro *build*, *run* a test závislostí na jiných balíčcích.

- **Uložiště** obsahují kolekce balíčků, které sdílejí společný systém VCS (*Version Control System*). Toho se využívá hlavně při verzování projektů na webových uložištích GitHub nebo GitLab, kde vývojáři mohou pomocí VCS sdílet a vydávat verze pomocí nástroje *caktin bloom*. Omezujícím faktorem uložišť je, že mohou obsahovat pouze jeden balíček nebo metabalíček.

- **Typy zpráv** definují datové struktury zpráv, které jsou odesílané v ROS. Tyto definice jsou uložené v podadresáři balíčku.

- **Typy služeb** definují datové struktury zpráv typu požadavek a odpověď, které jsou odesílané v ROS. Tyto definice jsou opět uložené v podadresáři balíčku [5].



Obr. 2: Struktura souborového systému

2.1.2 Výpočetní grafická vrstva

Druhou a zároveň nejdůležitější vrstvou je výpočetní a grafická vrstva. Jedná se o peer-to-peer síť ROS procesů, které společně zpracovávají data a také zařizují správnou funkci ROS. Základními částmi této vrstvy jsou:

- **Uzly** jsou základní procesní buňky, které buď zpracovávají příchozí data, nebo jen provádí zadané výpočty. Zároveň mezi sebou mohou komunikovat pomocí témat či služeb, které zajišťuje Master a parametrický server. Uzly jsou napsány pomocí klientské knihovny, jako je *roscpp* (*C++ library for ROS*) nebo *rospy* (*python library for ROS*).

  ROS je navržen od základu jako modulární systém, proto jsou uzly určeny k vykonávání minoritních úkonů, vůči hlavnímu celku. To znamená, že řídicí systém složitějšího robota bude obsahovat mnoho uzlů. Jeden uzel například ovládá laserový dálkoměr, druhý uzel ovládá motory pohonu robota, třetí uzel provádí lokalizaci, čtvrtý uzel provádí plánování cesty, pátý uzel poskytuje grafický pohled na systém a tak dále.

  Použití uzlů v ROS poskytuje celému systému několik výhod. Systém je celkově robustnější, protože selhání jednotlivých uzlů nijak drasticky neovlivní chod celého programu. Dále dělení činností do jednotlivých uzlů také snižuje komplikovanost kódu, ve srovnání s monolitickými systémy.
  Všechny běžící uzly mají specifický název v grafu, který je jednoznačně identifikuje od zbytku systému [5], [6].

- **Zprávy** slouží ke komunikaci mezi uzly. Zpráva je jasně daná datová struktura obsahující typovaná pole. Jsou podporovány standardní primitivní typy (int, float, boolean, atd.), stejně jako pole primitivních typů. Zprávy mohou obsahovat libovolně vnořené struktury a pole [5].

- **Témata** jsou pojmenované sběrnice, přes které si uzly vyměňují zprávy. Témata mají anonymní sémantiku – vydavatel / odběratel, která odděluje produkci informací od jejich spotřeby.

  Princip komunikace 2 uzlů přes téma: uzel č. 1 je vydavatel a publikuje zprávu, obsahující data, na konkrétně pojmenované téma. Uzel č. 2 má zájem o data na tomto tématu, takže se přihlásí k jeho odběru. Pro jedno téma může existovat více souběžných vydavatelů a odběratelů a jeden uzel může publikovat a odebírat více témat současně. Obecně si vydavatelé a odběratelé navzájem neuvědomují svojí existenci.

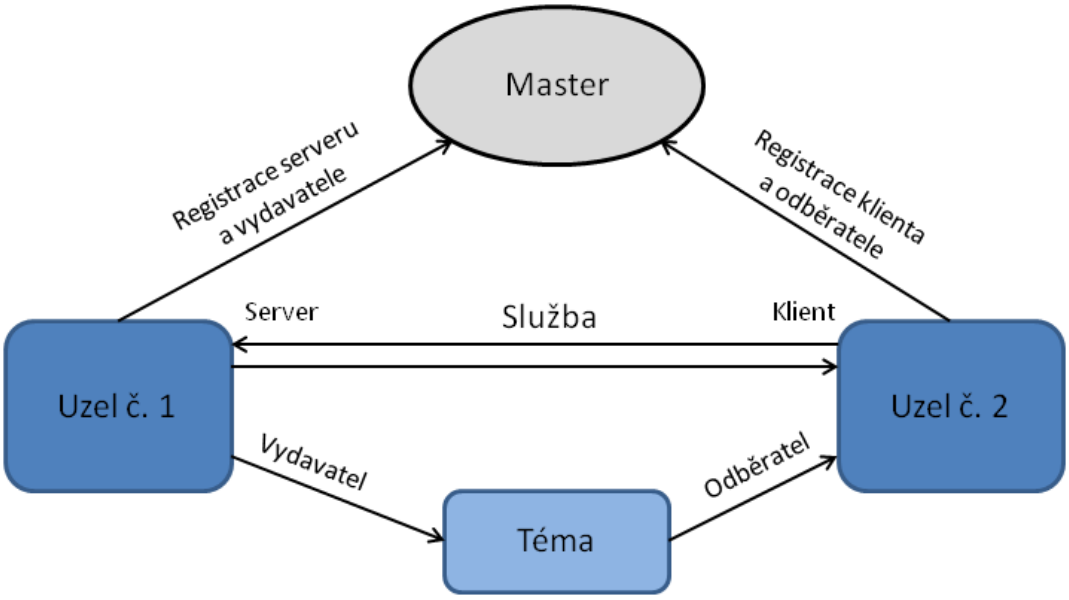
  Témata jsou určena pro jednosměrnou, streamovanou komunikaci. Uzly, které potřebují provádět vzdálená volání procedur, tj. přijímat odpověď na vyslaný požadavek, by měly místo toho používat služby, protože umožní efektivnější využití komunikace [5], [7].

- **Služby** je označení pro komunikaci stylem – požadavek / odpověď. Jsou definovány pomocí dvojice struktur zpráv: jedna pro požadavek a druhá pro odpověď. Poskytující uzel (server) nabízí službu pod jménem, kdy druhý uzel (klient) službu využívá odesláním zprávy s požadavkem a čeká na odpověď. Klientské knihovny ROS obecně prezentují tuto interakci programátorovi, jako by šlo o vzdálené volání procedury [5].

- **Master** poskytuje registraci jmen a vyhledávání ostatních ve výpočtové grafické vrstvě. Bez Mastera by uzly nemohly najít stejný komunikační kanál (téma nebo službu), protože ukládá a poskytuje registrační informace o většině prvcích umístěných ve výpočtové grafické vrstvě [5].

- **Parametr server** ukládá data pomocí klíče na centrálním místě. Ukládají se na něm registrační data všech prvků v této vrstvě a uzly si na něj mohou ukládat a načítat data za běhu. Je viditelný pro všechny prvky, takže kdokoliv může zkontrolovat aktuální stav systému a případně pozměnit jeho konfiguraci. V současné době je součástí Mastera [5], [8].

- **Tašky** jsou formáty, který umožňují ukládání a přehrávání dat ROS zpráv. Tašky zprostředkovávají ukládání dat například ze senzorů, které může být obtížné sbírat, ale jsou nezbytné pro vývoj a testování algoritmů [5].



Obr. 3: Ukázka ROS komunikace přes téma a službu

2.1.3 ROS komunita

Poslední a velmi podstatnou vrstvou je komunita ROS, která je klíčovou složkou úspěchu ROS na poli automatizace a robotiky. Aktivní a velmi rozvinutá vývojářská komunita zařídila, že jde v ROS použít mnoho rozšiřujících nástrojů a knihoven, díky kterým je tvorba programů mnohem jednodušší (není potřeba znovu vymýšlet kolo). Trochu na to určitě má i vliv fakt, že celá platforma ROS a její rozšiřující balíčky jsou *open source*. Kupříkladu balíček MoveIt, pomocí kterého lze ovládat více jak 100 robotů od mnoha výrobců, nebo velice známá knihovna OpenCV, pomocí které se řeší problematika strojového vidění. Do komunitní vrstvy také určitě patří komunitní fóra ROS Wiki nebo ROS Answers, na kterých lze najít nejrůznější návody, tutoriály nebo řešení konkrétního problému [5].

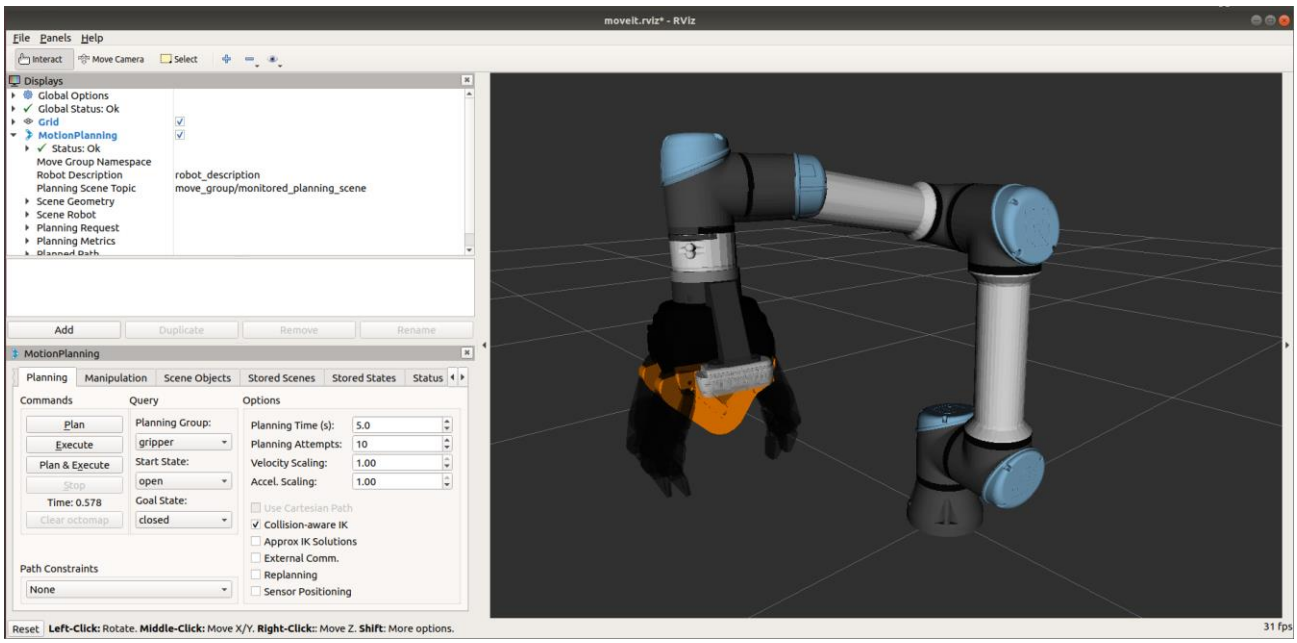
2.1.4 RViz

Rviz (*ROS Visualization*) je výkonný 3D vizualizační nástroj pro ROS. Umožňuje uživateli zobrazit simulovaný model robota, zaznamenávat informace ze senzorů robota a přehrávat zaznamenané informace ze senzorů. Díky vizualizaci toho, co robot vidí, myslí a dělá, může uživatel ladit robotovou aplikaci od senzorových vstupů až po plánované (nebo neplánované) akce.

Rviz dokáže vizualizovat data z 3D senzorů (např. hloubkových kamer nebo laserů) jako *Point Cloud* nebo hloubkovou mapu. Data 2D senzorů z RGB kamer nebo 2D laserových dálkoměrů lze v RViz prohlížet jako obrazová data, viz Obr. 22.

Pokud skutečný robot komunikuje s ROS, RViz zobrazí aktuální konfiguraci robota na virtuálním modelu robota. Témata ROS budou zobrazena jako živá reprezentace na základě dat ze senzorů publikovaných libovolnými kamerami, infračervenými senzory a laserovými skenery, které jsou součástí systému robota. Tato funkce se může využít pro vývoj a ladění úlohy [9].

Do RViz je možné také importovat další různé funkce z jiných používaných platforem. Například z balíčku MoveIt lze importovat funkce *MotionPlanning*, *PlanningScene*, *Trajectory* či *RobotState*, viz Obr. 4 a Obr. 13.



Obr. 4: Vizualizační prostředí RViz s manipulátorem UR5e

2.1.5 Využití ROS mimo výrobní průmysl

ROS se stal populární platformou pro výzkumné účely a pro vzdělávání v oblasti robotiky. Díky *open source* kódu a rozsáhlé komunitě uživatelů je ROS stále více využíván v průmyslových aplikacích a zlepšuje tak efektivitu a produktivitu v různých oblastech. Avšak jeho uplatnění lze najít i v neprůmyslovém prostředí. Jednou z nich je například autonomní řízení osobních automobilů, které se v posledních letech stává populárním trendem. Významní výrobci automobilů, jako je např. BMW, Nissan a Tesla, vkládají finanční prostředky a poskytují podporu společnostem, které se zaměřují na tuto oblast. Důvody, proč některé firmy v tomto případě využívají ROS, je snadná vizualizace a zpracování dat ze senzorů a na základě nich generování vhodné zpětné vazby pro pohonné jednotky a další důležité komponenty. Užití

ROS má však i své úskalí, programy musí být vytvořeny s naprostou přesností, aby nedocházelo k výpadkům, které by mohly narušit koordinaci mezi jednotlivými procesy. Další nevýhodou je samotná bezpečnost ROS, a proto většina firem musí své programy doplnit o prvky kyberbezpečnosti [10], [11].

Další neprůmyslové odvětví, kde se ROS využívá je kupříkladu kurýrní robot od firmy Relay Robotics. Jejich robot je kompletně ovládán pomocí ROS a sám výrobce tvrdí, že využívají podpory široké ROS komunity a zpět do ní přispívají. Kurýrní robot je primárně určen na hotelovou dovážku zásilek od recepce na pokoj a rozvážku potřebných zásilek ve zdravotnických zařízeních [12], [13].

V neposlední řadě ROS využívá i NASA například ve vesmírné humanoidní robotice, kdy byl pomocí ROS a knihovny MoveIt řízen humanoidní robot s označením R2 (Robonaut 2). Dále bude NASA využívat ROS2 v lunárním roveru Viper, který má v listopadu 2024 přistát na Měsíci. Použitý ROS2 software není v současnosti k dispozici, ale jakmile mise skončí, tým VIPER hodlá uvolnit ROS2 software roveru pro všeobecné použití. Použití platformy ROS umožnilo rychlý, agilní a nákladově efektivní způsob vývoje softwarových systémů roveru, který může být přínosem i pro budoucí vozítka na Měsíci i mimo něj. [14], [15], [16].

Obr. 5: NASA lunární rover Viper [16]

2.2 Projekt ROS Industrial

Projekt ROS Industrial je komunitní *open source* projekt výzkumníků a vývojářů soustředících se okolo také *open source* platformy ROS. Byl založen v roce 2012 s cílem zlepšit nasazení automatizační techniky v průmyslu a podpořit vývoj aplikací s vysokou úrovní automatizace a bezpečnosti. Z velké části se soustředí na kombinování relativně silných stránek ROS se stávajícími průmyslovými technologiemi (tj. spojuje vysokou úroveň funkčnosti ROS s nižší úrovní řídicích jednotek průmyslových robotů, která zajišťuje spolehlivost a bezpečnost).

Komunitní projekt se dále zaměřuje na vytvoření standardizovaných rozhraní pro průmyslový hardware, který usnadňuje jeho integraci do existujících průmyslových systémů. Tyto standardizované rozhraní jsou vytvořeny tak, aby byly interoperabilní (tj. kompatibilní a efektivní zároveň) s různými typy průmyslových systémů a robotů. To umožňuje průmyslovým firmám snížit náklady na integraci a zlepšit efektivitu procesů.

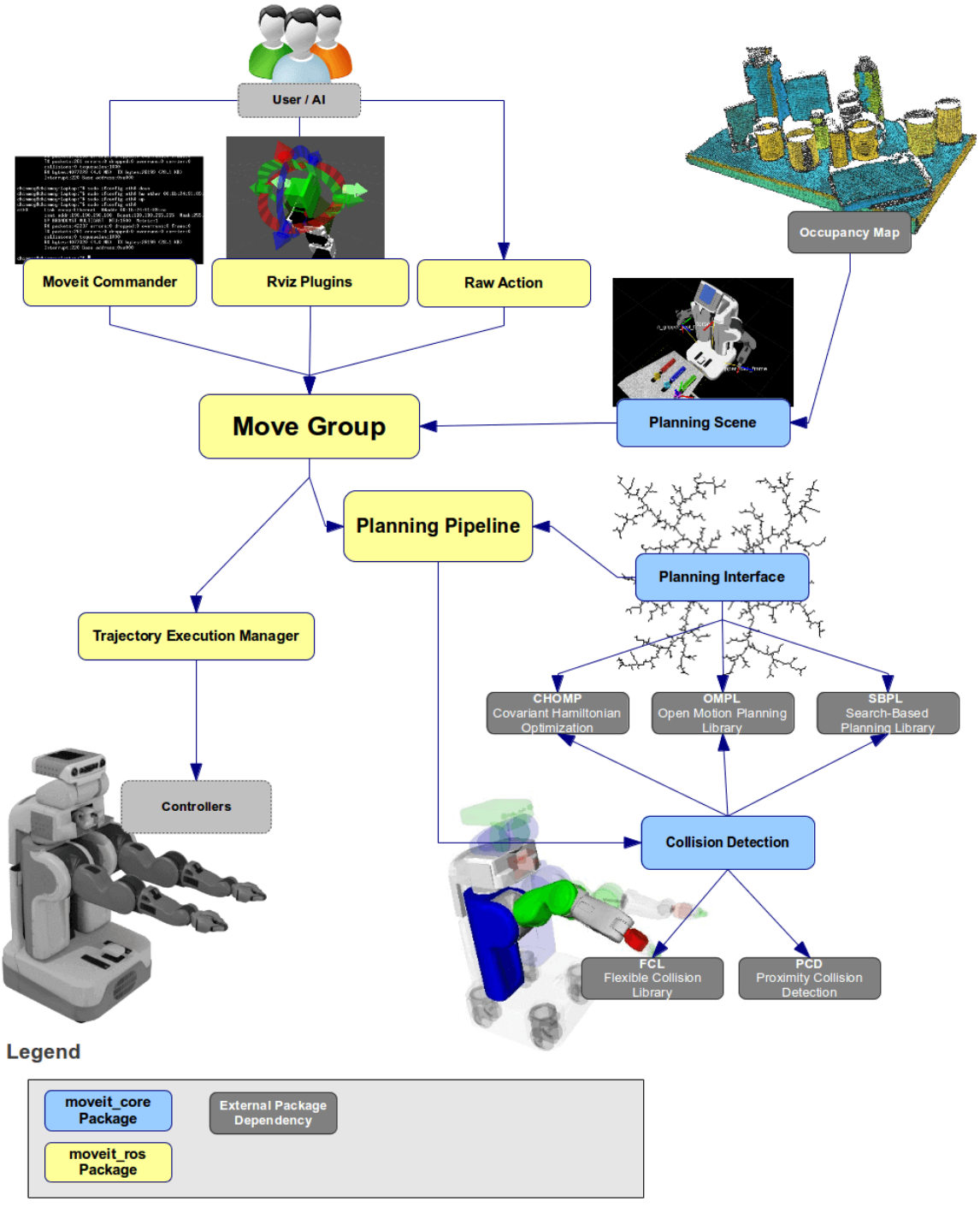
ROS Industrial na svém GitHubu také poskytuje nástroje pro vývoj aplikací s vysokou úrovní automatizace a bezpečnosti. Tyto aplikace umožňují průmyslovým firmám snížit náklady na práci a zlepšit produktivitu. Například ROS Industrial umožňuje vytváření aplikací pro výrobu, které dokážu řídit výrobní linky, sledovat procesy a řešit výjimky, jako jsou poruchy strojů nebo senzoriky. Díky tomu se zvyšuje kvalita výroby a snižují se náklady na práci.

Tento projekt je podporován řadou průmyslových partnerů a je široce využíván ve výzkumných a průmyslových aplikacích po celém světě. Ve svých 3 konsorciích (v Evropě, Americe a Asii) poskytuje nejen technickou, ale i komunitní podporu například v podobě různých školení na ROS atd. Konsorcia také zajišťují spolupráci firem na průmyslových projektech s cílem vyvinout nové schopnosti pro ROS Industrial [17], [18], [19].

Obr. 6: Logo projektu ROS Industrial [20]

2.3 Balíček MoveIt

MoveIt je základním prvkem pro řízení robotických ramen pomocí ROS. Hlavní funkcionalitou balíčku MoveIt je plánování pohybu pomocí různých plánovacích algoritmů, které řeší inverzní kinematiku robota. Důležitou součástí je také detekce sebe kolizí a vyhýbání se softwarově definovaným překážkám.



Obr. 7: Blokový diagram MoveIt architektury [21]

2.3.1 Architektura

Uživatelské rozhraní

Nejdůležitějším prvkem MoveIt architektury je uzel *move_group*, který spojuje všechny jeho jednotlivé komponenty dohromady, viz Obr. 7 a podrobnější komunikace na Obr. 8. Uživatel k *move_group* uzlu může přistupovat pomocí 3 základních uživatelských rozhraní:

- V programovacím jazyce C++ pomocí balíčku *move_group_interface*
- V Python pomocí balíčku *moveit_commander*
- V RViz pomocí *MotionPlanning* rozšíření

Konfigurace

Move_group potřebuje z ROS parametrického serveru získat 3 druhy konfiguračních informací:

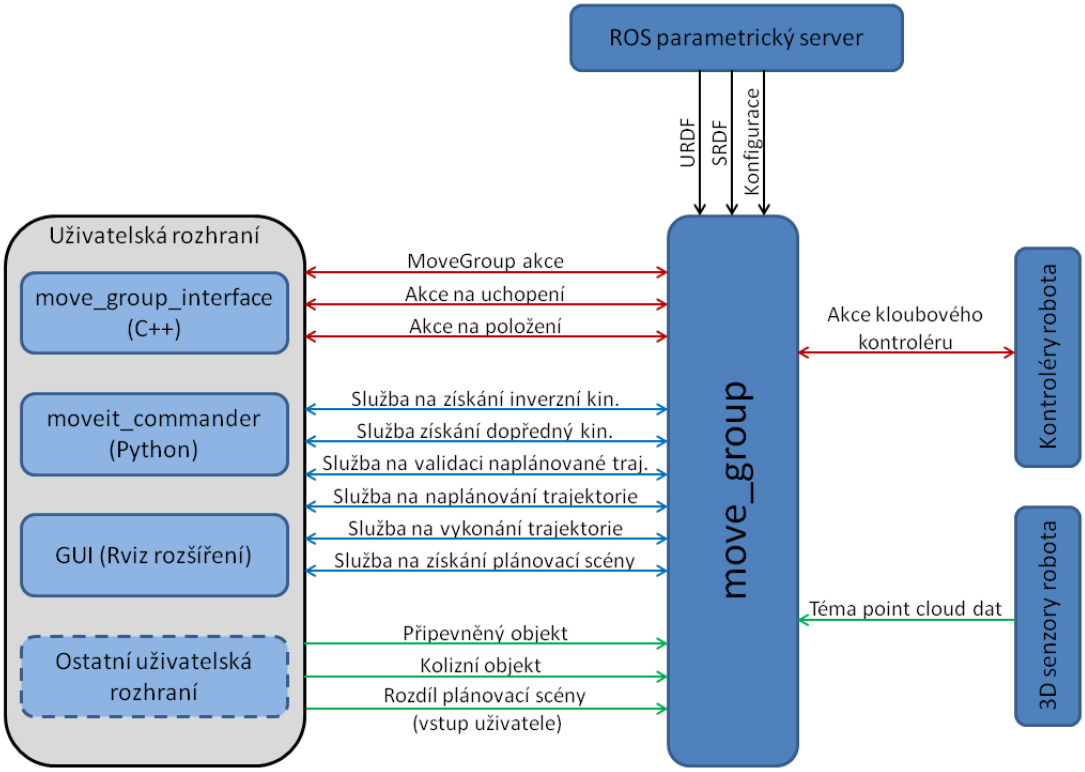
- URDF – *move_group* uzel hledá na parametrickém serveru *robot_description*, aby získal URDF (*Unified Robot Description Format*) ovládaného robota.
- SRDF – move_group uzel hledá parametr *robot_description_semantic* na parametrickém serveru, aby získal SRDF (*Semantic Robot Description Format*) ovládaného robota. SRDF soubor je obvykle vytvořen (jen jednou) uživatelem pomocí nástroje MoveIt Setup Assistant.
- MoveIt konfigurace – *move_group* vyhledá na parametrickém serveru konfiguraci specifickou pro MoveIt včetně kloubových limitů, kinematiky, plánování pohybu a dalších informací. Konfigurační soubory pro tyto komponenty jsou automaticky generovány asistentem nastavení MoveIt. Jsou uloženy v konfiguračním adresáři uvnitř konfiguračního balíčku MoveIt daného robota.

Rozhraní robota

Uzel *move_group* komunikuje s robotem prostřednictvím ROS témat a akcí (velmi podobné službám, ale jedná se o opakovanou dlouhodobější komunikaci). Informace o aktuální poloze kloubů robota získává odebíráním */joint_states* tématu. *Move_group* může odebírat více pozičních témat najednou. Požadavek na pohyb zadává kontroléru robota skrze *FollowJointTrajectoryAction* akci, pomocí které předá robotovi požadovanou trajektorii, ovšem výhradně jen v kloubových souřadnicích (dopředná kinematika). Důležité je také zmínit, že *move_group* je zde akční klient, který se připojuje na akční server robota.

Dále uzel monitoruje transformační data na základě, kterých má globální informace o pozici robota nebo o pozicích více robotů (mimo jiné). K udržování plánovací scény používá *Planning Scene Monitor*, který představuje svět a aktuální stav robota. Stav robota může zahrnovat jakékoli objekty připojené k robotu (které nese robot), které jsou považovány za pevně připojené k robotu. Zároveň může získávat i *Point Cloud* data z hloubkové kamery, popřípadě data z jiné senzoriky robota.

Architektura MoveIt je strukturována tak, aby byla snadno rozšiřitelná – jednotlivé funkce jako uchopení, položení, kinematika, plánování pohybu jsou ve skutečnosti implementovány jako samostatné pluginy se společnou základní třídou. Zásuvné moduly jsou konfigurovatelné v ROS prostřednictvím parametrů v souborech s příponou *.yaml* a pomocí knihovny ROS *pluginlib*. Většina uživatelů však nebude muset konfigurovat zásuvné moduly uzlu *move_group*, protože jsou automaticky nakonfigurovány ve spouštěcích souborech generovaných nástrojem MoveIt Setup Assistant [21].



Obr. 8: Komunikační struktura move_group uzlu [21]

2.3.2 Plánovače trajektorií

MoveIt spolupracuje s plánovači pohybu skrze rozhraní určené pro zásuvné moduly. To umožňuje MoveIt komunikovat a používat různé plánovače pohybu z více knihoven, takže MoveIt je snadno rozšiřitelný. Rozhraní k plánovačům pohybu je prostřednictvím akcí nebo služeb (nabízené *move_group* uzlem). Výchozí plánovače pohybu pro *move_group* jsou konfigurovány pomocí OMPL (*Open Motion Planning Library*) a rozhraní MoveIt k OMPL pomocí nástroje MoveIt Setup Assistant. Další plánovače, které jsou standardně dostupné, jsou *Pilz industrial motion planner*, CHOMP (*Covariant Hamiltonian optimization for motion planning*) nebo STOMP (*Stochastic Trajectory Optimization for Motion Planning*) [21], [22], [23].

Žádost o pohybový plán

Žádost o naplánování pohybu specifikuje, jaký pohyb má plánovač připravit. Plánovač pohybu je žádán, aby přesunul robotické rameno na jiné místo (v kloubním prostoru) nebo koncový člen do nové pozice. Ve výchozím nastavení jsou zaškrtnuty kolize (včetně sebe kolizí a nadefinovaných připojených objektů). Volbu plánovače lze také určit pomocí parametrů *planning_pipeline* a *planner_id* a omezení, která musí plánovač při pohybu zkontrolovat – vestavěná omezení poskytovaná MoveIt jsou kinematická omezení:

- Omezení polohy – omezují polohu spoje (*link*) tak, aby ležel v oblasti prostoru
- Omezení orientace – omezují orientaci spoje tak, aby ležela v určených mezích naklánění, rozteče nebo vybočení
- Omezení viditelnosti – omezují bod na spojnici tak, aby ležel v kuželu viditelnosti pro konkrétní senzor

- Vazby spojů – omezují spoj, aby ležel mezi dvěma hodnotami
- Uživatelem zadaná omezení – lze také zadat vlastní omezení pomocí uživatelem definovaného zpětného volání.

Výsledek plánu pohybu

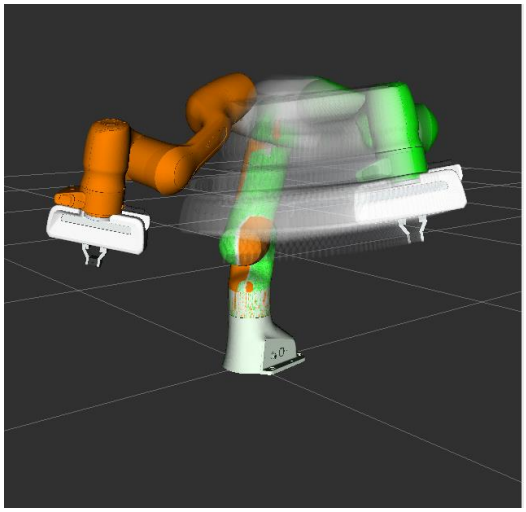
Uzel *move_group* vygeneruje požadovanou trajektorii jako odpověď na uživatelskou žádost o pohybový plán. Tato trajektorie přesune rameno (nebo jakoukoli skupinu kloubů) na požadované místo, pomocí maximální rychlosti a zrychlení (pokud jsou specifikovány) [21].

OMPL (*Open Motion Planning Library*)

OLPM je open source knihovna pro plánování pohybu, která primárně implementuje randomizované plánovače pohybu. MoveIt se integruje přímo s OMPL a používá plánovače pohybu z této knihovny jako svou primární/výchozí sadu plánovačů. Plánovače v OMPL jsou abstraktní; tj. OMPL nemá žádnou koncepci robota [21].

CHOMP (*Covariant Hamiltonian optimization for motion planning*)

Jedná se o postup optimalizace trajektorie založený na gradientu, který dokáže řešit mnoho běžných problémů s plánováním pohybu (Ratliff a další, 2009c). Většina vícedimenzionálních plánovačů pohybu rozdělují generování trajektorie do různých fází plánování a optimalizace. Ovšem tento algoritmus využívá kovariantního gradientu a funkčního gradientu ve fázi optimalizace k návrhu algoritmu plánování pohybu založeného výhradně na optimalizaci trajektorie. Pokud je trajektorie neuskutečnitelná, CHOMP reaguje podle okolního prostředí, aby rychle vytáhl trajektorii z kolize a současně optimalizoval dynamické veličiny, jako jsou kloubové rychlosti a zrychlení. Rychle konverguje k hladké trajektorii bez kolize, kterou lze na robotu efektivně provádět [22].

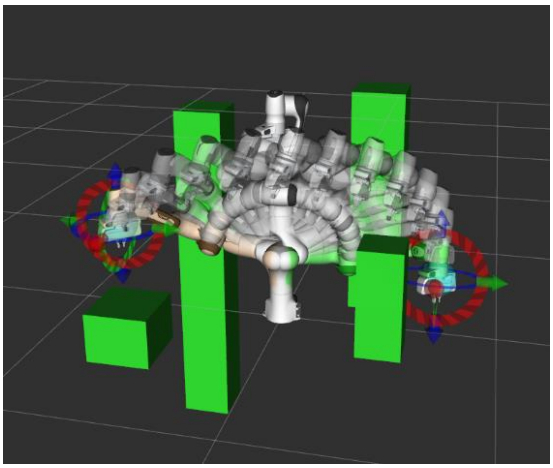


Obr. 9: Ukázka plánování trajektorie pomocí CHOMP [22]

STOMP (*Stochastic Trajectory Optimization for Motion Planning*)

STOMP je pravděpodobnostní optimalizační *framework* (Kalakrishnan a daší, 2011). Plánovač vytváří hladký průběh trajektorie bez kolize v rozumných časech. Tento přístup se opírá o generování mnoha trajektorií k prozkoumání prostoru kolem počáteční (pravděpodobně neproveditelné) trajektorie, které jsou poté kombinovány za účelem vytvoření aktualizované

trajektorie s nižšími náročností. Funkce náročnosti, která je založená na kombinaci daných překážek a hladkosti optimalizace, je v každé iteraci optimalizována. Pro konkrétní optimalizační algoritmus, který je používán, nejsou vyžadovány žádné informace o gradientu, takže obecná náročnost, pro kterou nemusí být dostupná derivace (např. náročnost odpovídající omezením a točivým momentům motoru), mohou být zahrnuty do náročností funkce [23].



Obr. 10: Ukázka plánování trajektorie pomocí STOMP [23]

2.3.3 Detekce kolizí

Kontrola kolize v MoveIt se konfiguruje uvnitř plánovací scény pomocí objektu *CollisionWorld*. MoveIt je nastaven tak, že uživatelé se nikdy nemusí starat o to, jestli a jak probíhala kontrola kolizí. Kontrola kolize v MoveIt se provádí hlavně pomocí balíčku FCL (*Flexible Collision Library*) – primární knihovny CC (*Collision Checking*) MoveIt [21].

Kolize objektů

MoveIt podporuje kontrolu kolizí pro různé typy objektů, jako jsou například:

- Sítě
- Primitivní tvary - např. krabice, válce, kužely, koule a roviny
- Octomapy – Octomapa jako objekt lze přímo použít pro kontrolu kolize

Povolená kolizní matice (ACM - *Allowed Collision Matrix*)

Kontrola kolizí je velmi časově a technicky náročná operace, která často představuje téměř 90 % zatížení výpočetní techniky během plánování pohybu. ACM binárně kóduje hodnotu odpovídající potřebě zkontrolovat kolizi mezi dvojicí těles (které mohou být na robotu nebo mimo něj). Pokud je hodnota odpovídající dvěma tělesům v ACM rovna 1, udává to, že kontrola kolize mezi dvěma tělesy není nutná. To by se stalo, pokud by například obě tělesa byla vždy tak daleko od sebe, že by ke srážce nikdy nedošlo [21].

2.3.4 Časová parametrizace

Plánovače pohybu obvykle generují pouze trajektorie, se kterými nejsou spojeny žádné informace o čase. MoveIt obsahuje několik algoritmů pro zpracování trajektorií, které mohou pracovat na těchto drahách a generovat trajektorie, které jsou správně časově parametrizované

a zohledňují limity maximální rychlosti a zrychlení uložené na jednotlivé klouby. Tyto limity jsou načítány ze speciálního konfiguračního souboru *joint_limits.yaml*, který je specifikován pro každého robota. Konfigurační soubor je volitelný a přepíše jakékoli limity rychlosti nebo zrychlení z URDF.

Doporučený algoritmus od ledna 2023 je *TimeOptimalTrajectoryGeneration* (TOTG). Pro tento algoritmus výrobce upozorňuje, že robot musí začínat a končit v klidu. Ve výchozím nastavení je časový krok TOTG 0,1 sekundy [24].

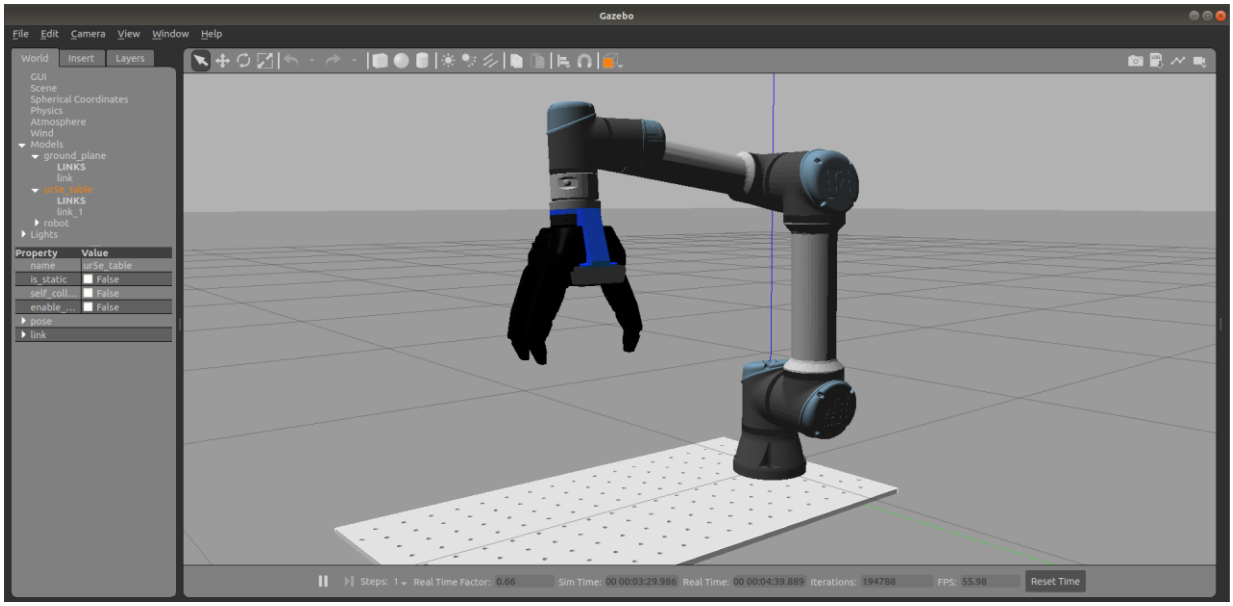
2.4 Balíček Gazebo

Gazebo je *open source* 3D simulátor robotiky, který velmi kvalitně dokáže simulovat fyziku reálného světa. Dokáže simulovat jak gravitaci, tak atmosféru, vítr nebo magnetické pole. Toho je docíleno pomocí 4 vysoce výkonných fyzikálních enginů: ODE, DART, Simbody a Bullet. Simulátor umožňuje rychlé a efektivní testování algoritmů.

Vývoj simulačního prostředí Gazebo (venkovní altán) pro venkovní prostředí začal na podzim roku 2002 na Univerzitě v Jižní Kalifornii. Původní záměr vycházel z potřeby simulovat roboty ve venkovním prostředí za různých podmínek. Záměr byl pouze rozšířit využití simulátoru pro vnitřní prostředí. V současné době většina uživatelů využívá simulátor Gazebo k simulacím i vnitřního prostředí, protože Gazebo představuje dva v jednom.

V roce 2009 byl do Gazebo integrován do ROS, od té doby se stal jedním z primárních nástrojů používaných v komunitě ROS. V roce 2012 se společnost Open Source Robotics Foundation (OSRF) oddělila od Willow Garage a stala se správcem projektu Gazebo. Od verze 1.9 není závislý na ROS a lze ho instalovat jako samostatný balík [25].

Komunikace s ROS je realizována pomocí pluginů. Díky tomu, že mají komunikační uzly stejné rozhraní jako celý ROS, je možné vytvořit simulaci, která se na venek tváří a chová jako skutečný hardware.



Obr. 11: Prostředí simulátoru Gazebo s manipulátorem UR5e

3 Postup řešení a výsledky řešení

V této praktické části je popsán způsob a možnosti ovládání robotického ramene UR5e pomocí ROS a také implementace Robotiq 3-Finger gripperu a Intel RealSense D435 hloubkové kamery do ROS. Následně je popsána laboratorní úloha, postup jejího řešení a zjištěné výsledky z jejího testování.

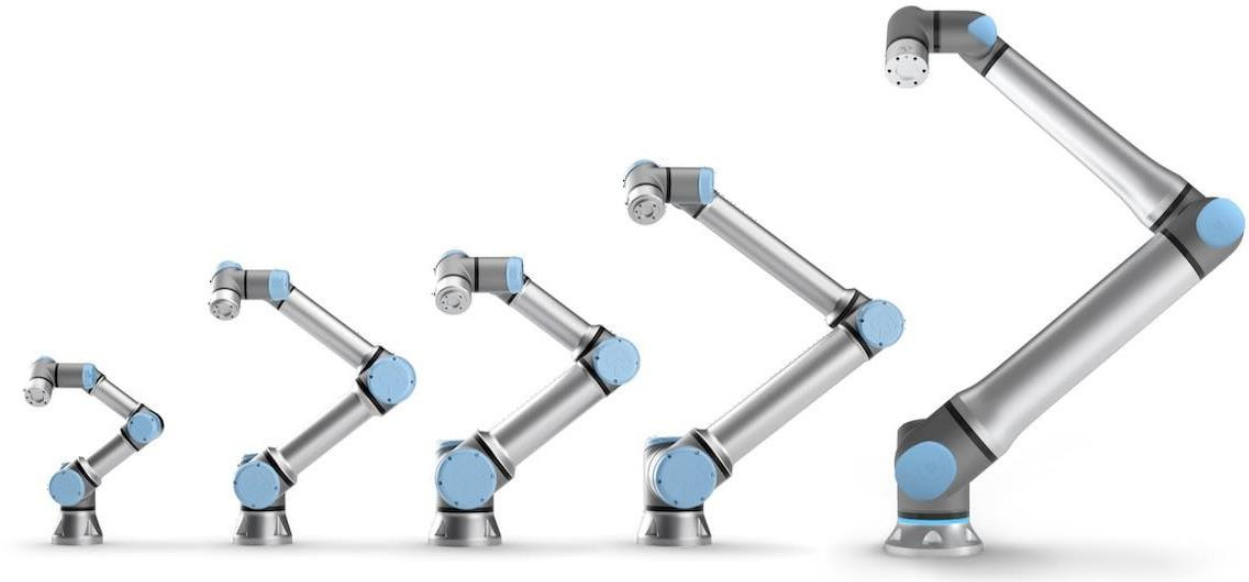
3.1 Implementace HW zařízení do ROS

V mechatronické laboratoři FSI VUT v Brně jsou k robotickému rameni UR5e dostupné 2 grippery, Onrobot RG2 a Robotiq 3-Finger gripper (dále jen Robotiq gripper). Pro laboratorní úlohu byl zvolen Robotiq gripper, pro jeho jednodušší implementaci do ROS (gripper Onrobot RG2 je také možné ovládat pomocí ROS, ale jeho implementace do ROS je náročnější). Jako hloubkový senzor byla na základě doporučení vedoucího bakalářské práce zvolena hloubková kamera Intel RealSense D435, která je kompatibilní s ROS.

Z důvodů kompatibility ROS s ostatním HW je nutné použít jako hostitelský operační systém Ubuntu starší verzi 18.04.6 Bionic Beaver a k němu odpovídající starší verzi ROS Melodic. Oba tyto hostitelské operační systémy je vhodné provozovat jako virtuální stroje, z důvodu kompaktnosti (možnost přenesení celého systému do jiného počítače) a jednoduchosti vytváření případných záloh. Při implementaci vyšší verze hostitelského operační systému Ubuntu 20.04.6 Focal Fossa pro ROS Noetic byla zjištěna nekompatibilita s ostatním HW.

3.1.1 Ovládání UR5e pomocí ROS

Řízení a ovládání robotických ramen od firmy Universal Robots pomocí ROS bylo nejdříve realizováno pomocí neoficiálního driveru (*ur_modern_driver*), který si velká komunita projektu ROS Industrial sama vytvořila. Avšak pak si tento driver převzala samotná firma Universal Robots a spravuje ho dodnes (pod názvem *ur_robot_driver*), stejně jako mnoho ostatních výrobců automatizační techniky, kteří jsou součástí komunitního projektu ROS Industrial [26].



Obr. 12: Kolaborativní UR roboti z řady e-Series [27]

Základní specifikace robotického ramene UR5e

Kolaborativní robotické rameno UR5e je jeden typ kobota z nejnovější řady e-Series od firmy Universal Robots. Firma roboty označuje UR (3e,5e,16e,10e a 20 ve stejném pořadí z leva na Obr. 12), kde typové číslo odpovídá maximální hmotnosti manipulované zátěže (kromě UR10e, jehož max. zátěž činí 12,5 kg). Pro tuto bakalářskou práci bylo použito robotické rameno o maximální nosnosti 5 kg. Hmotnost samotného robota byla jen 20,6 kg.

Robotické rameno je také kolaborativní díky tomu, že má tříosý silový a momentový senzor, přičemž silový senzor má měřitelný rozsah 50 N s rozlišením 2,5 N a přesností na 4 N. Momentový senzor má měřitelný rozsah 10 Nm s rozlišením 0,04 Nm a přesností na 0,3 Nm. Certifikace kolaborativního chování je zajištěna dle normy EN ISO 10218.

Jeho manipulační dosah činí až 850 mm s přesností opakovatelnosti pohybu ±0,03 mm (s plným zatížením, dle normy ISO 9283) při průměru základny pouhých 149 mm. Rameno má 6 otočných kloubů, které se mohou otáčet maximální rychlostí 180°/s. Samotný koncový člen se pak může pohybovat maximální rychlostí až 1 m/s [28].

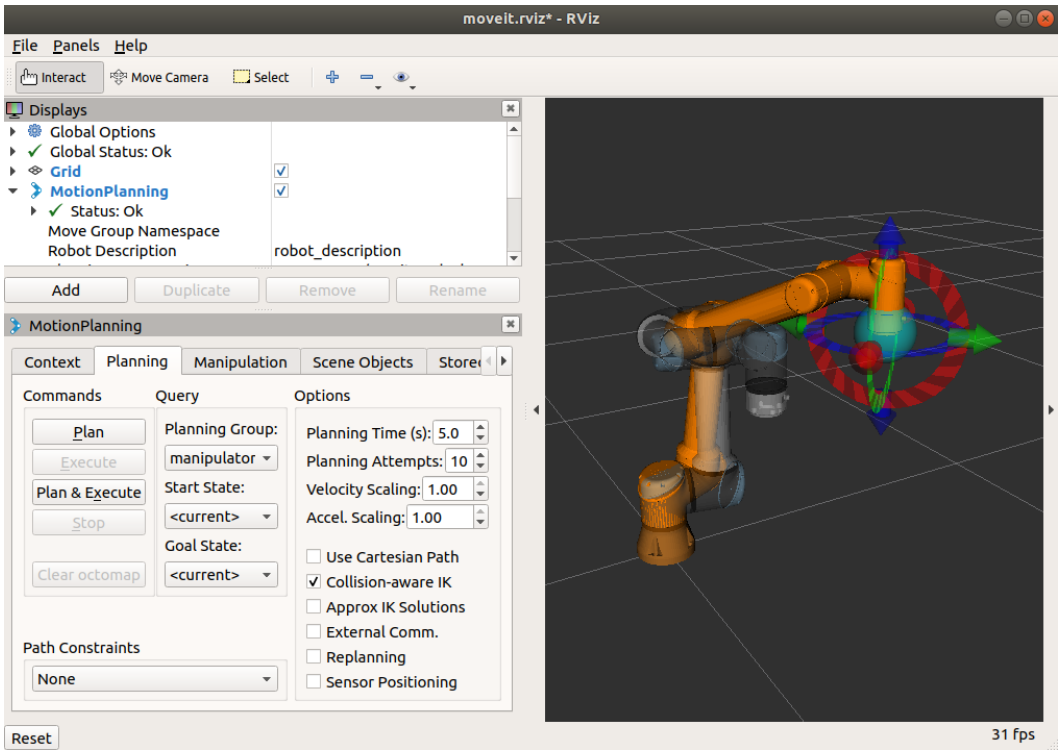
Způsob komunikace a ovládání robotického ramene UR5e

ROS driver pro UR robotická ramena komunikuje s robotem přes UTP ethernet kabel pomocí síťového protokolu TCP/IPv4. Uvnitř *polyskope* programu musí být umístěn programový blok *External Control* s odpovídající IP adresou hostitelského systému a síťový port musí být zvolen 50002. Potom robotické rameno s UR driverem může komunikovat přes následující 4 síťové porty:

- 50001 – **Rezervní port**, který driveru umožní přímou komunikaci mezi ním a řídící jednotkou robota.
- 50002 – **Port na posílání skriptů** využívá driver při nabídce rozhraní pro příjem UR_Script programu. Pokud se robot nemůže připojit k tomuto portu, *External Control* program se okamžitě zastaví.
- 50003 – **Port na posíláni trajektorie** použije driver při komunikaci pro předávání dat o trajektorii. Trajektorie je předána prostřednictvím kloubových souřadnic pro nastavení jednotlivých kloubů.
- 50004 – **Port na skriptové příkazy** otevře driver v případě potřeby předání skriptu příkazů do robota [26].

Nominální komunikační frekvence je 500 Hz [28]. Výrobce také doporučuje implementovat do hostitelského systému *real-time kernel* (obzvlášť při ovládání robota z řady e-Series), aby nedocházelo k výpadkům kontrolérů a vnitřních ROS komponent [26].

Pro jednoduché řízení (např. ruční za pomocí RViz prostředí, Obr. 13) robotického ramene pomocí ROS, stačí pouze dva metabalíčky: *Universal_Robots_ROS_Driver* [26] a *universal_robot* [29]. Tyto balíčky, již v sobě obsahují základní potřebné funkcionality MoveIt *framework*, ale pro komplexnější řízení je potřeba doplnit ještě několik rozšiřujících MoveIt balíčků.



Obr. 13: Ukázka ručního ovládání UR5e pomocí nástroje RViz

Popis hlavních možností použití UR ROS driveru

Pomocí Universal Robots balíčků do ROS lze řídit většinu UR robotů (kromě nové UR20). K tomu je také k dispozici velké množství kloubových a kartézských kontrolérů, které jsou jak pozičního, tak rychlostního charakteru. Avšak s MoveIt je kompatibilní jen kloubový kontrolér, podrobněji popsáno v kapitole 2.3.1. Součástí balíčků jsou také virtuální dvojčata všech ovladatelných robotických ramen, které jsou určeny do simulačního prostředí Gazebo nebo případně jen pro vizualizaci v RViz.

Tyto ROS balíčky ale také nabízí velké množství dalších funkcionalit, počínaje těmi nejzákladnějšími jako je získávání dat o aktuálním statusu robota, polohu všech kloubů robotického ramene, nebo také uživatelem nastavenou rychlost v *polyskope* robota. Dále nabízí zajištění komunikace s koncovým nástrojem, který je připojený v konektoru koncového člene robota.

Jak už bylo zmíněno v základních specifikacích robotického ramene, manipulátor je vybaven tříosým silovým a momentovým senzorem, jehož data je také možné získat v podobě sil a momentů působících na koncový člen. UR ROS drivery ale neposkytují možnost silového řízení, avšak na základě získávaných silových a momentových dat ho lze implementovat [26].

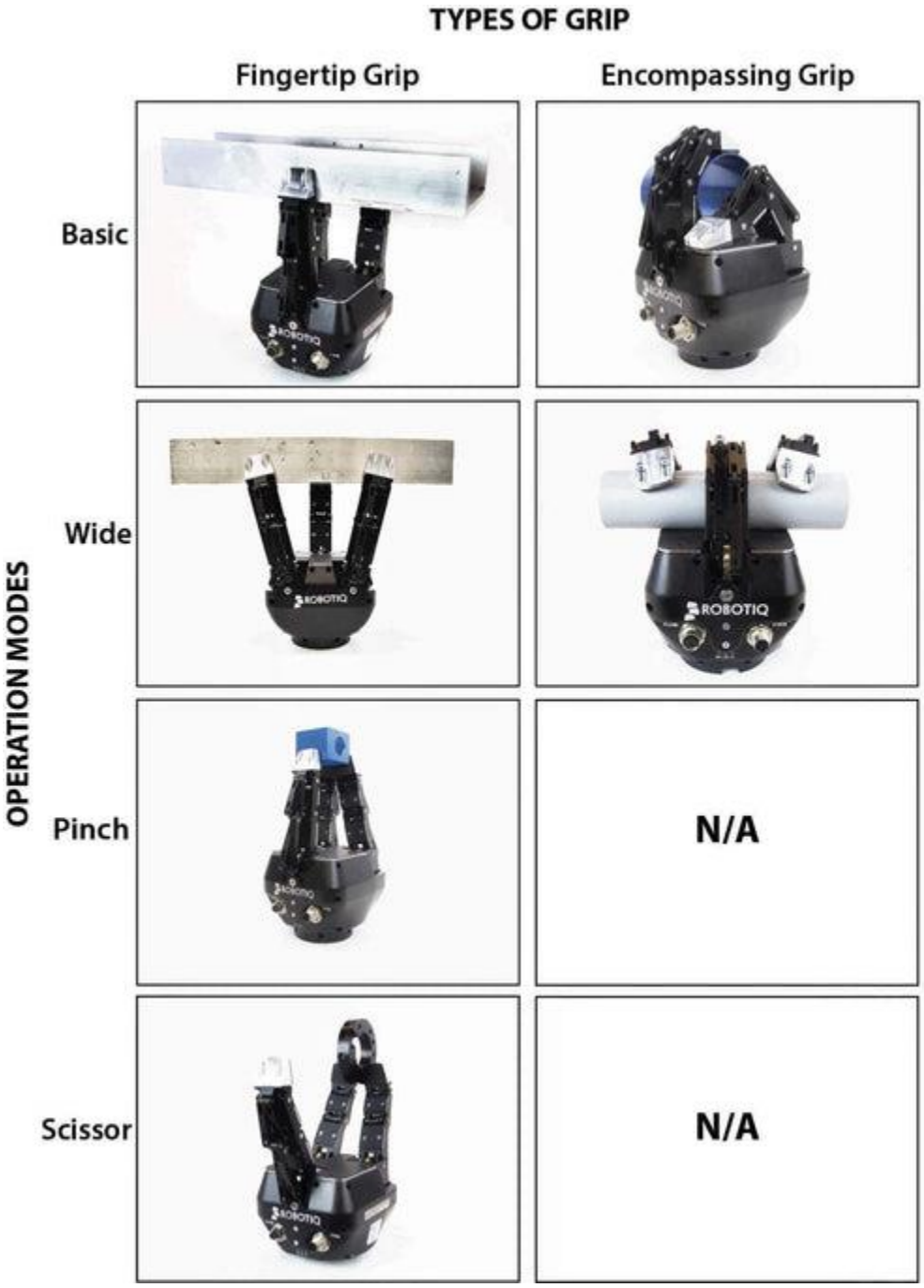
3.1.2 Ovládání Robotiq 3-Finger gripperu z ROS

Možnost řízení Robotiq gripperů pomocí ROS opět umožnil velký komunitní projekt ROS Industrial, což je na jejich webových stránkách zdůrazněno [30].

Obr. 14:Robotiq 3-Finger gripper [31]

Základní specifikace Robotiq 3-Finger gripperu

Rozsah rozevření čelistí gripperu v základní nemodifikované konfiguraci je až 155 mm (po fyzické úpravě lze docílit až 167 mm). Samotný váží 2,3 kg a je schopen uchopit předměty o váze až 10 kg, avšak z důvodu tření a bezpečnostního koeficientu, výrobce doporučuje uchopovat předměty jen do hmotnosti 2,5 kg. Úchopná síla a rychlost každého jednotlivého prstu lze specificky nastavit, síla od 15 do 60 N s krokem 0,175 N a rychlost od 22 do 110 mm/s s krokem 0,34 mm/s [31]. Díky svým třem prstům umožňuje gripper množství způsobů uchopení, a navíc díky paralelogramovýmu provedení pohybů jednotlivých článků prstů se prsty při uchopení přizpůsobí tvaru daného objektu, viz Obr. 15.



Obr. 15: Možnosti uchopení pomocí Robotiq gripperu [31]

Způsob komunikace a ovládání s gripperu

Gripper podporuje velkou škálu komunikačních protokolů (EtherNet/IP, Modbus TCP/IP, MODBUS RTU, PROFINET, EtherCAT, DeviceNet, a CANopen), avšak každý gripper je nakonfigurován vždy jen na 2 tyto komunikační protokoly (MODBUS RTU je vždy jeden z nich) [31]. Gripper v mechatronické laboratoři je nakonfigurován na komunikační protokol Modbus TCP/IP (vyznačeno na nalepeném štítku), s tím souvisí i použitý druh komunikačního kabelu. Jedná se o speciální ethernetový kabel, který má na jedné straně RJ-45 konektor, který se zapojí do počítače s ROS, a na straně druhé je závitový konektor M12,

který se připojí do gripperu. Z důvodu své komplexnosti a velikosti, gripper pro svoji správnou funkci potřebuje nejen komunikační kabel, ale také externí napájení 24 V.

Ovládání gripperu v ROS je realizováno pomocí Robotiq metabalíčku [32], který dokáže ovládat i dvouprsté Robotiq grippery. Součástí balíčku jsou i virtuální dvojčata těchto gripperů určené pro simulační prostředí Gazebo nebo jen vizualizaci v RViz prostředí, kde u tříprstého gripperu je potřeba dodatečně dopočítávat polohu jednotlivých článků prstů podle polohy jeho motorů (také součástí balíčku). Zároveň pro každý gripper jsou vytvořeny jednoduché uzly pro čtení všech stavů (Obr. 16) a zápis základních pohybových požadavků (Obr. 17).

```
-----
3F gripper status interpreter
-----
gACT = 1: Gripper activation
gMOD = 2: Wide Mode
gGTO = 1: Go to Position Request
gIMC = 3: Activation and mode change are completed
gSTA = 3: Gripper is stopped. All fingers reached requested position
gDTA = 3: Finger A is at requested position
gDTB = 3: Finger B is at requested position
gDTC = 3: Finger C is at requested position
gDTS = 3: Scissor is at requested position
gFLT = 0: No Fault
gPRA = 150: Echo of the requested position for the Gripper (or finger A in individual mode): 150/255
gPOA = 150: Position of Finger A: 150/255
gCUA = 0: Current of Finger A: 0 mA
gPRB = 100: Echo of the requested position for finger B: 100/255
gPOB = 100: Position of Finger B: 100/255
gCUB = 0: Current of Finger B: 0 mA
gPRC = 200: Echo of the requested position for finger C: 200/255
gPOC = 200: Position of Finger C: 200/255
gCUC = 0: Current of Finger C: 0 mA
gPRS = 0: Echo of the requested position for the scissor axis: 0/255
gPOS = 24: Position of the scissor axis: 24/255
gCUS = 0: Current of the scissor axis: 0 mA
```

Obr. 16: Interpretátor všech stavů Robotiq 3-Finger gripperu

```
Simple 3F gripper Controller
-----
Current command: rACT = 0, rMOD = 0, rGTO = 0, rATR = 0, rPRA = 0, rSPA = 0, rFRA = 0
-----
Available commands

r: Reset
a: Activate
c: Close
o: Open
b: Basic mode
p: Pinch mode
w: Wide mode
s: Scissor mode
(0-255): Go to that position
f: Faster
l: Slower
i: Increase force
d: Decrease force
-->
```

Obr. 17: Jednoduchý kontrolér Robotiq 3-Finger gripperu

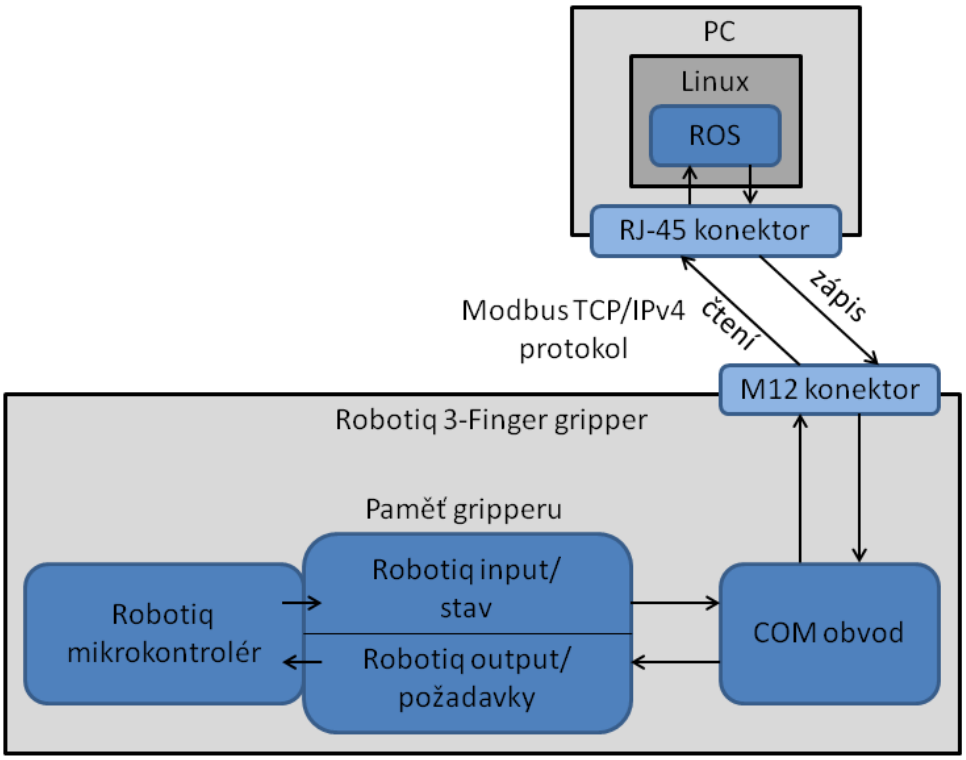
Komunikace přes Modbus TCP/IPv4 protokol probíhá tak, že si gripper a ROS neustále vyměňují až 32 bajtů dat s frekvencí 100 Hz (16 bajtů na každou stranu). Gripper zasílá ROS následující informace:

- Aktuální stav (aktivovaný/deaktivovaný, typ aktuálního uchopovacího módu, aktuálně v pohybu, otevřený/uzavřený, atd.)
- Informaci o uchopeném objektu (např. kolik prstů ho drží)
- Poruchový stav
- Zpětnou vazbu na zadanou polohu prstů
- Polohu a proud všech 4 motorů => všech 3 prstů jednotlivě (z proudu lze pak přepočítat výsledná uchopovací síla každého prstu).

Oproti tomu ROS zasílá gripperu následující požadavky:

- Požadovanou pozici všech prstů zároveň nebo každého jednotlivého zvlášť (může být s okamžitým vykonáním či jen se zadáním polohy a vyčkáním na vykonávací příkaz)
- Požadavek na uchopovací mód (základní, široký, úzký nebo nůžky)
- Nastavení rychlosti a uchopovací síly pro všechny prsty hromadně anebo opět každý pro každý zvlášť.

Komunikace mezi ROS a gripperem může probíhat na vysoké úrovni (např. prostředním prstem pohni rychlostí *x* do polohy *y* a gripper po chvíli může odpovědět s tím, že se prst pohnul, ale nedojel do zadané polohy, pohnul se jen do polohy *z*), protože gripper má vlastní interní kontrolér s pamětí, který se stará o veškerou regulaci rychlostí a předepsaných sil, viz blokové schéma na Obr. 18.



Obr. 18: Blokové schéma spojení a komunikace gripperu [31]

3.1.3 Intel RealSense D435

Společnost Intel není přímo součástí projektu ROS Industrial, ale jejich snaha o jednoduchost a univerzálnost použití jejich hloubkových kamer způsobila, že jsou nyní kompatibilní s více jak 10 různými vývojovými platformami, mezi nimiž je i ROS [33].

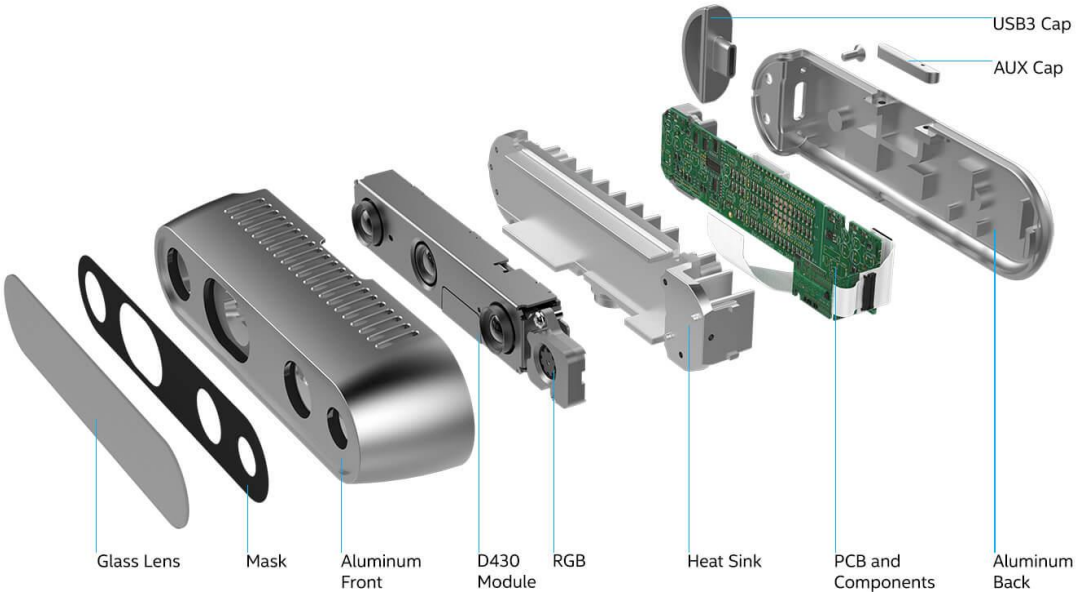
Obr. 19: Intel RealSense D435 [34]

Základní specifikace hloubkové kamery RealSense D435

Hloubková kamera RealSense D435 je prostorový senzor z řady D400 od firmy Intel. Pro tuto řadu je charakteristický Intel RealSense D4 procesor na zpracování obrazu (*Intel RealSense Vision Processor D4*). V prostorovém senzoru D435 je tento obrazový procesor v kombinaci s infračerveným, širokoúhlým, hloubkovým modulem D430 (*Intel RealSense Module D430*) a RGB kamerou, viz Obr. 20.

Hloubkový modul pracuje na principu stereoskopie. Je širokoúhlý, protože jeho zorné úhly jsou horizontálně 87° a vertikálně 58° (tj. diagonálně 95°). Maximální rozlišení hloubkového obrazce je až 1280 x 720 (HD), s maximálně 30 FPS (pro všechny nižší rozlišení je až 90 FPS). Jeho přesnost při měřené vzdálenosti do 2 metrů je méně jak 2% a minimální měřitelná vzdálenost je 280 mm (při maximálním rozlišení, se snižujícím rozlišením, klesá i minimální měřitelná vzdálenost).

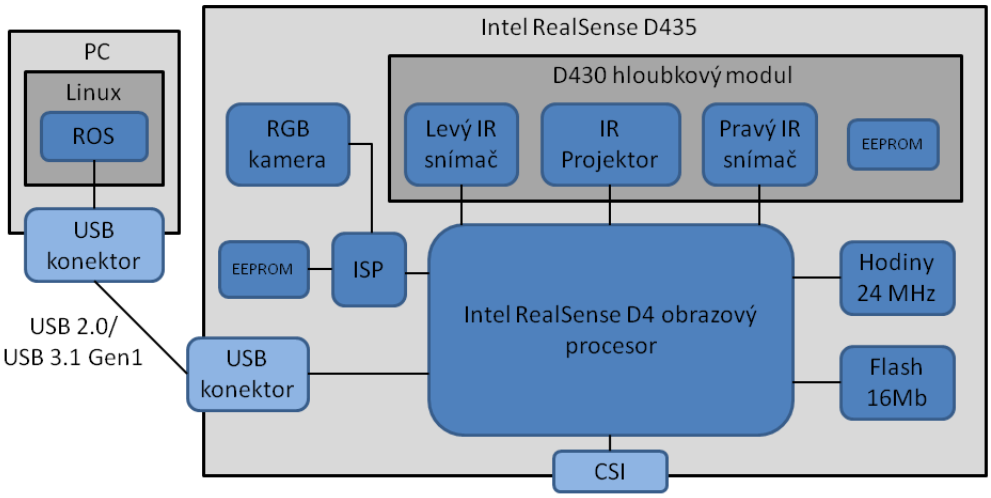
RGB kamera dokáže snímat obraz v rozlišení až 1920 x 1080 (Full HD) s maximálně 30 FPS (stejné FPS je i pro 1280 x 720, pro všechny nižší je až 60 FPS). Zorné úhly RGB kamery jsou horizontálně 69° a vertikálně 42° (tj. diagonálně 77°) [35].



Obr. 20: Jednotlivé komponenty hloubkové kamery RealSense D435 [33]

Způsob spojení a získávání dat z hloubkové kamery RealSense D435

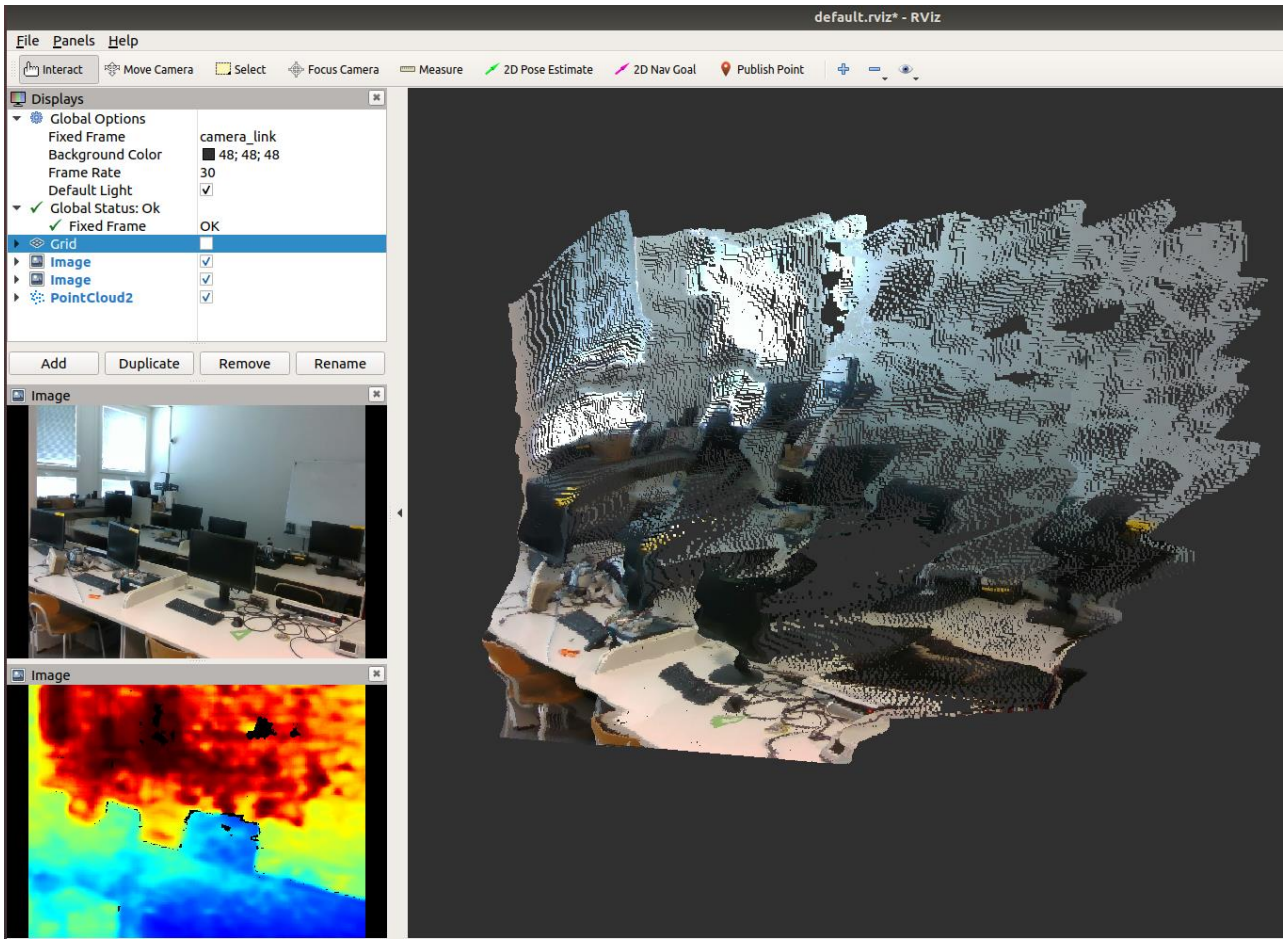
Hloubková kamera se připojuje pomocí USB kabelu nebo také lze využít CSI (*Camera Serial Interface*), viz Obr. 21. Pro získávání obrazu z kamery je potřeba importovat dva metabalíčky: realsense-ros a Intel RealSense SDK 2.0 (*Software Development Kit*) [36]. Dále Je zapotřebí zajistit, aby firmware v hloubkové kameře byl o jednu verzi starší, než je odpovídající verze k aktuálnímu RealSense SDK 2.0. Sobě odpovídající verze se ukázaly jako nefunkční.



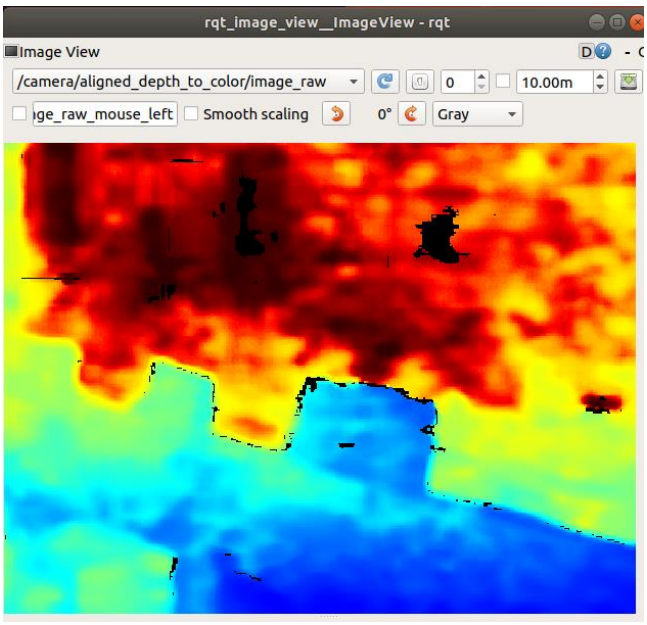
Obr. 21: Blokové schéma vnitří komunikace hloubkové kamery RealSense D435 [35]

Obrazová a prostorová data z hloubkové kamery jde získávat dohromady v podobě *Point Cloudu*, který je tvořen obrazovým procesorem RealSense D4 z obrazu RGB kamery a dat prostorových z hloubkového modulu. Ovšem tento procesor také umožňuje získat tyto data odděleně, obrazová data v např. RGB8, BGRA8 nebo YUYV kódování a hloubková data v podobě hloubkové mapy v Z16 kódování.

Vizualizace všech těchto dat lze pomocí nástroje RViz (Obr. 22), nebo užitím nástroje *rqt_image_view* (Obr. 23), který ale vizualizuje jen obrazové formáty.



Obr. 22: Vizualizace všech dat z hloubkové kamery pomocí nástroje RViz (vlevo obraz s barevně vykreslenou hloubkovou mapou a vpravo Point Cloud)

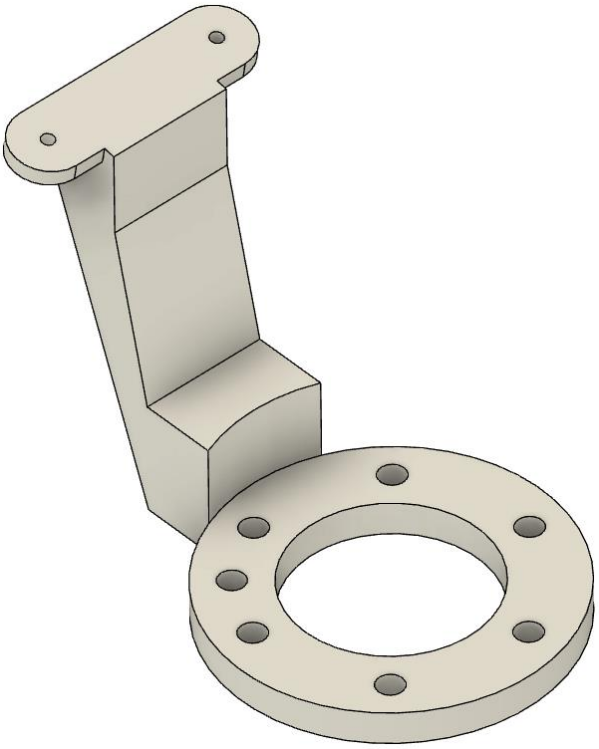


Obr. 23: Vizualizace dat pomocí nástroje rqt_image

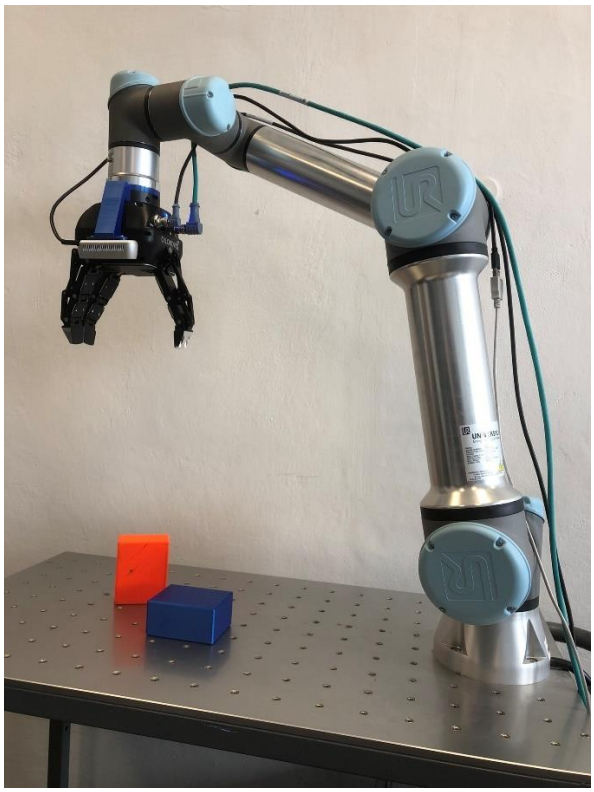
3.2 Realizace laboratorní úlohy

Hlavním cílem laboratorní úlohy je použít robotické rameno UR5e, Robotiq gripper a hloubkovou kamerou RealSense D435 s ROS v praxi. HW komponenty a jejich implementace do ROS jsou popsány v kapitole 3.1.

Při vytváření úlohy byl robot v konfiguraci, která zahrnuje veškerý popsaný HW, pojízdný stůl, a 3D vytištěný držák kamery (Obr. 24), který je upevněný mezi koncovým členem robotického ramene a gripperem. Kovová deska stolu představuje platformu, ke které je připevněno samotné robotické rameno, ale také funguje jako pracovní plocha při práci s různými objekty, které lze, v případě potřeby, k desce i připevnit, viz Obr. 25.



Obr. 24: 3D model držáku hloubkové kamery



Obr. 25: Použité uspořádání robota

3.2.1 Popis úlohy

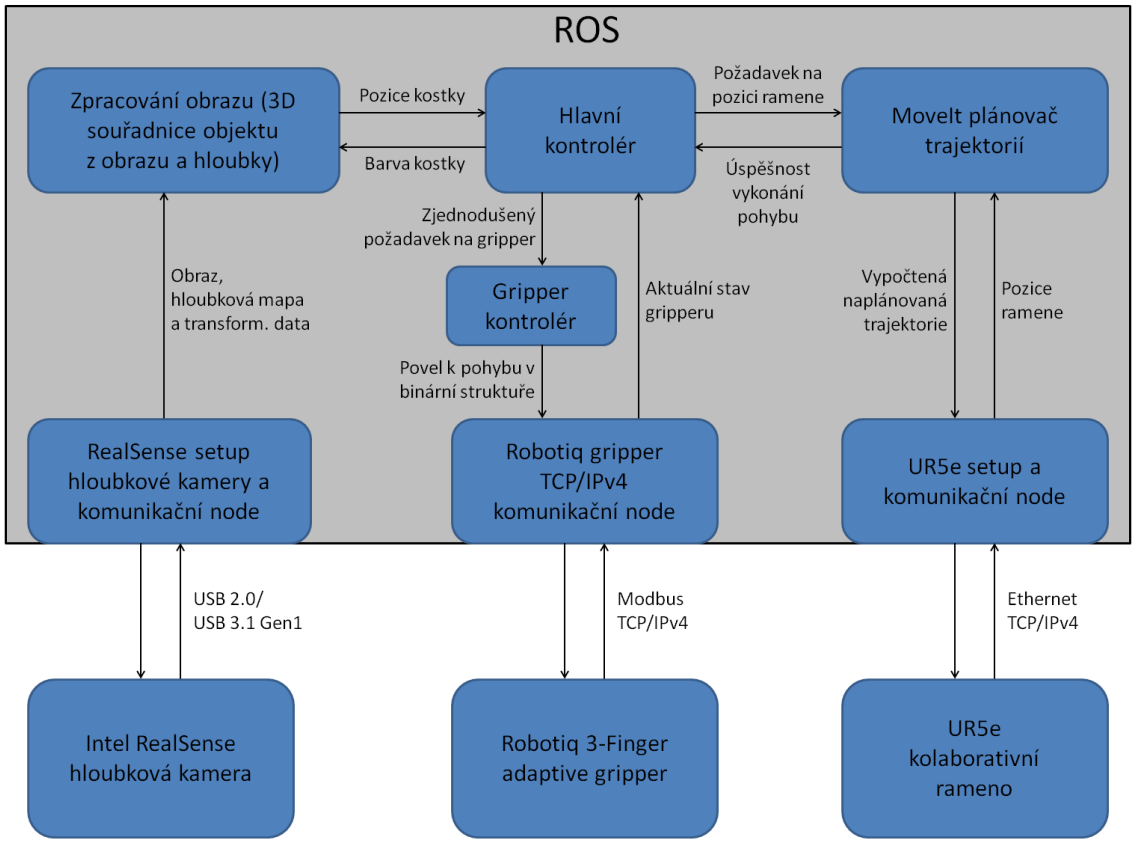
Laboratorní úloha má demonstrovat uchopení objektu v prostoru na základě dat z hloubkového senzoru. Podle tohoto požadavku byla navržena laboratorní úloha, kde robot pomocí dat z hloubkové kamery postupně uchopí barevné kostky a bude je pokládat na sebe na určeném místě. Kostky jsou nahodile rozloženy a orientovány na pracovní desce robota a zároveň jsou v zorném poli hloubkové kamery.

Přesný popis průběhu navržené úlohy:

Robot pomocí hloubkové kamery lokalizuje pozici objektu č. 1. Rozevřený gripper přesune nad objekt a po následném snížení gripperu do uchopovací výšky, uchopí objekt. Potom objekt přesune na pokládací místo, kde ho umístí na pracovní desku. Po uvolnění objektu robotické rameno pokračuje zpět do výchozí polohy, aby pomocí hloubkové kamery zaměřil pozici objektu č. 2.

Následný pohyb je stejný jako při manipulaci s objektem č. 1, s tím rozdílem, že objekt č. 2 bude položen na objekt č. 1. Jakmile robot postaví objekt č. 2 na objekt č. 1, vrací se opět do výchozí polohy. Při každém úchopu je ověřováno, zda byl objekt správně uchopen. Pokud tomu tak není, robot rozevře gripper, vrátí se do výchozí polohy a program je ukončen.

Takto navržená laboratorní úloha má za cíl otestovat řízení robotického ramene UR5e pomocí nadřazeného algoritmu, který využívá balíčku MoveIt. Dále také testuje přesnost získávání dat z hloubkové kamery a v neposlední řadě zpětnovazební ovládání Robotiq gripperu.



Obr. 26: Zjednodušené blokové schéma komunikace mezi jednotlivými uzly a zařízeními

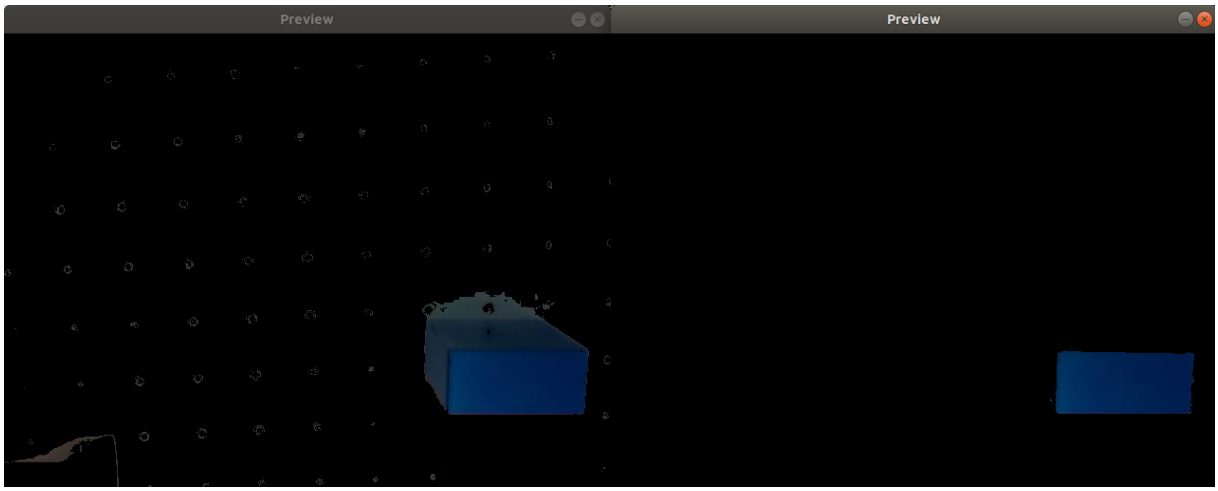
3.2.2 Zpracování obrazu

Obrazová data jsou zpracovávána ve 2 hlavních krocích, získání 2D souřadnic středu objektu a jeho natočení v barevném obraze a výpočet 3D souřadnic z hloubkové mapy a transformačních konstant hloubkové kamery. Pro správnou kombinaci těchto 2 kroků musí mít barevný obraz a hloubková mapa stejné rozlišení a obnovovací frekvenci.

Zpracování barevného obrazu

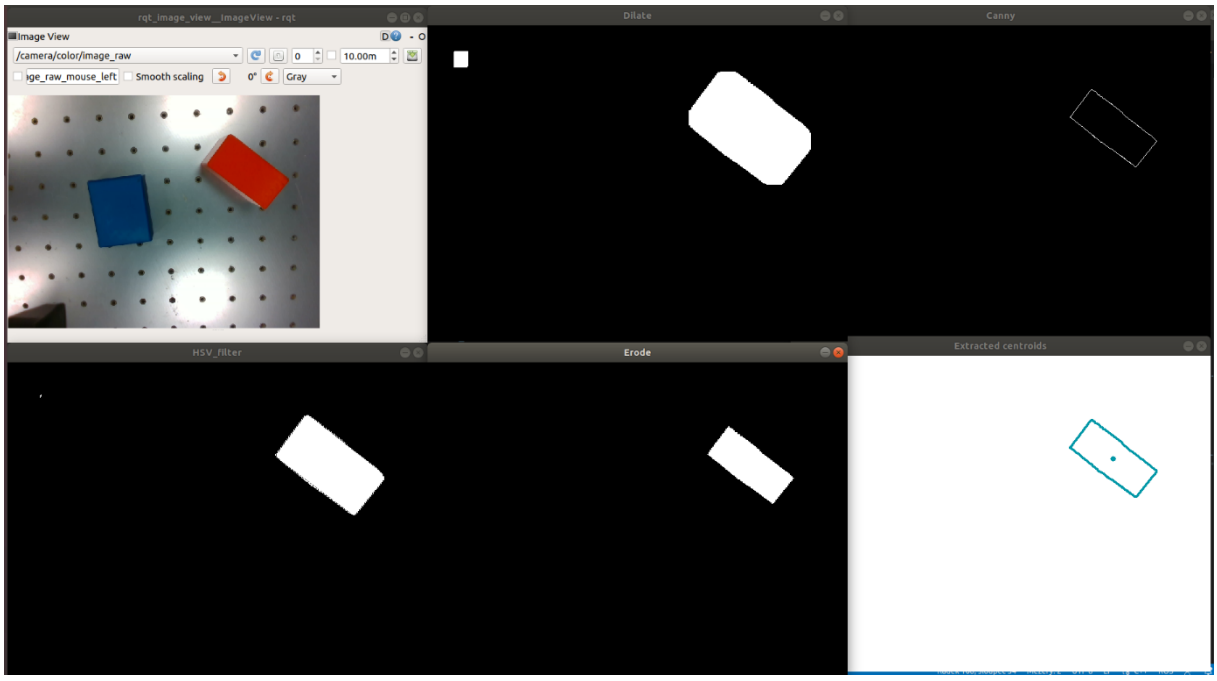
Barevný obraz je z velké části zpracováván pomocí knihovny OpenCV. Objekt známé barvy je v obraze izolován pomocí HSV (*Hue, Saturation, Value*) filtru, který podle nastavených konstant vytvoří černobílou masku objektu v obraze. HSV filtr byl zvolen, protože např. RGB (*Red, Green, Blue*) filtr nedisponuje dostatečně dobrým izolováním konkrétního

barevného spektra, viz Obr. 27. Černobílá maska je následně ještě vyfiltrována od případného šumu.



Obr. 27: Porovnání izolování barevného objektu v obraze pomocí RGB a HSV filtru (ve stejném pořadí zleva)

Podle barevných přechodů černobílé masky se následně vytvoří kontury objektu, pro které se pomocí obrazových momentů vypočítají souřadnice jednotlivých středů. Tyto středy kontur se už jen zprůměrují do jednoho hlavního středu. Úhel natočení objektu je získán pomocí vepsání elipsy do kontury s největší plochou. Jednotlivé kroky zpracování barevného obrazu jsou zachyceny na Obr. 28.



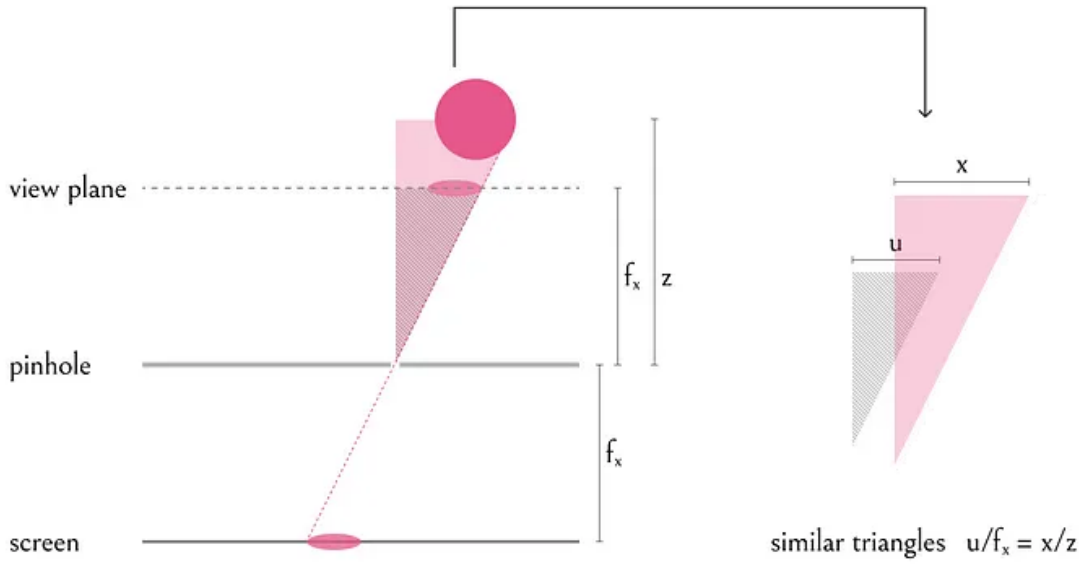
Obr. 28: Vizualizace jednotlivých kroků zpracování barevného obrazu

Výpočet 3D souřadnice z hloubkové mapy

Získání Z souřadnice (hloubky) konkrétního pixelu z černobílé hloubkové mapy, se realizuje převedením jeho odstínu černobílé barvy na číslo. Výpočet X a Y souřadnice vychází ze znalosti Z souřadnice, ohniskové vzdálenosti f_x, f_y, skutečného středu kamery C_x, C_y a podobností 2 trojúhelníků. Ohniskové vzdálenosti a skutečné středy kamery se nachází v transformačních datech kamery (v pixelovém formátu). Na Obr. 29 je vidět podobnost projekčních trojúhelníků předmětu a obrazce v *Camera obscura*, pak pro případ hloubkové kamery lze psát:

$$X = Z \cdot \frac{(X_p - C_{x,p})}{f_{x,p}} \tag{3.1}$$

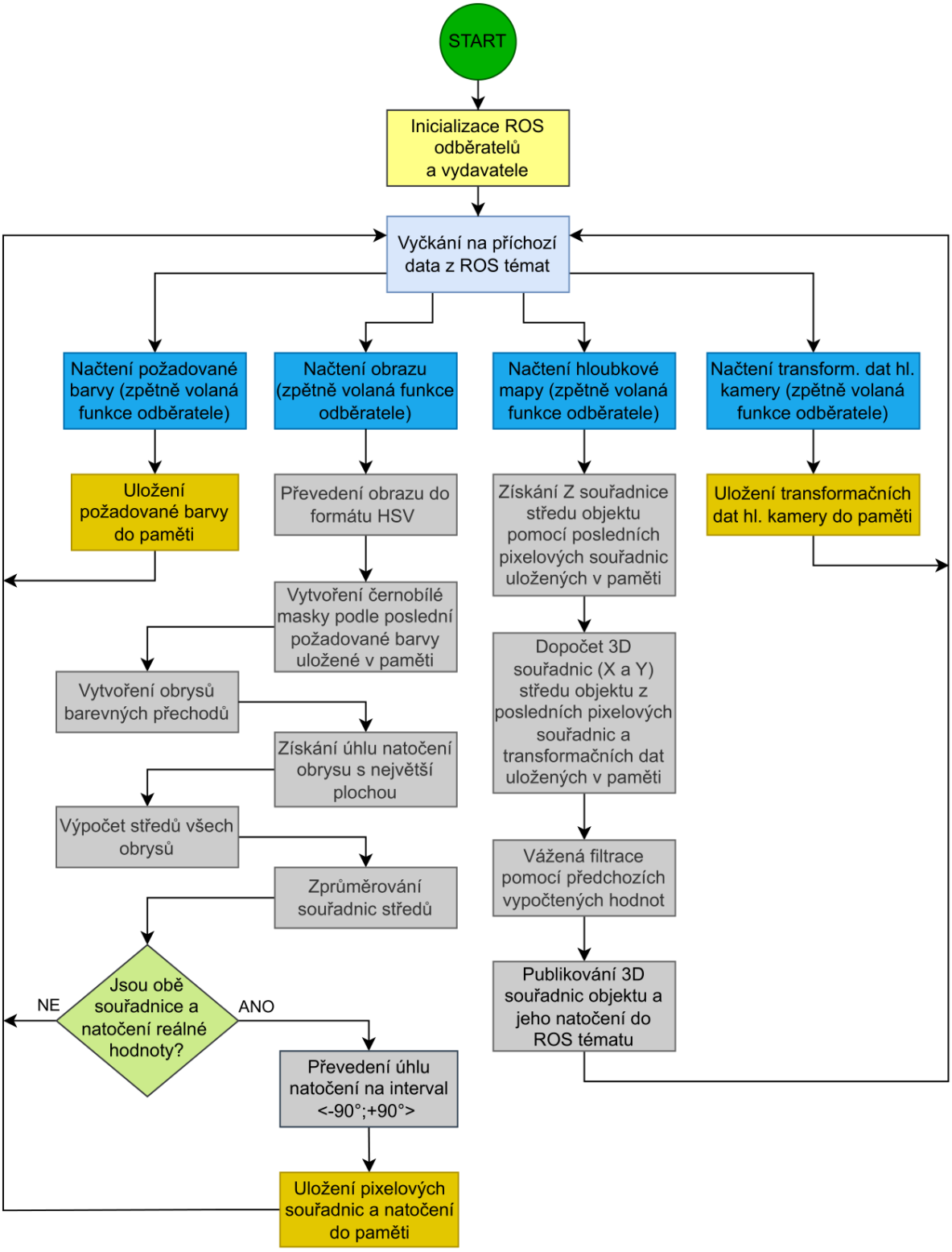
kde X_p je pixelová souřadnice středu objektu získaná ze zpracování barevného obrazu. Y souřadnice se vypočítá stejným způsobem s odpovídajícími hodnotami.



Obr. 29: Ukázka podobnosti trojúhelníků předmětu a obrazce v Camera obscura [37]

Program na zpracování obrazu

Celý program uzlu na zpracování obrazu je znázorněn vývojovým diagramem na Obr. 30.



Obr. 30: Vývojový diagram – zpracování obrazu

3.2.3 Programy pro robota a gripper

Ovládání robota UR5e a Robotiq gripperu je realizováno pomocí 2 uzlů, kde jeden je tzv. hlavní kontrolér, který na základě dat z hloubkové kamery ovládá robotické rameno a gripper. Druhý uzel, tzv. gripper_kontrolér, už jen překládá zjednodušený požadavek na pohyb gripperu od hlavního kontroléru do binární komunikační struktury gripperu, viz Obr. 26.

Laboratorní úloha má 2 opakující se části, které se odlišují jen o poziční hodnoty objektu a výšku položení objektu. Z toho důvodu velká část programu hlavního kontroléru je funkce „find_and_grab“, do které vstupuje 3D pozice hledaného objektu a výška jeho následného umístění (tj. výška předchozího objektu). Tato funkce zároveň také vrací výšku aktuálně manipulovaného objektu, takže volání funkce „find_and_grab“ jde za sebou tzv. řetězit.

Ovládání robotického ramene UR5e pomocí MoveIt

Robotické rameno je ovládáno pomocí rozšiřujícího balíčku MoveIt, konkrétně pomocí jeho Python rozšíření *moveit_commander*. Na začátku programu je potřeba zinicializovat objekt *MoveGroupCommander* s příslušným jménem plánovací skupiny. Jméno slouží jako unikátní identifikátor pro načtení konfiguračních souborů do *move_group* uzlu.

Následně se už může ovládat pohyb robotického ramene třemi různými způsoby:

- pomocí kloubových souřadnic – umožňuje zadávat požadovanou koncovou polohu robota definováním natočení každého jeho kloubu.
- pomocí kartézských souřadnic a natočení v prostoru – navede koncový člen robotického ramene na zadanou kartézskou souřadnici s prostorovým natočením definovaným pomocí kvaternionů.
- pomocí skupiny bodů v kartézských souřadnicích a natočení v nich – naplánuje a vykoná trajektorii pro koncový člen ramene, která bude procházet zadanými body v prostoru. Zároveň trajektorie bude respektovat prostorové natočení v každém bodě zadané skupiny.

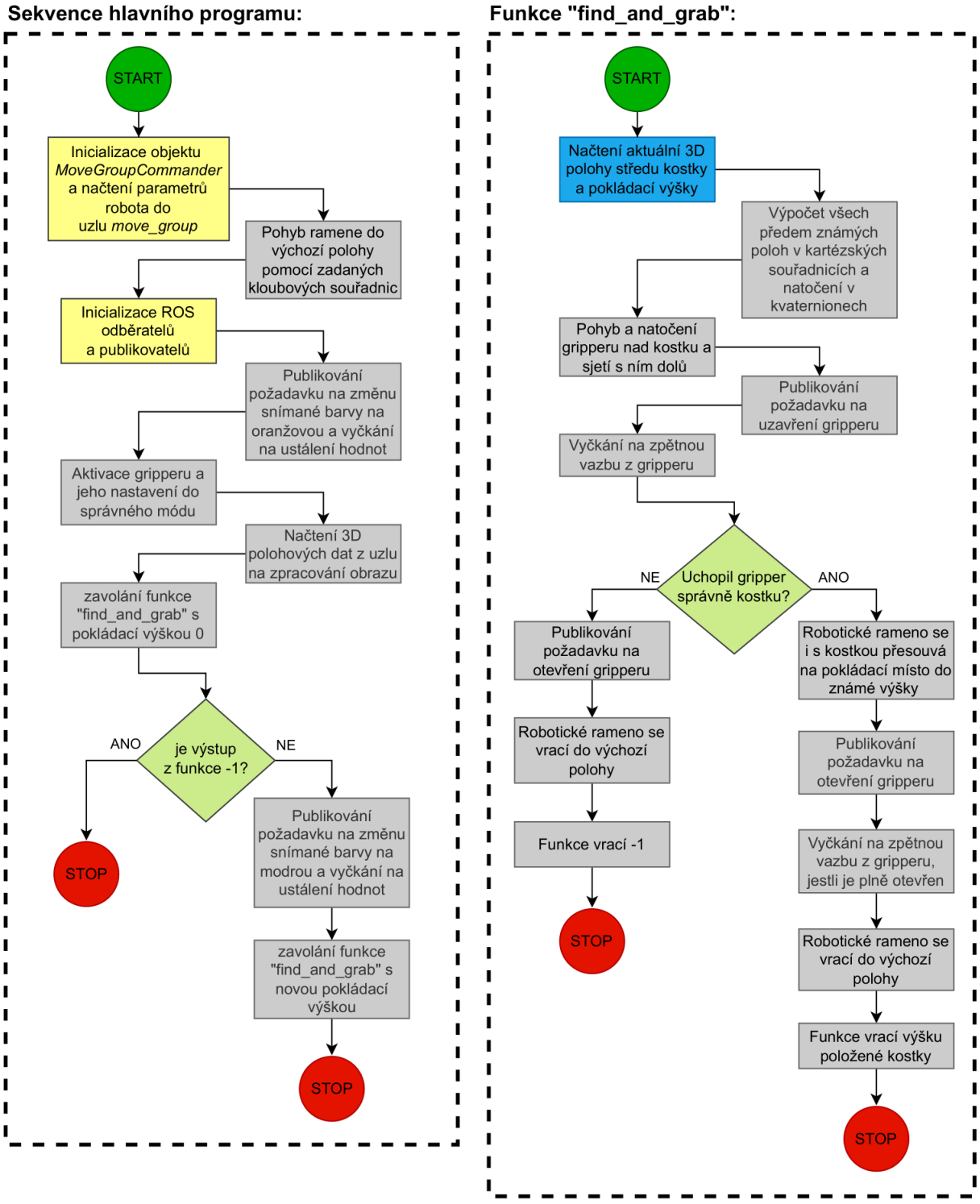
Prvotní pohyb ramene do výchozí polohy při inicializaci hlavního kontroléru je realizován na základě zadaných kloubových souřadnic. Pro následné další pohyby je už používáno zadávání pozice pomocí předem vypočtených pozic v kartézském souřadném systému. Zadávání skupiny bodů v kartézských souřadnicích pro naplánování trajektorie se ukázalo, jako příliš složité na implementaci do laboratorní úlohy.

Moveit_commander dále umožňuje získat aktuální natočení všech kloubů, nebo pozici koncového členu v kartézských souřadnicích a natočení opět v kvaternionech. Při plánování trajektorie je možnost jí vizualizovat v RViz před jejím samotným vykonáním.

Trajektorie jsou plánovány pomocí MoveIt výchozí OMPL plánovací knihovny a konkrétně podle jejího RRT (*Rapidly-exploring Random Tree*) plánovače.

Program hlavního kontroléru:

Program uzlu hlavního kontroléru a jeho funkce „find_and_grab“ jsou znázorněny vývojovými diagramy na Obr. 31.



Obr. 31: Vývojový diagram – hlavní kontrolér

3.2.4 Testování a výsledky

Funkčnost uchopovacího algoritmu byla testována opakovaným prováděním laboratorní úlohy (nalezení a uchopení barevných kostek v prostoru a jejich následné sestavení na sebe). Celkem bylo provedeno 50 opakování laboratorní úlohy. Tabulka 1 zachycuje úspěšnost jejího vykonání.

Tabulka 1: úspěšnost uchopovacího algoritmu

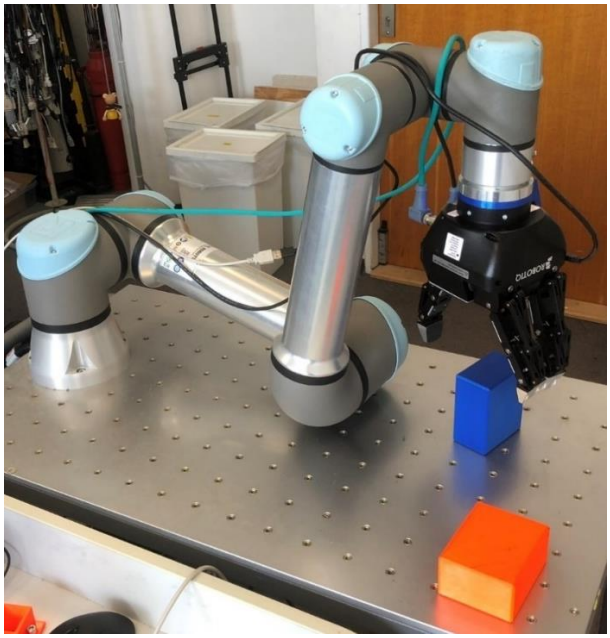
Celkem pokusů	Úspěšné pokusy	Chybné pokusy	Chyba ramene	Chyba gripperu	Chyba kamery
50	37	13	10	2	1

Proběhlo 37 úspěšných pokusů provedení laboratorní úlohy, což odpovídá 74 % úspěšnosti uchopovacího algoritmu.

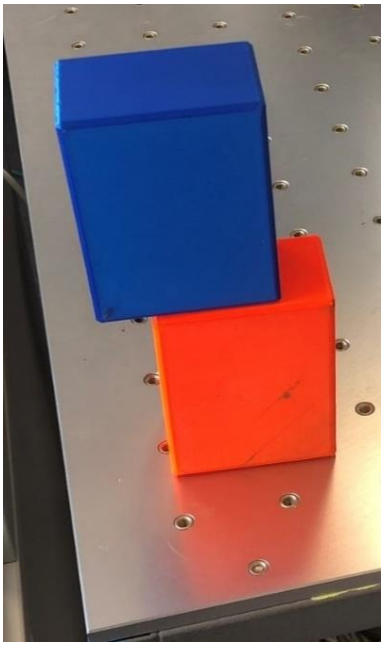
Do chyb robotického ramene byly zahrnuty chyby plánovače trajektorií a chyba komunikace robotického ramene s ROS (v hostitelském systému nebyl implementován *real-time kernel*, viz kapitola 3.1.1). Chyby plánovače byly způsobeny jeho volbou. Při testování úlohy byl využíván plánovač RRT, který nalezne více řešení a nejoptimálnější vykoná. Jak je vidět na Obr. 32, nalezená řešení nebyla vždy optimální.

Chyby gripperu byly jen jednoho druhu. Gripper jen naznačil uchopení, ale hned se rozevřel. Nejspíše způsobeno špatnou zpětnou vazbou.

Chyba kamery nastala v podobě úplně špatného zaměření kostky (gripper kostku vůbec neměl šanci uchopit). Chybovost kamery měla ve statistickém souboru nejmenší hodnotu, ale zvolená metoda zpracování obrazu nemá dostatečnou přesnost na lícování sestavovaných kostek na sebe, viz Obr. 33. Efektivnější by bylo zpracovávat data z hloubkové kamery pomocí knihovny PCL (*Point Cloud Library*).



Obr. 32: Příklad chyby RRT plánovače



Obr. 33: ukázka sestavení kostek

ZÁVĚR

Cílem bakalářské práce bylo seznámení se s problematikou zahrnující Robotický operační systém, jeho rozšiřujícími balíčky MoveIt a Gazebo a projektem ROS Industrial. Výsledkem práce bylo vytvoření uchopovacího algoritmu, který podle dat získaných z hloubkové kamery uchopí předmět v prostoru. Tento algoritmus řídil robotické rameno UR5e, (od firmy Universal Robots), Robotiq 3-Finger gripper a hloubkovou kameru Intel RealSense D435. Uchopovací algoritmus byl také testován na laboratorní úloze – nalezení a uchopení barevných kostek v prostoru a jejich následné sestavení na sebe.

První část práce se věnuje základním principům fungování Robotického operačního systému a jeho různých využití. V následující části rešerše je čtenáři představen komunitní projekt ROS Industrial a všechny možnosti ROS, které může vývojářům poskytovat. Dále jsou přiblíženy vnitřní funkce pohybového balíčku MoveIt a krátce představen simulační balíček Gazebo.

Druhá část práce se dělí na dvě podkapitoly. První podkapitola popisuje technické a softwarové specifikace jednotlivých HW zařízení a jejich následná implementace do ROS. Druhá podkapitola popisuje realizaci laboratorní úlohy. Obraz byl zpracováván pomocí knihovny OpenCV a robotické rameno UR5e řízeno pomocí balíčku MoveIt. Zvolené řešení laboratorní úlohy má 74 % úspěšnosti vykonání na statistickém souboru o 50 pokusech. Pro zvýšení úspěšnosti realizace zadané úlohy, by bylo potřeba implementovat do hostitelského systému balíček *real-time kernel* a zvolit jiný plánovač trajektorie. Dále přesnější uchopování kostek by se dalo realizovat zpracováním dat z hloubkové kamery pomocí knihovny PCL (*Point Cloud Library*).

SEZNAM POUŽITÝCH ZDROJŮ

[1] Introduction. In: *ROS Wiki: Documentation* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-18]. Dostupné z: http://wiki.ros.org/ROS/Introduction/

[2] Robot Operating System. In: *Wikipedia: the free encyclopedia* [online]. San Francisco: Wikimedia Foundation, 2001-2023 [cit. 2023-05-24]. Dostupné z: https://en.wikipedia.org/wiki/Robot_Operating_System

[3] Distributions. In: *ROS Wiki: Documentation* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-23]. Dostupné z: http://wiki.ros.org/Distributions

[4] ROS programming software. In: *Direct INDUSTRY* [online]. Marseille: VirtualExpo group, 2023 [cit. 2023-05-23]. Dostupné z: https://www.directindustry.fr/prod/automationware/product-192516-2394360.html

[5] Concepts. In: *ROS Wiki: Documentation* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-23]. Dostupné z: http://wiki.ros.org/ROS/Concepts

[6] Nodes. In: *ROS Wiki: Documentation* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-23]. Dostupné z: http://wiki.ros.org/Nodes

[7] Topics. In: *ROS Wiki: Documentation* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-23]. Dostupné z: http://wiki.ros.org/Topics

[8] Parameter Server. In: *ROS Wiki: Documentation* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-23]. Dostupné z: http://wiki.ros.org/Parameter%20Server

[9] ROS Robotics By Example: Rviz. In: *Packt* [online]. Birmingham: Packt Publishing, 2023 [cit. 2023-05-25]. Dostupné z: https://subscription.packtpub.com/book/iot-&-hardware/9781782175193/2/ch02lvl1sec17/rviz

[10] TELLEZ, Ricardo. How to Start with Self-Driving Cars Using ROS. In: *The Construct* [online]. Barcelona: The Construct Sim, 2023 [cit. 2023-05-23]. Dostupné z: https://www.theconstructsim.com/start-self-driving-cars-using-ros/

[11] Automated Driving with ROS at BMW. In: *ROSCon: ROSCon 2015 - Presentations* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-25]. Dostupné z: https://roscon.ros.org/2015/presentations/ROSCon-Automated-Driving.pdf

[12] *Relay Robotics* [online]. San Jose: Relay Robotics, 2023 [cit. 2023-05-23]. Dostupné z: https://www.relayrobotics.com/

[13] Relay. In: *Robots your guide to the world of robotics* [online]. Manhattan: IEEE (Institute of Electrical and Electronics Engineers), 2023 [cit. 2023-05-23]. Dostupné z: https://robotsguide.com/robots/relay/

[14] FLIGHT SOFTWARE WORKSHOP. *FSW 2022: The Wrong Way to Integrate cFS and ROS* [online]. In: . Youtube video [cit. 2023-05-23]. Dostupné z: https://youtu.be/Gvjxwtf6WMo

[15] DUNBAR, Brian, Rick CHEN, ed. VIPER: VIPER's Mission Operations. In: *National Aeronautics and Space Administration* [online]. Washington, D.C.: National Aeronautics and Space Administration, 2023 [cit. 2023-05-23]. Dostupné z: https://www.nasa.gov/viper/lunar-operations

[16] DUNBAR, Brian, Tricia TALBERT, ed. NASA Replans CLPS Delivery of VIPER to 2024 to Reduce Risk. In: *National Aeronautics and Space Administration* [online]. Washington, D.C.: National Aeronautics and Space Administration, 2023 [cit. 2023-05-25]. Dostupné z: https://www.nasa.gov/feature/nasa-replans-clps-delivery-of-viper-to-2024-to-reduce-risk

[17] Description. In: *ROS-Industrial: About* [online]. San Antonio: Southwest Research Institute, 2023 [cit. 2023-04-30]. Dostupné z: https://rosindustrial.org/about/description/

[18] Our Brief History. In: *ROS-Industrial: About* [online]. San Antonio: Southwest Research Institute, 2023 [cit. 2023-05-24]. Dostupné z: https://rosindustrial.org/briefhistory

[19] Industrial. In: *ROS Wiki* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-24]. Dostupné z: http://wiki.ros.org/Industrial

[20] Project Governance. In: *ROS 2 Documentation: Rolling* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-24]. Dostupné z: https://docs.ros.org/en/rolling/The-ROS2-Project/Governance.html

[21] Concepts. In: *MoveIt: Docs* [online]. Boulder: PickNik Inc, 2023 [cit. 2023-05-23]. Dostupné z: https://moveit.ros.org/documentation/concepts/

[22] CHOMP Planner. In: *MoveIt: Tutorials* [online]. Boulder: PickNik Inc, 2023 [cit. 2023-05-25]. Dostupné z: http://docs.ros.org/en/melodic/api/moveit_tutorials/html/doc/chomp_planner/chomp_planner_tutorial.html

[23] STOMP Planner. In: *MoveIt: Tutorials* [online]. Boulder: PickNik Inc, 2023 [cit. 2023-05-25]. Dostupné z: http://docs.ros.org/en/melodic/api/moveit_tutorials/html/doc/stomp_planner/stomp_planner_tutorial.html

[24] Concepts: Trajectory Processing. In: *MoveIt: Docs* [online]. Boulder: PickNik Inc, 2023 [cit. 2023-05-25]. Dostupné z: https://moveit.picknik.ai/humble/doc/concepts/trajectory_processing.html

[25] Gazebo: Get Started. In: *Gazebo* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-25]. Dostupné z: https://classic.gazebosim.org/#getstarted

[26] Universal_Robots_ROS_Driver. In: *GitHub: Universal Robots* [online]. Odense: Universal Robots, 2023 [cit. 2023-05-17]. Dostupné z: https://github.com/UniversalRobots/Universal_Robots_ROS_Driver/

[27] GREENFIELD, David. Universal Robots Unveils New 20kg Payload Cobot. In: *Automation World automationworld.com* [online]. Nashville: Endeavor Business Media, 2023 [cit. 2023-05-17]. Dostupné z: https://www.automationworld.com/factory/robotics/article/22288597/universal-robots-unveils-new-20kg-payload-cobot/

[28] UR5e technical details. In: *Universal Robots* [online]. Odense: Universal Robots, 2023 [cit. 2023-05-17]. Dostupné z: https://www.universal-robots.com/media/1802778/ur5e-32528_ur_technical_details_.pdf/

[29] Universal_robot. In: *GitHub: ros-industrial* [online]. San Antonio: Southwest Research Institute, 2023 [cit. 2023-05-17]. Dostupné z: https://github.com/ros-industrial/universal_robot/

[30] 3-Finger Adaptive Robot Gripper. In: *Robotiq* [online]. Québec: Robotiq inc., ©2008-2021 [cit. 2023-05-16]. Dostupné z: https://robotiq.com/products/3-finger-adaptive-robot-gripper/

[31] 3-Finger Adaptive Robot Gripper Instruction Manual. In: *Robotiq* [online]. Québec: Robotiq inc., ©2008-2021 [cit. 2023-05-16]. Dostupné z: https://assets.robotiq.com/website-assets/support_documents/document/3-Finger_PDF_20210617.pdf?_ga=2.144086333.989610477.1684185339-624752183.1667314603/

[32] Robotiq. In: *ROS Wiki* [online]. Mountain View: Open Source Robotics Foundation, 2023 [cit. 2023-05-16]. Dostupné z: http://wiki.ros.org/robotiq/

[33] Intel® RealSense™ Depth Camera D435. In: *Intel® RealSense™* [online]. Santa Clara: ©Intel Corporation, 2023 [cit. 2023-05-19]. Dostupné z: https://www.intelrealsense.com/depth-camera-d435/

[34] Intel Hloubková kamera 0.2m, model: D435 1280 x 720. In: *RS Components* [online]. Varšava: © RS Components Sp. z o.o. [cit. 2023-05-19]. Dostupné z: https://cz.rs-online.com/web/p/kamery-se-snimanim-hloubky/1720981/

[35] Intel® RealSense™ Camera 400 Series Product Family Datasheet. In: *Intel® RealSense™* [online]. Santa Clara: ©Intel Corporation, 2023 [cit. 2023-05-19]. Dostupné z: https://www.intelrealsense.com/download/20289/?tmstv=1680149335/

[36] Realsense-ros. In: *GitHub: IntelRealSense* [online]. Santa Clara: ©Intel Corporation, 2023 [cit. 2023-05-19]. Dostupné z: https://github.com/IntelRealSense/librealsense/

[37] From depth map to point cloud. In: *Medium* [online]. Londyn: A Medium Corporation, 2023 [cit. 2023-05-22]. Dostupné z: https://medium.com/yodayoda/from-depth-map-to-point-cloud-7473721d3f

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ROS	Robotický operační systém
HW	Hardware
SW	Software
HSV	Hue, Saturation, Value
RGB	Red, Green, Blue
OMPL	Open Motion Planning Library
RRT	Rapidly-exploring Random Tree
PCL	Point Cloud Library

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH