UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

SW PRO OVLÁDÁNÍ ROBOTICKÉHO RAMENE Z PROSTŘEDÍ MATLAB

Dominik Varga

Bakalářská práce
2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2017/2018**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Dominik Varga**
Osobní číslo:	**I15025**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Řízení procesů**
Název tématu:	**SW pro ovládání robotického ramene z prostředí MATLAB**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Cílem práce je vytvořit SW v pro ovládání robotického ramene Universal Robots UR3 z prostředí MATLAB. SW umožní komunikovat s programem Polyscope běžícím na HW robota a umožní tak zasílání příkazů pro ovládání robota.
Teoretická část bude věnována problematice robotických ramen a popisu pohybu robota v prostoru.
V praktické části bude uveden přehled podobných řešení, popsáno rameno UR3 a program Polyscope. Bude uvedena realizace samotné komunikace mezi softwarem robota a prostředím MATLAB a ukázána jednoduchá úloha pro demonstraci funkčnosti navrženého řešení.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

BENEŠ, Pavel. Automatizace a automatizační technika: prostředky automatizační techniky. 5., rozš. a aktualiz. vyd. Brno: Computer Press, 2014. ISBN 978-80-251-3747-5.
KNOFLÍČEK, Radek. Mobilní roboty pro průmyslové využití - VUTBOT 2. Brno: CERM, 2005. ISBN 80-7204-387-0.
NOVÁK, Petr. Mobilní roboty: pohony, senzory, řízení. Praha: BEN - technická literatura, 2005. ISBN 80-7300-141-1.
DUŠEK, F., HONC, D. Matlab a Simulink, Úvod do používání. skriptum, Univerzita Pardubice, vydání první, Pardubice, 2005.

Vedoucí bakalářské práce: **Ing. Daniel Honc, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **6. prosince 2017**
Termín odevzdání bakalářské práce: **11. května 2018**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 12. prosince 2017

Prohlášení autora

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 11. 05. 2018

Dominik Varga

Poděkování

Tímto bych rád poděloval vedoucímu bakalářské práce Ing. Danielu Honcovi, Ph.D. za vedení práce, ochotu, rady a připomínky ke zpracování. Dále bych rád poděkoval rodině a přítelkyni za podporu po celou dobu studia.

V Pardubicích dne 11. 05. 2018

Dominik Varga

ANOTACE

Cílem této bakalářské práce je vytvořit software v prostředí MATLAB, který umožní komunikovat s programem robotického ramene UR3 a umožní jeho ovládání. V teoretické části práce se snažím obeznámit čtenáře s pojmy robot, průmyslový robot a kooperující robot. Dále uvádím jednoduchou kategorizaci průmyslových robotů a popisuji, jak se pohybují v prostoru. Praktická část je věnována popisu známějších kooperujících ramen a podrobnému popisu ramene UR3 včetně jeho řídícího softwaru PolyScope. Dále je ukázáno řešení samotné komunikace mezi softwarem robota a MATLAB. K závěru práce čtenář nalezne ukázku jednoduché úlohy prezentující funkčnost návrhu.

KLÍČOVÁ SLOVA

Matlab, UR3, PolyScope, ovládání, Universal Robots, pohyb robota.

TITLE

SW FOR ROBOTIC ARM MANIPULATION FROM MATLAB ENVIRONMENT

ANNOTATION

The objective of this work is creating a software in the MATLAB environment, which allows communication witch the UR3 robotic arm and allows to control it. In the theoretical part, I try to familiarize the reader with the term of robot, industrial robot and cooperative robot. Next, I give a simple categorization of industrial robots and describe how the robots move in space. The practical part is dedicated to the cooperating arms and a detailed description of the UR3 arm including its PolyScope control software. Furthermore, is describe a solution of the communication between the robot software and the MATLAB. The conclusion of the thesis is a simple task presenting the functionality of the communication solution.

KEYWORDS

Matlab, UR3, PolyScope, Control, Universal Robots, Robot movement.

OBSAH

SEZNAM ZKRATEK A ZNAČEK

DoF	Degrees of Freedom
PUMA	Programmable Universal Manipulation Arm
TCP	Tool Center Point
TCP/IP	Transmission Control Protocol/Internet Protocol

SEZNAM SYMBOLŮ PROMĚNNÝCH VELIČIN A FUNKCÍ

$\boldsymbol{Rx}$	matice rotace v ose x
$\boldsymbol{Ry}$	matice rotace v ose y
$\boldsymbol{Rz}$	matice rotace v ose z
$\boldsymbol{rx}$	vektor rotace v ose x, rad
$\boldsymbol{ry}$	vektor rotace v ose y, rad
$\boldsymbol{rz}$	vektor rotace v ose z, rad
$\boldsymbol{x}$	vektor polohy robota v ose x, mm
$\boldsymbol{y}$	vektor polohy robota v ose y, mm
$\boldsymbol{z}$	vektor polohy robota v ose z, mm
ψ	úhel natočení TCP robota v ose x, °
θ	úhel natočení TCP robota v ose y, °
ϕ	úhel natočení TCP robota v ose z, °
x	souřadnice osy x, mm
y	souřadnice osy y, mm

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

ÚVOD

Dnešní doba je rychlá a náročná, již od pradávna platí tvrzení, že čas jsou peníze. Zejména v průmyslu toto tvrzení platí dvojnásob. I to je důvod, proč se většina odvětví v průmyslu 21. století snaží o kompletní, nebo alespoň i částečnou automatizaci svého výrobního procesu. Automatizace výrobního procesu by měla za následek nesčetně výhod, jako třeba zrychlení výroby, bezpečnost, přesnější a kvalitnější výrobky a spoustu dalšího.

Věda, která se tímto problémem zabývá, se jmenuje automatizace. Mezi hlavní prostředky automatického řízení patří mimo jiné regulátory, senzory, mikropočítače atd., ale významnou roli zde hrají roboti a manipulátory. Plně automatizovaný podnik s použitím robotů nebo průmyslových manipulátorů dokáže lépe konkurovat podnikům, které využívají lidské zdroje. Redukcí lidského faktoru a automatickým nastavením průmyslových robotů se snižují nejen nároky na lidské zdroje, ale zároveň se minimalizuje i chybovost, protože programovatelný robot se drží zadání a dokáže se celý den věnovat i těm nejnudnějším činnostem. Z toho vyplývá vyšší produktivita práce i lepší kvalita finálního výrobku. V průmyslu se nejčastěji používají robotická ramena, která jsou konstruována tak, aby svým chováním mohla plně nahradit lidskou paži. Díky takovéto robotické paži lze nahradit člověka všude tam, kde se jedná o monotónní práci, nebo práci ve škodlivém nebezpečném prostředí.

V této práci se zabývám ovládáním robotického ramene Universal Robots UR3, pomocí softwarové komunikace mezi programem PolyScope a MATLAB. V teoretické části uvádím zásadní rozdíl mezi pojmem robot, průmyslový robot a popisuji co je „cobot“ a k čemu jej lze využít. Dále uvádím jednoduchou klasifikaci robotických ramen a seznamuji čtenáře se způsoby a typy jejich pohybů v prostoru. V praktické části uvádím několik známějších společností a jejich kooperujících zástupců. Dále je zde uvedené a podrobně popsané robotické rameno UR3 společnosti Universal Robots a jsou zde zmíněny a popsány nejdůležitější funkce jeho řídícího softwaru PolyScope. V závěru práce je ukázána jednoduchá úloha robota, která prezentuje funkčnost softwarové komunikace.

1 TEORETICKÁ ČÁST

Teoretická část této práce s příslušnými oddíly a pododdíly je věnována problematice robotických ramen. V této kapitole se hlavně zabývám rozdílem mezi klasickým průmyslovým robotem a kooperujícím robotem. Dále uvádím některé známější metody kategorizace robotů a snažím se popsat základní druhy pohybu robota v prostoru a styl jejich pohybu.

1.1 ROBOT, PRŮMYSLOVÝ ROBOT A COBOT

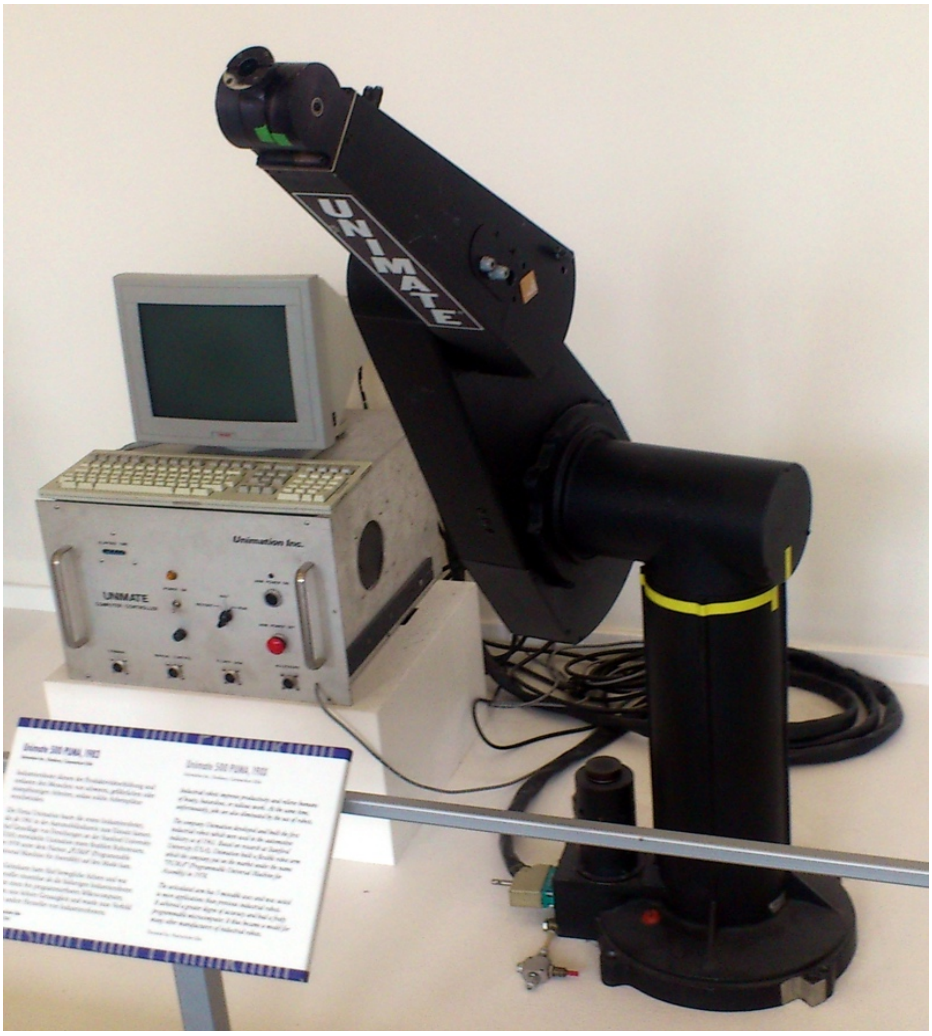
1.1.1 Robot a průmyslový robot

Slovo „robot“ bylo poprvé použito v roce 1920 v divadelní hře R.U.R spisovatele Karla Čapka a bylo odvozeno od slova „robota“ neboli nucená práce. Přímá definice robota jako taková asi neexistuje a pod tímto pojmem si lze představit opravdu mnoho, od malých robotických vozítek, kuchyňských spotřebičů, průmyslových robotů až po složité humanoidní roboty. Hlavní myšlenkou robota vždy bylo a je multifunkční automatizované zařízení, které pracuje s elektrickou nebo jinou formou energie a provádí nejrůznější úkony namísto člověka. Hlavním úkolem robotů je tedy nějakým způsobem odlehčit člověku a pracovat místo něj. Požadavkem na každého robota je, aby prováděné úkony byly přesné, opakovatelné a v případě potřeby jej bylo možné změnit na činnost jinou. Každý složitější robot je řízen nějakým programem a v případě požadavku na změnu úlohy je zapotřebí, aby jej bylo možné přeprogramovat. V dnešní době je robot jedním z nejdůležitějších nástrojů oboru automatizace a využívá se zejména při automatizaci celého výrobního procesů, ale využití robotů je už dneska prakticky neomezené.

V technické praxi 21. století se ale místo obecného pojmu robot využívá spíše výraz průmyslový robot, nebo manipulátor. Tímto způsobem dostaneme přesnější definici robota, která jej definuje jako přeprogramovatelný funkční manipulátor, který je určený k přenosu materiálů, dílů, nástrojů nebo speciálních zařízení pomocí předem naprogramovaných pohybů po předem stanovené dráze. Průmyslový robot je řízen programem a při správném výběru robota a jeho nástroje je průmyslový robot schopný provádět i složitější funkčně zaměřené úkony (Nof, 1985; Gupta, 2017).

Jeden z prvních patentovaných průmyslových robotů vznikl v roce 1954 a byl vytvořen Georgem Devolem. Tento robot dokázal přenášet věci z jednoho místa na místo druhé, a to o vzdálenosti 12 stop. O pár let později v roce 1975 vytvořil Victor Scheinman programovatelný univerzální flexibilní manipulátor zvaný PUMA, tento manipulátor byl schopný přemístit

objekt a umístit jej do libovolné polohy i orientace na požadované místo. Na obrázku 1.1 je znázorněný jeden z prvních robotických manipulátoru Puma 500 společnosti Unimate (Jeffus, 2012).



Obrázek 1.1 – Robotické rameno Puma Unimate 500 (Programmable Universal Machine for Assembly, 2001)

1.1.2 Části průmyslového robota

Hlavní části průmyslových robotů můžeme zařadit do čtyř hlavních subsystémů. Níže jsou uvedeny a stručně popsány jednotlivé subsystémy průmyslových robotů, mezi které patří zejména:

- mechanická jednotka,
- pohonný systém,
- kontrolní systém,

- nástroje.

Mechanická jednotka

Mechanická jednotka je spjatá zejména s ramenem robota a její základnou. Skládá se z konstrukčního rámu s podporou mechanické vazby, pohonů, ventilů, nejrůznějších bezpečnostních prvků a mnoho dalšího. Fyzické rozměry konstrukce a síla pohonu závisí na požadavcích a druhu aplikace (Gupta, 2017).

Pohonný systém

Pohonný systém je jedním z nejdůležitějších systémů průmyslových robotů sloužící k dodání energie pro mechanický pohyb robota. Pohony robotů mohou být elektrické, hydraulické nebo pneumatické.

- Elektrické motory jsou v dnešní době nejvíce používané, neboť tyto pohony mají mnohem menší nároky na údržbu a nižší vliv teploty na svou činnost.
- Hydraulické motory se používají všude tam, kde je potřeba manipulovat s velkými a těžkými předměty.
- Pneumatické motory jsou vhodné pro použití u jednodušších manipulátorů s menší nosností.

Řídicí systém

Řídicí jednotka (viz obrázek 1.2) je jednou z nejdůležitějších částí každého robota, jinak řečeno, slouží jako mozek robota. Mimo jiné zajišťuje komunikaci a zpracovává informace, které se iniciují. Dále má na starosti zastavení robota nebo koordinuje jeho pohybové sekvence. Téměř všechny robotické zařízení obsahují počítačové, nebo mikroprocesorové řídící jednotky, které provádí výpočetní funkce a mají na starost propojení senzorů, nástrojů a ostatních periferií. Nastavení řídící jednotky je možné měnit přeprogramováním (Gupta, 2017).

Obrázek 1.2 – Řídící jednotka robotického ramene UR3 (Universal Robots, 2018)

Nástroje

Každý průmyslový robot musí mít k dispozici alespoň jeden nástroj, aby mohl vykonávat nějakou činnost. Nástroj robota, někdy též známý jako koncový efektor robota, může být vestavěný a funkčně zaměřen, nebo v případě nutnosti ručně vyměnitelný za jiný druh nástroje. Příkladem robota s funkčně zaměřeným nástrojem znázorněným na obrázku 1.3 je robot určený k nějaké složitější aplikaci jako je například sváření, zatímco robot s vyměnitelným nástrojem většinou plní nějakou jednodušší mechanickou funkci, jako je např. přenos objektů apod.

Obrázek 1.3 – Svářecí robot společnosti ABB (IRB 1520ID, 2018)

Dalším významným nástrojem používaným při řízení robota jsou senzory. Senzory mají na starosti mnoho úkolů a jejich přítomnost je už dneska nezbytností. Mezi hlavní povinnosti senzorů patří podávání informací o stavu robota, také slouží jako nástroj ke komunikaci robota s okolním světem nebo k zajištění většího bezpečí spolupracovníků (Gupta, 2017).

1.1.3 Kooperující robot

Kooperující robot je speciální druh průmyslového robota, někdy též označovaný pojmem „cobot“. Slovo cobot vzniklo sloučením anglických slov collaborative a robot, což se dá volně přeložit jako robot spolupracovník. Koncept kooperujících robotů se zrodil přibližně kolem roku 1995 jako projekt General Motors Foundation, se záměrem vytvořit ještě

efektivnějšího průmyslového robota k přímé spolupráci s člověkem. Cobot se na první pohled od normálního průmyslového robota nijak zvlášť neliší, ale při nasazení do výrobního procesu sebou přináší zásadní rozdíly a výhody (Vojáček, 2017).

Při nasazení klasického průmyslového ramene do výrobního procesu bylo zapotřebí mnoho ochranných prvků, jako např. ochranná klec (viz obrázek 1.3), aby při pohybu robota nedošlo ke zranění člověka. Kooperativní robot žádné takové klece nepotřebuje, protože je speciálně navržený tak, aby mohl pracovat v těsné blízkosti člověka. Aby mohl robot pracovat s člověkem ruku v ruce a bez dalších bezpečnostních bariér, musí být vybaven nejrůznějšími bezpečnostními prvky, jako např. senzorem pro zastavení v případě kolize s člověkem a mnoho dalších senzorů. Cobot je obvykle vyroben z lehkého a flexibilního materiálu, což jej činí ještě obratnějším a rychlejším pomocníkem. Použití kooperujících robotů ve výrobním procesu tedy přináší mnoho výhod, jako zvýšení bezpečnosti lidských pracovníků, lepší využitelnost pracovního místa atd. Kooperujícího robota můžeme vlastně považovat za vylepšeného průmyslového robota (Vojáček, 2017).

Obrázek 1.4 – Ochranná klec pro průmyslové roboty ve výrobním procesu (Fixed perimeter guarding with weld curtains, 2008)

1.2 KATEGORIZACE PRŮMYSLOVÝCH ROBOTŮ

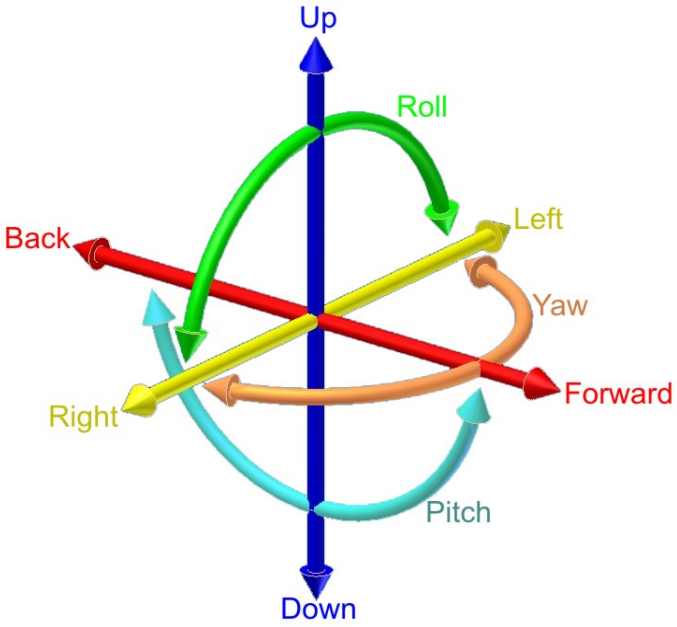
Průmyslový robot nebo „cobot“ je mocný nástroj, který dokáže ulehčit lidem práci, zrychlit nebo zdokonalit výrobu. Dneska už asi neexistuje odvětví průmyslu, které by nešlo nějakým způsobem automatizovat, ale jak již z předchozího oddílu 1.1 plyne, není robot jako robot. Při výběru robota je třeba dbát na jeho vlastnosti ať už pohybové, fyzické, nebo jiné. Z tohoto důvodu se roboti nějakým způsobem klasifikují a odlišují. Průmyslové roboty a manipulátory můžeme kategorizovat podle různých kritérií, a právě tato kritéria mají velký význam při výběru robota pro konkrétní činnost. Je mnoho způsobů kategorizace robotů, zde jsou uvedeny a stručně popsány některé ze známějších typů kategorizace, mezi které patří zejména:

- kategorizace podle počtu stupňů volnosti,
- kategorizace podle použité kinematické struktury,
- kategorizace podle geometrie pracovního prostoru.

Dalším významným způsobem kategorizace robotů je kategorizace podle použitých pohonů, o kterých byla zmínka v popisu pohonného systému v pododdíle o průmyslových robotech.

1.2.1 Kategorizace podle počtu stupňů volnosti

Počet stupňů volnosti udává takový počet nezávislých pohybů, které je robot schopen realizovat s ohledem na svoji základnu. Jinak řečeno je to minimální počet parametrů rotace a translace, který jednoznačně popisuje polohu bodu nebo tělesa v rovině či prostoru. Každý kloub robota zavádí jeden nový stupeň volnosti. Aby robot mohl dosahovat všech poloh a orientací v trojrozměrném prostoru, musí mít alespoň šest stupňů volnosti. V dnešní době je tento způsob asi nejčastěji používaným typem kategorizace průmyslových robotů, neboť umožnuje hned na první pohled určit, jaké budou pohybové schopnosti robota. Obrázek 1.5 znázorňuje všechny pohybové možnosti robota, který využívá šest stupňů volnosti (Skařupa, 2007).



Obrázek 1.5 – Šest stupňů volnosti pro pohyb v prostoru (LONESCU, 2010)

Jednoduchá kategorizace podle počtu stupňů volnosti je následující:

- Univerzální robot – vybavený 6 stupni volnosti jednoznačně vymezuje polohu i orientaci manipulujícího objektu v kartézském souřadném systému.
- Redundantní robot – jak již z názvu lze odvodit, je robot s více než 6 stupni volnosti. Tento robot se využívá především tam, kde je zapotřebí obcházení překážek, nebo k pohybu ve stísněném prostoru.
- Deficitní robot – je opakem robota redundantního a vyznačuje se nedostatkem stupňů volnosti. Tento robot má méně než 6 stupňů volnosti a je vhodný zejména u nenáročných aplikací z důvodu omezených pohybových možností.

1.2.2 Kategorizace podle kinematického řetězce

Kinematika studuje pohyb a trajektorii tělesa, a to bez příčiny tohoto pohybu. Při studiu kinematiky robota je nesmírně důležité, jakého typu kinematického řetězce robot využívá, protože to ovlivňuje jeho pohybové vlastnosti. Spojením několika segmentů, myšleno ramen, získáme kinematické dvojice, které jsou vzájemně pohyblivě spojeny nějakou vazbou. Spojením několika kinematických dvojic dohromady pak vznikne tzv. kinematický řetězec.

Kinematické řetězce jsou tedy tvořeny z jednotlivých členů a vazeb tzv. kinematických dvojic. Jinak řečeno jedná se o množinu ramen spojených klouby. Takovéto řetězce mohou být

otevřené nebo uzavřené, podle toho, jak jsou jeho členy připojeny k rámu. Kombinací obou typů řetězců vytvoříme řetězce smíšené neboli hybridní (Švejda, 2011).

Sériový robot

Sériový typ robota se vyznačuje otevřeným kinematickým řetězcem manipulátoru neboli tzv. open-loop chain. Tento typ robotů je v 90 % nejčastěji používaným typem a je charakterizován tím, že poslední člen má volný. Mezi obrovské výhody tohoto typu robotů patří jeho pracovní dosah a všeobecná univerzálnost. Základním nedostatkem sériového typu robota je menší přesnost než u robotů paralelních, k tomuto nedostatku dochází z důvodu sčítání chyb na jednotlivých kinematických dvojicích. Obrázek 1.6 znázorňuje sériové robotické rameno společnosti KUKA (Skařupa, 2007).

Obrázek 1.6 – Sériový manipulátor společnosti KUKA (Kuka Robotics, 2015)

Paralelní robot

Paralelní robot je představitelem uzavřeného kinematického řetězce manipulátoru tzv. closed-loop chain. Tento typ robotů je charakterizovaný tím, že vstupní i výstupní člen je upevněný k rámu. Mezi obrovské nevýhody tohoto typu robota patří složitost konstrukce, menší pracovní oblast a náročnější řídicí systém. Paralelní robot disponuje i velkými výhodami, jako je větší přesnost a tuhost než u robotů sériových. Příklad paralelního robota je znázorněn na obrázku 1.7 (Skařupa, 2007).

Obrázek 1.7 – Paralelní manipulátor společnosti ABB
(IRB 360 FlexPicker™, 2018)

Hybridní robot

Hybridní robot využívá kombinaci mechanismů jak otevřeného, tak uzavřeného kinematického řetězce. Jinak řečeno je hybridní robot vlastně sérioparalelním robotem. Tento typ robota kombinuje vlastnosti obou dvou typů, tím je docíleno obrovských výhod, jako např. překonání omezeného pracovního prostoru, což byl velký nedostatek paralelních robotů.

1.2.3 Kategorizace podle geometrie pracovní plochy

Popis geometrie má za cíl popis pohybu koncového efektoru v závislosti na čase, stanovení trajektorie, rychlosti a zrychlení. Nejčastěji používané typy struktur jsou uvedeny níže.

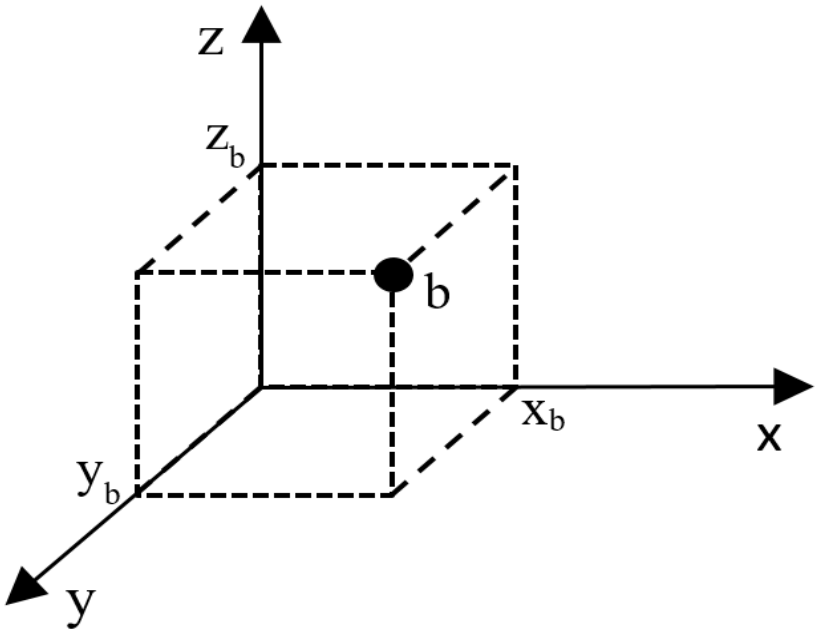
- Kartézská pohybová struktura, někdy též označovaná jako TTT struktura, se vyznačuje třemi posuvnými neboli translačními pohyby. Pracovní prostor této struktury má tvar hranolu a při pohybu nedochází ke změně orientace objektu. Kartézský robot, někdy též označovaný jako robot portálový, se nejčastěji využívá při podávání nebo přesouvání objektů.
- Cylindrická pohybová struktura, někdy též RTT struktura, se vyznačuje jedním rotačním pohybem a dvěma translačními pohyby. Pracovním prostorem této struktury je válec a při pohybu dochází ke změně orientace objektu.
- Sférická pohybová struktura má také označení jako RRT struktura a vyznačuje se dvěma rotačními a jedním posuvným pohybem. Tato struktura má pracovní prostor kulový a používá se především jako nástroj svařovacích linek.
- Angulární pohybová struktura, též RRR struktura, se vyznačuje třemi rotačními pohyby a v praxi se využívá nejčastěji. Tato struktura má pracovní prostor ve tvaru kulového vrchlíku a mimo jiné se vyznačuje dobrými dynamickými vlastnostmi.

1.3 POHYB ROBOTA V PROSTORU

Pohybové schopnosti každého robota nebo manipulátoru závisí na počtu použitých kinematických dvojic, o kterých byla zmínka v pododdílu 1.2.2 o kategorizaci robotů podle kinematické struktury. Počet nezávislých pohybů robota je dán počtem stupňů volnosti, o kterých byla rovněž zmínka v pododdílu 1.2.1 o kategorizaci podle stupňů volnosti. Základní pohyb robota v prostoru (viz obrázek 1.4) se skládá z pohybu posuvného, někdy též translačního, a pohybu rotačního. Při práci s robotickým ramenem bude uživatele s největší pravděpodobností zajímat zejména poloha nástroje, tedy poloha TCP. TCP je v tomto ohledu chápán jako středový bod nástroje a pohyb, ať už posuvný, nebo rotační, se tedy provádí právě s ohledem na koncový efektor robota.

1.3.1 Translační pohyb

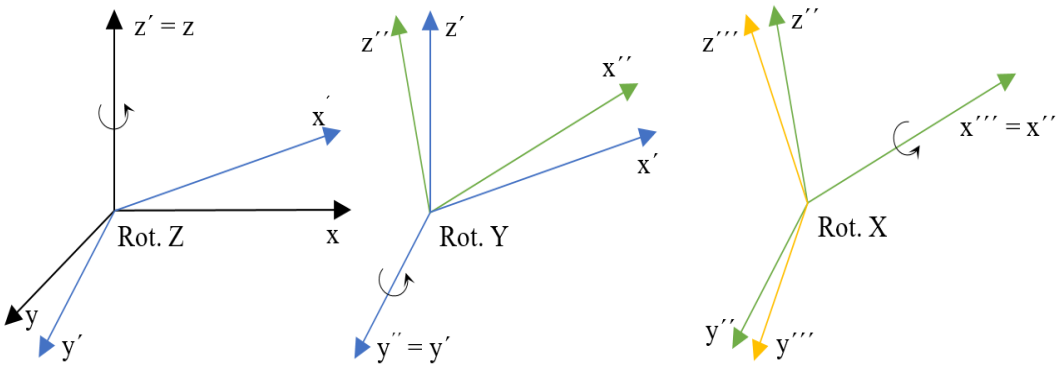
Aby robot mohl provádět nejrůznější úkony, potřebuje se efektivně přesouvat v prostoru. Jedním z hlavních a jednodušších částí pohybu v prostoru je pohyb translační. Při translačním pohybu se každý bod pohybuje po stejné trajektorii. Na obrázku 1.8 je vidět, že pro určení polohy objektu v trojrozměrném prostoru, se nejčastěji využívají pravoúhlé kartézské souřadnice (Švejda, 2011).



Obrázek 1.8 – Translační pohyb v kartézském souřadném systému

1.3.2 Rotační pohyb

Rotační pohyb je druhou a složitější částí ze základního pohybu robota. Při rotačním pohybu jakéhokoliv tělesa se všechny body pohybují po kružnici se středem na ose rotace. Je více způsobů, jak popisovat či přímo provádět rotaci robota, jednou z možností je použití tzv. transformačních matic, které slouží k natáčení os souřadných systémů. Obrázek1.9 znázorňuje, že k natočení dochází vždy mezi dvěma souřadnými systémy, tedy mezi původním souřadným systémem a budoucím nově vzniklým systémem.



Obrázek 1.9 – Postupná rotace souřadných systémů

Transformační matice někdy též zvané jako matice rotace jsou maticemi 3x3 a vyjadřují rotaci kolem některé osy souřadného systému o nějaký libovolný úhel. Tyto úhly jsou nazývány Eulerovými úhly a umožnují popis rotace objektu v prostoru. Někdy jsou Eulerovy úhly pro lepší představu rotace prezentovány jako úhly Roll, Pitch a Yaw, viz obrázek 1.10. Jednotlivé rotační matice znázorňují rovnice 1.1 až 1.3 (Švejda, 2011; Mostýn, 2012).

Rotace kolem osy x o úhel ψ

$$\boldsymbol{R_x}(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{bmatrix}. \tag{1.1}$$

Rotace kolem osy y o úhel ϕ

$$\boldsymbol{R_y}(\phi) = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix}. \tag{1.2}$$

Rotace kolem osy z o úhel θ

$$\boldsymbol{R_z}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.3}$$

V případě nulového pootočení souřadného systému získáme matici 1.4. Dále pak při pootočení o libovolný úhel dostaneme výslednou matici rotace, která vznikne postupným násobením jednotlivých rotačních matic 1.1 až 1.3. Výsledkem je postupná rotace kolem všech os souřadného systému, viz rovnice 1.5.

$$\boldsymbol{R} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{1.4}$$

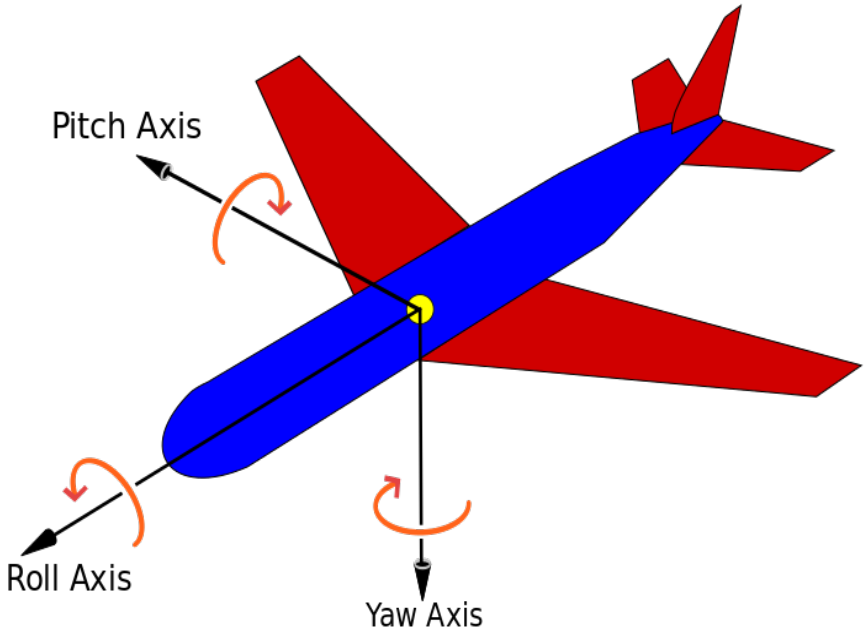
$$\boldsymbol{R} = \boldsymbol{R_x}(\psi) \cdot \boldsymbol{R_Y}(\phi) \cdot \boldsymbol{R_z}(\theta), \tag{1.5}$$

kde $\boldsymbol{R}$ – je nově vzniklá matice rotace,

$\boldsymbol{R_x}$ – je rotační matice kolem osy x o úhel ψ někdy označována jako matice pro roll,

$\boldsymbol{R_y}$ – je rotační matice kolem osy y o úhel ϕ někdy též označována jako matice pro pitch,

$\boldsymbol{R_z}$ – je rotační matice kolem osy z o úhel θ někdy označována jako matice pro yaw.



Obrázek 1.10 – Eulerovy úhly reprezentované jako roll, pitch, yaw (Yaw Axis Corrected, 2010)

1.3.3 Druhy pohybů robota

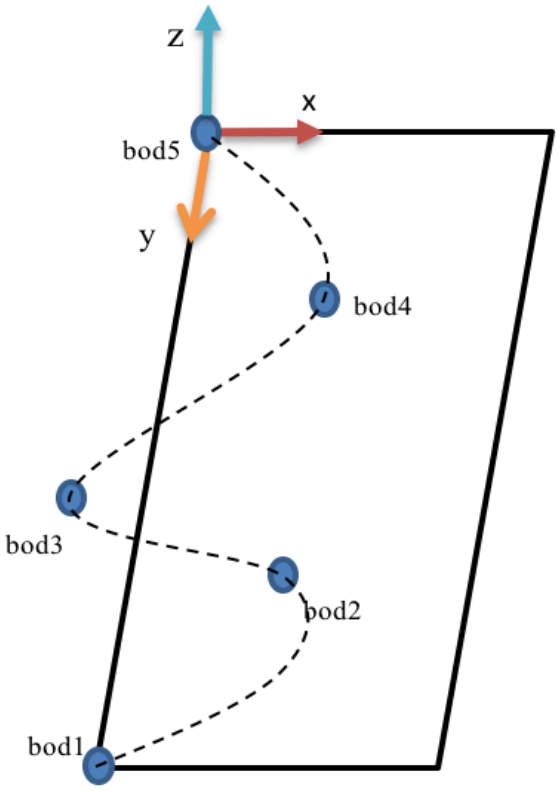
Při pohybu robota nezáleží jenom na tom, jestli se robot přesouvá pohybem posuvným, nebo rotuje kolem nějaké osy souřadného systému, ale také záleží na tom, jakým způsobem se k finální pozici dostane. Volbou správného typu pohybu se robot v prostoru pohybuje efektněji a mnohem lépe plní zadaný úkol. Robot při svém pohybu může využívat různé druhy pohybů, a ty mu umožňují dostat se na určenou pozici více způsoby. Zde budou uvedeny základní druhy pohybů robotického ramene UR3 společnosti Universal robots, které bude podrobně popsáno v praktické části této práce. Základní druhy pohybu robota se dělí na:

- pohyb kloubový – někdy též označován jako „PohybJ“,
- pohyb lineární – někdy též označován jako „PohybL“,

- pohyb smíšený – někdy též označován jako „PohybP“, je druh speciálního lineárního pohybu s konstantní rychlostí a kruhovými kombinacemi.

Pohyb kloubový

„PohybJ“, někdy též označovaný jako pohyb kloubový, se snaží počítat dráhu pohybu mezi klouby robota. Tento typ pohybu provádí veškeré pohyby robota tak, aby všechny klouby dosáhly požadované koncové polohy současně. Obrázek 1.11 znázorňující tento pohyb ukazuje, že trajektorie tohoto typu pohybu je křivka. Mezi sdílené parametry požívané u tohoto pohybu patří zejména maximální rychlost a zrychlení kloubu. Při práci s klouby a jejich zrychlením a rychlostí, které se u tohoto typu pohybu využívají, jsou použité jednotky $deg \cdot s^{-1}$ a $deg \cdot s^{-2}$. Tento druh je jedním z nejrychlejších druhů pohybů, které se při práci s robotem používají, za předpokladu, že nezáleží na trase nástroje robota.

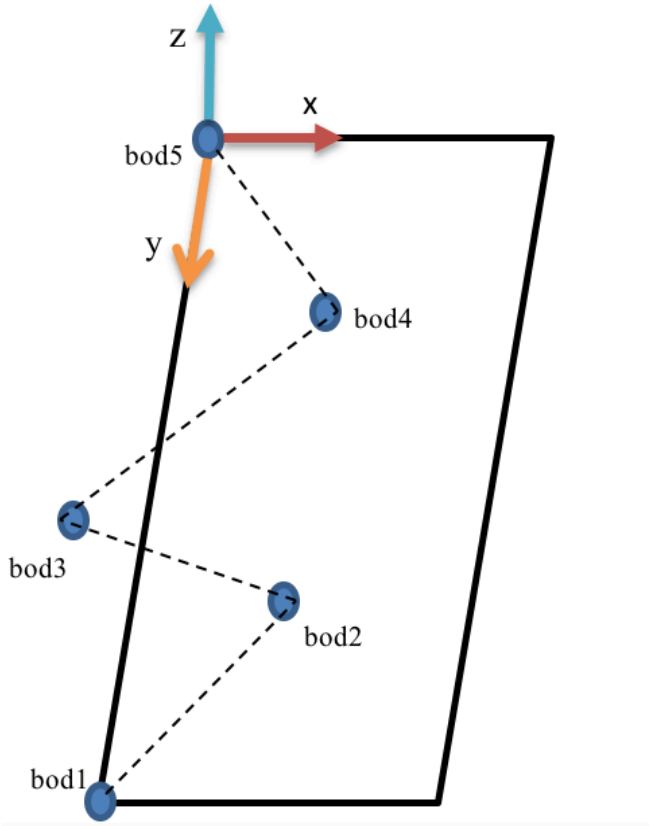


Obrázek 1.11 – Kloubový pohyb robota

Pohyb lineární

PohybL je někdy též označovaný jako pohyb lineární. Při lineárním pohybu se robot snaží provádět složité pohyby lineárně a pohybuje se tak mezi dvěma body, jako by kreslil

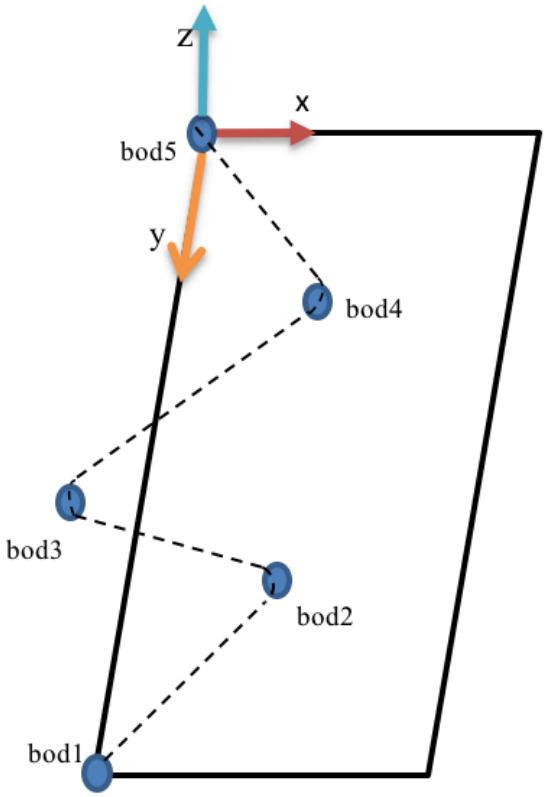
přímku. Tento druh pohybu je znázorněn na obrázku 1.12, na kterém je nakloněný trojrozměrný prostor a pohyb mezi pěti body je přímkový. Tento druh je nejbezpečnějším a nejjednodušším pohybem, který může robot vykonávat. Použití lineárního pohybu je vhodné zejména u těch robotů, u kterých je kladen důraz na pohyb po přesně definované dráze, nebo je zapotřebí preciznosti a opatrnosti. Sdílené parametry tohoto typu pohybu robota jsou rychlost a zrychlení nástroje v $mm \cdot s^{-1}$ a $mm \cdot s^{-2}$.



Obrázek 1.12 – Lineární pohyb robota

Pohyb smíšený

Jak již bylo zmíněno, tento typ je vlastně lineárním typem pohybu s přidanými kruhovými kombinacemi. Smíšený typ pohybu, který znázorňuje obrázek 1.13, se tedy pohybuje stejně jako v případě pohybu lineárního, ale využívá kruhové kombinace pro lepší obratnost robota. Při tomto pohybu se robot pohybuje konstantní rychlostí, a proto se výborně hodí zejména u speciálních druhů úloh, jako je lepení či dávkování.



Obrázek 1.13 – Lineární pohyb s kruhovými kombinacemi

2 PRAKTICKÁ ČÁST

V praktické části této práce a jejích příslušných oddílech a pododdílech čtenář nalezne několik známějších společností a jejich kooperujících zástupců, kteří jsou krátce popsáni. Dále je zde uvedené a podrobně popsané robotické rameno UR3 společnosti Universal Robots a jsou zde zmíněny a popsány nejdůležitější funkce jeho řídícího softwaru PolyScope. Dále je pak uvedené řešení softwaru pro ovládání robotického ramene z programu Matlab a podrobně popsáno, jakým způsobem probíhá komunikace mezi softwarem PolyScope a softwarem Matlab. V závěru této kapitoly i práce je ukázána jednoduchá úloha, založená na principu přenosu objektů, která prezentuje funkčnost softwarové komunikace.

2.1 KOOPERUJÍCÍ ROBOTICKÁ RAMENA

Při popisu kooperujících robotů v teoretické části této práce byly zmíněny důležité rozdíly mezi klasickým průmyslovým robotem a kooperujícím robotem. Není proto divu, že se v dnešní době objevují spíše roboti kooperující nežli klasičtí průmysloví roboti.

2.1.1 ABB

Robot YuMi s modelovým označením IRB 14000, znázorněný na obrázku 2.1, je dvouramenný průmyslový robotický pomocník, který je vyrobený švédsko-švýcarskou společnosti ABB. YuMi je plně kooperující robot, což dokazuje i jeho chytře zvolený název, který vznikl anglickou sloučeninou slov „you and me", neboli v překladu „ty a já". Tento robot je složený ze dvou na sobě nezávislých plně programovatelných ramen a mnoho dalších bezpečnostních prvků, což mu v případě nebezpečí umožnuje zastavení a následně opětovné spuštění v řádu milisekund.

Každé rameno je vyrobené z lehké hořčíkové slitiny, což mu dodává lehkost a obratnost. Rovněž se ramena dokáží ohýbat v sedmi osách za účelem efektivního napodobení pohybů lidské paže. Pracovní rozsah ramen je 500 mm a přesnost opakovatelného nastavení polohy je $\pm 0{,}02$ mm. Celková hmotnost robota YuMi je ± 38 kg a každé rameno disponuje užitečným zatížením 0,5 kg. YuMi se díky dvěma nezávislým ramenům výborně hodí zejména u činností, kde je vyžadována preciznost, ale není zapotřebí příliš velká zvedací schopnost robota, jako je tomu u výroby v odvětví spotřební elektroniky (YuMi, 2015).

Obrázek 2.1 – Kooperující robotické rameno YuMi společnosti ABB (ABB YuMi® - IRB 14000, 2018)

2.2 KUKA

Průmyslový robot s označením LBR iiwa 7 R800 od německé společnosti KUKA, znázorněný na obrázku 2.2, je dalším schopným a velice žádaným pomocníkem patřícím do skupiny plně kooperujících robotických ramen. Tento robot disponuje jedním plně ovladatelným ramenem s pracovním rozsahem 800 mm. Tento robot se výborně hodí do průmyslové výroby, neboť disponuje přesností opakovatelného nastavení polohy $\pm 0{,}1$ mm a díky senzorům manipuluje s díly opatrně a dokáže se intuitivně vyhýbat překážkám. Rovněž tento robot, podobně jako předchozí rameno, disponuje možností ovládání ramene v 7 osách. Celková hmotnost robota je ± 22 kg a samotná nosnost robota je 7 kg. Doporučená montážní plocha robota je na podlahu, strop či stěnu (KUKA LBR iiwa, 2018).

Obrázek 2.2 – Kooperující robotické rameno společnosti KUKA (Collaborative robotics, 2018)

2.2.1 FANUC

Průmyslový robot s označením modelu CR-35iA od společnosti FANUC, znázorněný na obrázku 2.3, je dalším zástupcem plně kooperujícího průmyslového robota. Tento robot, podobně jako předchozí dvě zmíněná ramena, disponuje jedním plně ovladatelným ramenem, ale při srovnání nabízí velké rozdíly. Celková hmotnost ramene FANUC CR-35iA je ± 990 kg, ale zato manipulační nosnost robota je až 35 kg. Co se přesnosti opakovatelného nastavení polohy týče, ani tento robot nezůstává pozadu a nabízí přesnost $\pm 0{,}03$ mm. Robota lze ovládat v šesti osách a dosah robotického ramene je až 1813 mm. Tento model kooperujícího robota je díky své schopnosti zvedat těžší předměty vhodný spíše k těžším a náročnějším manipulacím a proto, je nutné jej umístit na zem (Collaborative Robot, 2011–2017).

Obrázek 2.3 – Kooperující robotické rameno společnosti FANUC (CR-35iA, 2018)

2.3 UNIVERSAL ROBOTS

Robotické rameno UR3, znázorněné na obrázku 2.4, je nejmladší a nejmenší zástupce z rodiny robotů dánské společnosti Universal Robots. UR3 patří do zvláštní skupiny kooperujících robotických ramen popsaných v teoretické části práce, které jsou určeny k přímé interakci s lidmi. Zejména proto je tento model robota univerzálním a opravdu všestranným pomocníkem v mnoha oblastech průmyslu. Rameno UR3 se výborně hodí zejména u montážních prací, jako jsou např. lepení, leštění, šroubování nebo třeba pájení. Své uplatnění si ale nachází i u mnohem honosnějších pracovišť, jako jsou např. farmaceutická, vědecká nebo elektrotechnická pracoviště. Model robota UR3 je speciálně navržený tak, aby mohl pracovat i ve stísněném prostoru a v případě potřeby jej lze umístit i na pracovní stůl (Universal Robots, 2018).

Obrázek 2.4 – Kooperující robotické rameno společnosti Universal Robots

Jak již bylo zmíněno, rameno UR3 patří do skupiny plně kooperujících robotů, a proto je jeho využití dosti všestranné. UR3 ale neumožnuje jenom nahrazení lidských pracovníků tam, kde se jedná o jednotvárné operace nebo nebezpečné prostředí, ale také umožnuje přímou spolupráci s člověkem, a to v těsné blízkosti. UR3 má k dispozici 15 nejrůznějších vyspělých a

nastavitelných bezpečnostních prvků, které z něj činí prvotřídního spolupracovníka téměř pro každou činnost.

2.3.1 Pohybové vlastnosti ramene

Universal robots UR3 je příkladem sériového robota s otevřeným kinematickým řetězcem. Pohybové vlastnosti ramena jsou dány otáčivým podstavcem, který má pracovní rozsah $\pm 360°$. Dále se robot skládá z ramene, loktu a tří zápěstí. Dvě ze tří zápěstí je rovněž možné otáčet o $\pm 360°$ a třetí koncové zápěstí disponuje nekonečným otáčením z důvodu maximálního využití koncového nástroje. Maximální rychlost všech pohybových segmentů je $360° \cdot s^{-1}$. Tento robot se tedy vyznačuje šesti otáčivými stupni volnosti, a to mu zajišťuje bezproblémový translační i rotační pohyb v prostoru (Universal Robots, 2018).

2.3.2 Fyzické vlastnosti ramene

Fyzické rozměry robota jsou $\emptyset\, 180\,\text{mm}$ a je vyroben kombinací hliníku a polypropylenového plastu, což ho činí lehkého, rychlého a obratného. Pracovní dosah ramene robota UR3 je $500\,\text{mm}$ od kloubu základny. Celková hmotnost robota je přibližně $11\,\text{kg}$ včetně kabelu a samotná manipulační nosnost robota je $3\,\text{kg}$. Montáž robota je libovolná s možností umístění i na pracovní stůl. Robot může pracovat v teplotním rozsahu $0 - 50\,℃$, ale při nepřetržitém provozu robota s plnou rychlostí jednotlivých kloubů se povolený teplotní rozsah může snížit. UR3 nabízí opakovatelnou přesnost $\pm 0{,}1\,\text{mm}$. Příkon robota se pohybuje v rozmezí minimální $90\,\text{W}$, typický $125\,\text{W}$a maximální udávaný příkon je $250\,\text{W}$ (Universal Robots, 2018).

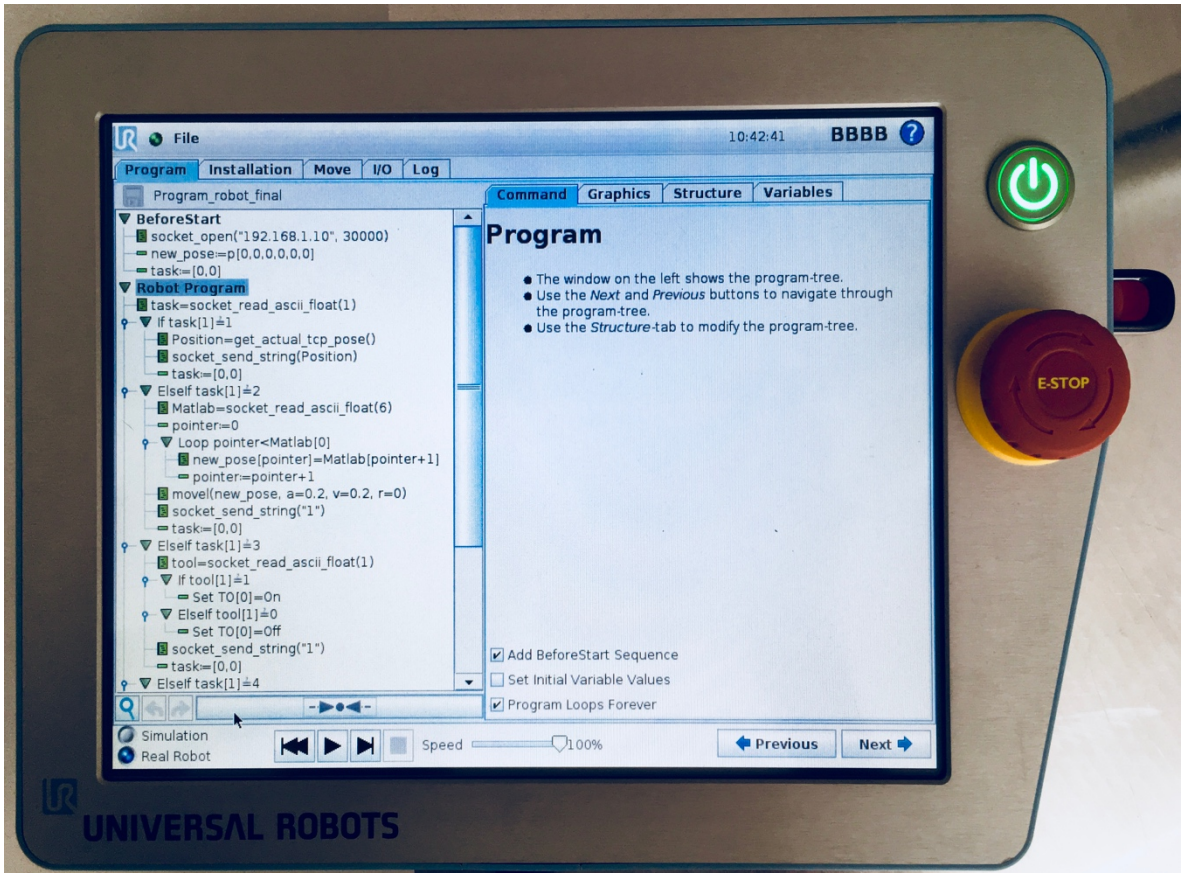
2.3.3 PolyScope

PolyScope, znázorněný na obrázku 2.5, je grafické programovací rozhraní, které je připojené k řídící jednotce robota a slouží k ovládání a programování robotického ramene UR3. PolyScope je sofistikovaný a všestranný nástroj, který při svém spuštění nabídne uživateli úvodní obrazovku, zobrazenou na obrázku 2.5, se základní nabídkou možností, která je podrobně popsána pod obrázkem 2.5.

Obrázek 2.5 – Úvodní obrazovka PolyScope

- Spustit program – tato možnost umožnuje uživateli nejjednodušší způsob, jak uvést robota do pohybu nebo ovládat pohyb robota. Dále je v této nabídce možné ovládat vstupně výstupní zařízení jako např. nástroj robota. Je zde také možnost zobrazení protokolu o technickém stavu robota.
- Naprogramovat robota – při volbě této nabídky se uživateli nabídne možnost načíst existující program z paměti robota, nebo z externího uložiště. Tato volba rovněž slouží k přeprogramování načteného programu, nebo lze použít existující šablonu. Dále má uživatel možnost provádět nejrůznější instalace robota jako např. konfiguraci nástroje, bezpečnostních prvků nebo vstupně výstupních zařízení robota.
- Nastavení robota – tato volba umožnuje uživateli změnit důležité nastavení. Slouží např. k inicializaci robota, je zde možné změnit jazyk programu, aktualizaci softwaru nebo k nastavení sítě atp.
- Vypnout robota – tato volba vypne robotické rameno, ovládací jednotku i program PolyScope.

Software PolyScope běží na dotykovém zařízení, které zobrazuje obrázek 2.6. Zmíněnému dotykovému zařízení se říká „Teach Pendant“ a slouží k pohodlnému a jednoduchému ovládání celého ramene pomocí dvanácti palcové plně dotykové obrazovky.



Obrázek 2.6 – Teach pendant

Teach Pendant nabízí mimo jiné také tlačítko pro spuštění režimu volného pohybu nebo tlačítko zvané E-STOP, které umožnuje nouzové zastavení robota v případě problému.

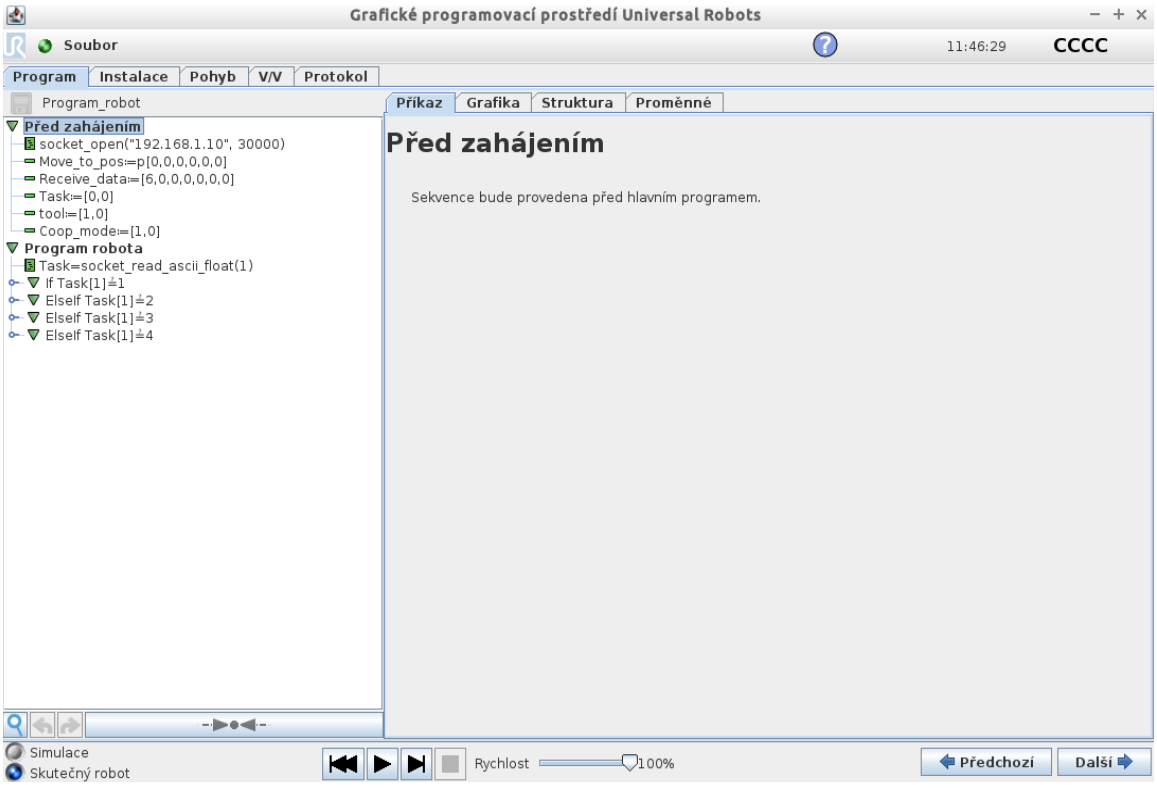
2.3.4 Programování robota

Při výběru možnosti naprogramovat robota se uživateli nabídne obrazovka znázorněná na obrázku 2.7, ve které lze načtený program různě editovat nebo lze vytvořit program nový. Na levé straně obrazovky vidí uživatel svůj načtený programový strom. Pravá strana obrazovky pak nabízí uživateli novou nabídku možností skládající se z karet:

- Příkaz – tato volba zobrazuje vybraný příkaz z programového stromu a umožnuje uživateli příkaz přepsat, popřípadě nějakým způsobem upravit. Například je zde možné přiřadit příkazu jinou proměnnou atd.

- Grafika – při výběru této možnosti je zobrazeno grafické znázornění aktuálního programu. Trasa TCP je zde znázorněná ve 3D zobrazení.
- Struktura – tato možnost nabízí uživateli editor struktury programu, který umožnuje programátorovy vložit, kopírovat, přesunout nebo odstranit daný typ příkazu. Také je zde na výběr ze základních a pokročilých druhů příkazů, mezi které patří podmínky, cykly atd.
- Proměnné – karta proměnné obsahuje nejrůznější hodnoty použitých proměnných v programu. Proměnné se zde zobrazují pouze tehdy, obsahují-li nějaké informace, které by měl programátor sledovat.

Mimo již zmíněné nabídky a příkazy je na této obrazovce možné např. změnit rychlost robota pomocí posuvníku v dolní části obrazovky, nebo pozastavit či opětovně spustit robota.



Obrázek 2.7 – Nabídka pro editaci programu v PolyScope

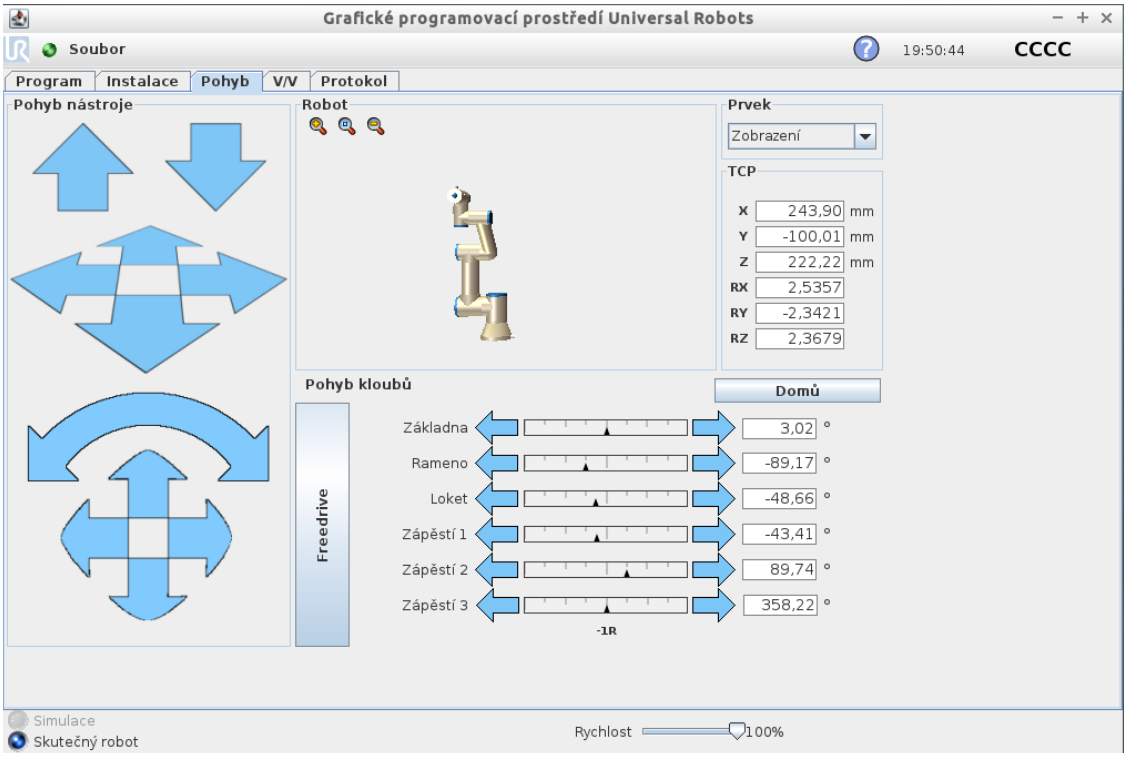
2.3.5 Pohyb robota v prostředí PolyScope

Jak již bylo zmíněno, PolyScope je sofistikovaný a všestranný nástroj, který nabízí spoustu možností pro programování a ovládání robotického ramene, včetně ovládání jeho pohybu. Obrázek 2.8 zobrazuje právě pohybovou nabídku prostředí PolyScope. V teoretické části této práce bylo zmíněno, že pohyb robota se skládá z části posuvné a rotační. Při pohybu

robotického ramene uživatele asi nejvíce zajímá poloha TCP. Rameno UR3 prezentuje polohu TCP pomocí vektoru pozice, který se skládá ze šesti souřadnic ***x, y, z, rx, ry, rz***.

Souřadnice ***x, y, z*** udávají vektor translační polohy TCP, které jsou dle zvyklostí prezentovány v kartézském souřadném systému a PolyScope je zobrazuje v milimetrech. Další trojice ***rx, ry, rz*** zase udává orientaci TCP a ta je dána jako vektor rotace. Vektor rotace lze v PolyScope zobrazit v radiánech nebo ve stupních. Další možností, jak prezentovat rotaci nástroje, je pomocí RPY a rovněž ji lze zobrazit ve stupních či radiánech. RPY neboli roll, pitch a yaw, o kterých byla zmínka v teoretické části při popisu rotačního pohybu robota, je způsob reprezentace rotace TCP, pomocí úhlů náklonu, sklonu a zatáčení. RPY je tedy jedním ze způsobů reprezentace orientace, který slouží pro lepší představu při rotaci nástroje robota. Kombinací vektorů rotace a polohy vznikne vektor pozice, který je v PolyScope zobrazen jako p[***x, y, z, rx, ry, rz***] (Universal Robots, 2018).

Na levé straně obrazovky znázorněné na obrázku 2.8, je ohraničená nabídka zvaná pohyb nástroje, která umožnuje měnit polohu TCP robota. Ovládání pohybu nástroje je jednoduché a provádí se pomocí šipek. Podobně jak se dá ovládat pohyb nástroje, se zde nabízí i ovládání jednotlivých kloubů ramene, pomocí nabídky zvané „pohyb kloubů“. Dalším velmi užitečným nástrojem je tlačítko „Free drive“, které umožnuje přesouvat robota do zvolené pozice ručně. Veškerý prováděný pohyb je zobrazován ve 3D. Pravá strana obrazovky



Obrázek 2.8 – Pohyb robota v software PolyScope

zobrazuje souřadnice TCP, které je po kliknutí možné zadávat manuálně (Universal Robots, 2018).

2.3.6 Srovnání kooperujících ramen

V tomto a předchozím oddíle byly uvedeny některé známé společnosti a jejich kooperující ramena. Tabulka 2.1 znázorňuje jednoduché srovnání zde zmíněných ramen, ze které je zřejmé, že fyzické vlastnosti ramen jsou si dost podobné. Výjimkou je rameno CR-35iA společnosti FANUC, který nabízí mnohonásobně větší pracovní prostor, ale zato je mnohem těžší a je nutné jej umístit na zem. Rameno také disponuje o dost větším pracovním rozsahem, nežli je tomu u zbylých tří ramen.

Tabulka 2.1 – Srovnání kooperujících ramen

Rameno	Hmotnost ramene, kg	Zvedací síla ramene, kg	Přesnost nastavení polohy, mm	Pracovní dosah ramene mm
YuMi	38	0,5	0,02	500
LBR iiwa	22	7	0,1	800
CR–35iA	990	35	0,03	1813
UR3	11	3	0,1	500

2.4 SW PRO OVLÁDÁNÍ RAMENE Z MATLABU

Jak již bylo zmíněno v oddílu 2.3 o popisu robotického ramene, UR3 je ovládáno a programováno pomocí řídícího softwaru PolyScope. Abychom tedy byli schopni ovládat rameno UR3 pomocí programu Matlab, je zapotřebí vytvořit jednoduchý program v prostředí PolyScope a propojit ho s programem Matlab. Spojení je navázáno pomocí TCP/IP komunikace, která uživateli umožní zasílání jednoduchých příkazů právě z prostředí Matlab.

V této části práce čtenář nalezne podrobný popis funkcí a příkazů, použitých při řešení softwarové komunikace s robotem. Tento oddíl tedy obsahuje několik pododdílů, načež každý obsahuje podrobný popis kódů komunikace pro jednotlivé úlohy robota. Nejdříve je ukázaný kód v softwaru PolyScope a následně kód pro navázání komunikace a odeslání požadavku ze softwaru Matlab. Dále je pak uvedený skript v programu Matlab, ve kterém jsou ukázány jednotlivé ukázkové aplikace robota, které využívají jednotlivých funkcí, popsaných v tomto oddílu.

Cílem této komunikace je tedy vzdálené ovládání některých základních funkcí robota z programu Matlab, mezi které patří zejména:

- zjištění aktuální polohy nástroje robota,
- translační pohyb a rotace koncového efektoru robota,
- sepnutí a rozepnutí nástroje,
- aktivace nebo deaktivace volného režimu robota,
- vzdálené vypnutí robota a prostředí PolyScope.

2.4.1 TCP/IP komunikace

Softwarová komunikace probíhá mezi programem Matlab a PolyScope, a proto je v první řadě nutné definovat spojení pomocí TCP/IP soketu. Kód pro definici TCP/IP spojení v programu Matlab bude popsán v následujícím samostatném oddílu při popisu skriptu ukázkových aplikací. V PolyScope je soket otevřen pomocí příkazu `socket_open`, který pracuje s dalšími parametry a je mu tak řečeno, aby se připojil k serveru s IP adresou 192.168.1.10 na portu 30 000.

Dále je zde také definovaná proměnná zvaná *task*, která se v průběhu mění a slouží jako ukazatel probíhající úlohy. Další proměnná, kterou je zde nutné definovat, je proměnná zvaná *new_pose*, která reprezentuje, v jakém tvaru robot pracuje se svými souřadnicemi. Do této proměnné se rovněž ukládají přijaté souřadnice ze soketu TCP/IP spojení za účelem přesunu robota do nových souřadnic. Kód pro vytvoření soketu pro spojení PolyScope k Matlabu je následující:

Kód v PolyScope

```
socket_open ("192.168.1.10", 30000)
new_pose ≔ p[0, 0, 0, 0, 0, 0]
task ≔ [0, 0]
```

2.4.2 Volba úlohy robota

K volbě úlohy robota slouží proměnná zvaná *task*, která může nabývat celočíselných hodnot 0 až 5. Naplnění hodnoty proměnné *task* a výběr daného typu úlohy robota v PolyScope je zařízeno pomocí příkazu `socket_read_ascii_float`. Teno příkaz čte číselnou hodnotu přijatou ze soketu a očekává právě jedno číslo v rozmezí 0 až 5. Pro zadávání

číselné hodnoty soketu slouží proměnná *t*, která je vstupním parametrem každé funkce v Matlabu a bude o ní zmínka při popisu prováděných úloh robota.

Dalším krokem po přijetí číselné informace je porovnání přijaté číselné hodnoty uložené v proměnné *task* pomocí podmíněných příkazu `If` a `ElseIf`. Následuje kontrola splnění některé z podmínek a je vybrána a provedena požadovaná úloha. Kód pro naplnění proměnné *task* a volbu požadované úlohy robota v programu PolyScope je následující:

Kód v PolyScope

```
task = socket_read_ascii_float (1)
```

2.4.3 Poloha robota

Jednou z číselných hodnot, kterou může proměnná *task* v PolyScope nabývat, je hodnota 1. Nastavením proměnné *task* na hodnotu 1 se iniciuje úloha, která používá příkaz `get_actual_tcp_pose`, která vyčte aktuální polohu nástroje robota a uloží ji do proměnné *position*. V následujícím kroku je použit příkaz `Socket_send_string`, který zařídí odeslání obsahu proměnné *position* jako řetězec soketem. Jak již bylo zmíněno v pododdílu 2.3.5 o pohybu robota v prostředí PolyScope, robot prezentuje polohu nástroje v milimetrech, a proto je zde potřeba si dávat pozor, neboť poloha uložená v proměnné *position* je v metrech. Dalším krokem je vynulování proměnné *task* a program se vrací do čekací fáze, kde čeká na volbu úlohy. Kód úlohy pro získání polohy nástroje v programu PolyScope je následující:

Kód v PolyScope

```
ElseIf task[1] ≟ 1
position = get_actual_tcp_pose()
Socket_send_string(position)
task ≔ [0,0]
```

Abychom mohli z Matlabu robotovi vzdáleně přikázat, aby se někam přesunul, je důležité vědět, kde se robot nachází. Funkce v Matlabu, která má za úkol vzdáleně nastavit požadovanou úlohu robota, pracuje se vstupním parametrem *t*. Tento parametr je proměnná, která slouží jako ovládací prvek soketu. V tomto případě se do proměnné *t* zapíše hodnota 1 a pošle se přes soket. Tato hodnota proměnné vyhoví podmínce v programu PolyScope a nastaví úlohu robota pro získání polohy nástroje. Funkce dále čeká pomocí cyklu „while“, než bude proměnná rovna hodnotě 0, čímž je zaručena posloupnost příkazů.

Robot provede svoji úlohu, viz kód v PolyScope výše, a pošle zpět souřadnice polohy a orientace nástroje. Souřadnice polohy jsou poslány v metrech a orientace nástroje pak v radiánech. Souřadnice polohy a orientace nástroje jsou následně uloženy do proměnné *Odpoved* a je provedena kontrola obdrženého výsledku. V případě, že je výsledek ve správném formátu, převede se pomocí příkazu `str2num` z řetězce na čísla a souřadnice polohy nástroje jsou následně pro pohodlnější představu a lepší zadávání v řídícím skriptu převedeny na milimetry a uloženy do výstupní proměnné této funkce zvané `Souradnice`. Kód pro zjištění aktuální pozice a orientace nástroje je následující:

Kód v Matlabu

```
function [Souradnice] = SoradniceTcp(t)
fprintf(t, '(1)');

while t.BytesAvailable == 0
end

Odpoved = fscanf(t, 'c', t.BytesAvailable);

If ~ strcmp(Odpoved(1), 'p') || ~ strcmp(Odpoved(end), ']')
error ('Chybně poslané souřadnice!')
else
Odpoved = Odpoved(2: end);
Souradnice = str2num(Odpoved);
Souradnice(1:3) = Souradnice(1:3) *1000;
end
```

2.4.4 Pohyb robota

Pro nastavení pohybové úlohy robota je zapotřebí, aby hodnota proměnné *task* v programu PolyScope byla nastavená na hodnotu 2. V případě, že je této podmínce vyhověno, příkaz `socket_read_ascii_float` zařídí naplnění proměnné zvané *new_pose* a to tak, že očekává šest hodnot ze soketu TCP/IP komunikace. Přijatá šestice hodnot je právě šesticí souřadnic prezentujících polohu translace a rotace nástroje robota, popsaných v pododdílu 2.3.5 o pohybu robota v prostředí PolyScope. Dalším použitý příkazem je `movel`, který definuje lineární typ pohybu se zrychlením 0,1 $mm \cdot s^{-2}$, rychlostí 0,1 $mm \cdot s^{-1}$ a přesunem na souřadnice

definované v proměnné *new_pose*. Po provedení pohybu robota na určenou polohu je příkazem `Socket_send_string` poslána číselná hodnota 1, která informuje uživatele v programu Matlab o úspěšnosti této úlohy. Následně je proměnná *task* vynulována a program se opět vrací k volbě úlohy, kde čeká na zaslání číselné hodnoty. Kód pro pohyb robota v programu PolyScope je následující:

Kód v PolyScope

```
If task[1] ≟ 2
new_pose = socket_read_ascii_float(6)
movel(new_pose, a = 0.1, v = 0.1, r = 0)
Socket_send_string("1")
task ≔ [0, 0]
```

V pododdílu 2.3.5 o popisu pohybu v řídícím programu PolyScope bylo zmíněno, že rameno UR3 prezentuje polohu TCP nástroje jako šest souřadnic, z nichž první tři reprezentují vektor translačního pohybu ramene v milimetrech a zbylé tři reprezentují vektor natočení nástroje robota v radiánech. Pro zaslání požadavku ke změně polohy robota přes TCP/IP soket je nutné dbát na to, abychom posílali souřadnice ve správném tvaru a ve správných jednotkách.

Pro vzdálenou aktivaci pohybové úlohy z programu Matlab je opět nutno nastavit hodnotu proměnné *t* na požadovanou hodnotu, podobně jako v pododdílu 2.4.3. V tomto případě se hodnota proměnné *t* nastaví na hodnotu 2, čímž opět vyhoví dané podmínce v programu PolyScope. Další dva vstupní parametry této funkce jsou proměnné *Nova_translace* a *Nova_rotace*. Do těchto proměnných se v řídícím skriptu zadávají nové hodnoty souřadnic translace a rotace, způsob naplnění proměnných bude ukázán v následujícím oddílu 2.5. Nové hodnoty souřadnic se uloží do proměnné *Pozice* a ty jsou následně pomocí příkazu `num2str` převedeny na tvar, kterému robot rozumí. Nově vzniklé souřadnice jsou převedeny na metry a opět poslány soketem. Následně se čeká na dokončení úlohy robota a po dokončení úlohy se čte proměnná *t*, kde se očekává návratová hodnota 1, která informuje o úspěšnosti operace. Kód funkce pro odeslání pohybového požadavku je následující:

Kód v Matlabu

```
Function [Odpoved] = PohybRobota(t, Nova_translace, ...
Nova_rotace)
fprintf(t, '(2)');
Pozice = [Nova_translace, Nova_rotace];
```

```
Pozice(1:3) = Pozice(1:3) * 0.001;
Pozice_retezec = ['(', num2str(Pozice(1)), ',', ...
num2str(Pozice(2)), ',', num2str(Pozice(3)), ',', ...
num2str(Pozice(4)), ',', num2str(Pozice(5)), ',', ...
num2str(Pozice(6)),')'];
fprintf(t, Pozice_retezec);

while t.BytesAvailable == 0
end

Odpoved = fscanf(t, '%c', t.BytesAvailable);

if ~strcmp(Odpoved, '1')
Error ('Chybná pozice robota!')
else
disp ('Robot změnil pozici na nové souřadnice');
end
end
```

2.4.5 Ovládání nástroje

Další nezbytností při použití robota je ovládání jeho nástroje, v tomto případě se jedná o spínání nebo rozepínání relé. Chce-li uživatel vzdáleně ovládat nástroj robota, musí nastavit hodnotu proměnné *task* v PolyScope na hodnotu 3. V případě, že je této podmínce vyhověno, použije se příkaz `socket_read_ascii_float`, který očekává číselnou hodnotu 1 nebo 0. Tato hodnota se uloží do proměnné *tool* a jestliže byla obdržena hodnota 1 dojde k nastavení (`TO` – „Tool Output") na logickou hodnotu „True", čímž se nástroj aktivuje neboli sepne. V opačném případě je (`TO`) nastavený na logickou hodnotu „False" a dojde k rozepnutí nástroje. Po sepnutí či rozepnutí nástroje je pomocí příkazu `Socket_send_string` odeslán uživateli znak 1, čímž robot zasílá uživateli informaci o dokončení úlohy. Následně dojde k vynulování proměnné *task* a program v PolyScope se opět vrací do čekací fáze. Část kódu pro řízení stavu nástroje v prostředí PolyScope je následující:

Kód v PolyScope

```
ElseIf task[1] ≟ 3
```

```
Tool = socket_read_ascii_float(1)
If tool[1] ≟ 1
Set TO[0] = On
ElseIf tool[1] ≟ 0
Set TO[0] = Off
Socket_send_string("1")
Task ≔ [0,0]
```

Funkce pro nastavení vzdáleného ovládání nástroje robota zde využívá kromě proměnné soketu *t* ještě další vstupní parametr zvaný *nastav*. Tato proměnná slouží k nastavení stavu nástroje a nastavuje se v hlavním ovládacím skriptu, který bude podrobně popsaný v samostatném oddílu 2.5. Proměnná *nastav* může nabývat hodnot 0 pro rozepnutí nástroje, nebo 1 pro sepnutí nástroje. Řídící proměnná soketu *t* je v tomto případě nastavená na hodnotu 3 a poslána soketem robotovi. Tato hodnota vyhoví podmínce v PolyScope a spustí úlohu pro ovládání nástroje. Stejně jako v předchozích funkcích se čeká na dokončení úlohy a na návratovou hodnotu robota, která informuje o úspěšnosti úlohy. Kód pro ovládání nástroje z programu Matlab je jednoduchý a je následující:

Kód v Matlabu

```
function [Odpoved] = NastrojRobota(t, nastav)
fprintf(t,'(3)');
pause(0.1);
fprintf(t, nastav);

while t.BytesAvailable == 0
end

Odpoved = fscanf(t, '%c', t.BytesAvailable);

if ~strcmp(Odpoved,'1')
error ('Chyba při ovládání nástroje!')
end
```

2.4.6 Režim volného pohybu

Režim volného pohybu robota neboli „free drive mode“, je velmi užitečným a mocným nástrojem, který umožnuje volně pohybovat s každým kloubem robota do příslušné polohy. Použití tohoto režimu je vhodné zejména jedná-li se o situaci, kdy uživatel potřebuje volně manipulovat s ramenem robota a přesunout ho do nějaké polohy bez znalosti příslušných souřadnic.

Vzdálená aktivace tohoto režimu je možná nastavením číselné hodnoty 4 do proměnné *task* v programu PolyScope. Je-li číselná hodnota správná, a iniciuje se úloha volného pohybu robota, je nutné ještě nastavit proměnnou *free_drive_mode* na hodnotu 1 nebo 0 a určit, jestli dojde k aktivaci či deaktivaci tohoto režimu. Naplnění této proměnné je podobně jako u ovládání nástroje docíleno příkazem `socket_read_ascii_float`, čímž je očekávaná právě jedna hodnota ze soketu TCP/IP spojení. Je-li příchozí požadavek hodnota 1, použije se příkaz `teach_mode`, čímž dojde k aktivaci příslušného režimu. V opačném případě je použit příkaz `end_teach_mode`, který režim volného pohybu vypne. V závěru je uživatel v Matlabu opět informován o úspěšnosti úlohy a program v PolyScope se vrací do čekací fáze. Kód pro aktivaci, respektive deaktivaci volného režimu v PolyScope je následující:

Kód v PolyScope

```
ElseIf task[1] ≟ 4
free_drive_mode=socket_read_ascii_float(1)
If free_drive_mode[1] ≟ 1
teach_mode()
ElseIf free_drive_mode[1] ≟ 0
end_teach_mode()
Socket_send_string("1")
task ≔ [0,0]
```

Funkcionalita a struktura funkce v Matlabu pro vzdálenou aktivaci režimu volného pohybu je stejná jako při ovládání nástroje. Tentokrát se do proměnné *t* zapíše hodnota 4 a pošle se přes soket, čímž dojde k výběru požadované úlohy v PolyScope. Také tahle funkce má ještě další vstupní parametr zvaný *nastav*, který slouží, podobně jako tomu bylo v případě nastavení nástroje, k ovládání stavu režimu volného pohybu robota a nastavuje se v hlavním ovládacím skriptu, který bude podrobně popsán v následujícím oddílu 2.5. Kód pro vzdálenou aktivaci volného režimu z Matlabu je následující:

Kód v Matlabu

```
function [Odpoved] = RezimVolnehoPohybu(t, nastav)
fprintf(t, '(4)');
pause(0.5);
fprintf(t, nastav);

while t.BytesAvailable == 0
end

Odpoved = fscanf(t, '%c', t.BytesAvailable);

if ~strcmp(Odpoved,'1')
    error ('Chyba při nastaveni režimu volného pohybu!')
end
end
```

2.4.7 Vypnutí robota a rozhraní PolyScope

Poslední úlohou v programu PolyScope je vypnutí celého robota, PolyScope a teach pendantu. Abychom docílili vypnutí robota, je nutné nastavit do proměnné *task* hodnotu 5. Při splnění této podmínky dojde k volání příkazu `Socket_send_string`, čímž je uživateli v programu Matlab odeslána hodnota 1, která slouží jako zpětná vazba o přijetí požadavku na vypnutí robota. Následně se použijí příkazy `socket_close` a `powerdown`, čímž je soket ukončen a dochází k vypnutí robota a PolyScope. Kód pro vypnutí robota je následující:

Kód v PolyScope

```
Socket_send_string("1")
socket_close()
powerdown()
```

Rovněž funkce v Matlabu pro vzdálenou aktivaci pracuje se vstupní hodnotou proměnné soketu *t*, do které se v tomto případě zapíše hodnota 5. Číselná hodnota je následně odeslána soketem do PolyScope, kde iniciuje příslušnou úlohu robota. V dalším kroku se opět čeká na příchozí odpověď robota o úspěchu či neúspěchu operace. Kód v Matlabu pro vzdálené odeslání požadavku pro vypnutí robota je následující:

Kód v Matlabu

```
function [Odpoved] = VypnoutRobota(t)
fprintf(t,'(5)');
pause(0.5);
Odpoved = fscanf(t, '%c', t.BytesAvailable);

if ~strcmp(Odpoved,'1')
    error ('Chyba při vypnuti robota!')
else
    disp ('Robot je odpojeny a bude vypnut!');
end
end
```

2.5 UKÁZKOVÉ APLIKACE

V tomto oddílu je popsán skript v softwaru Matlab, který obsahuje popsané ukázkové aplikace a pomocí kterého lze vybírat z jednotlivých základních funkcí robota. Skript potom na základě volby uživatele volá pomocné funkce, které byly popsány v předchozím oddílu o softwaru ovládání ramene.

Úkolem tohoto řídícího skriptu je nabídnout uživateli základní nabídku úloh, které je možné ze softwaru Matlab ovládat a umožnit i zadávání některých důležitých informací, jako jsou např. souřadnice polohy a orientace nástroje, ovládání režimu volného pohybu, nebo ovládat stav nástroje atd. Dalším užitečným nástrojem řídícího skriptu je detekce kulatých objektů za účelem stanovení jejich translačních souřadnic. Toto řešení využívá webkamery a umožnuje detekovat a určit translační souřadnice kulatých objektů za účelem jejich manipulace s ramenem. Zde budou uvedeny a popsány jednotlivé úlohy řídícího skriptu a zmíněny jaké doplňkové funkce volá.

2.5.1 TCP/IP komunikace

První věc, kterou je nutné v řídícím skriptu definovat ještě před samotným zasíláním jakéhokoliv požadavku, je navázání spojení s ramenem. Jak již bylo zmíněno v předchozím oddílu o softwaru pro spojení s ramenem, spojení se provádí pomocí TCP/IP komunikace a zdrojový kód ve skriptu Matlabu je následující:

Kód v Matlabu

```
%% TCP/IP spojení s ramenem
```

```
disp ('Vítejte v programu pro vzdálené ovládaní robotického
ramene UR3, Prosím počkejte na spojeni s robotem!');
Socket = tcpip('0.0.0.0', 30000, 'NetworkRole','server');
fclose(Socket);
disp ('Prosím stlačte tlačítko play v PolyScope')
fopen(Socket);
disp ('Spojeni s robotem je navázáno!');
```

Příkaz `tcpip` slouží k vytvoření objektu TCP/IP komunikace a pomocí parametrů mu je přiřazena role serveru. Další parametr „0.0.0.0“ zařídí, že náš server připojí první zařízení, které se pokusí navázat spojení na portu 30 000. TCP/IP spojení se vytvoří do proměnné *`Socket`* a jakákoliv další manipulace se soketem se provádí pomocí této proměnné. Následně je uživateli pomocí příkazové řádky oznámeno, aby stiskl tlačítko pro spuštění na rozhraní robota, tím dojde k otevření soketu za účelem komunikace s ramenem. Dále je pak uživateli pomocí příkazové řádky v Matlabu oznámeno, že bylo navázáno spojení s ramenem.

2.5.2 Poloha robota

V této ukázkové úloze je ukázáno, jakým způsobem lze robotovi poslat požadavek ke zjištění aktuálních souřadnic nástroje robota. K tomuto účelu slouží proměnná *`Pozice_TCP`*, která volá funkci `SouradniceTcp`, která byla popsána v předchozím oddílu. Tato funkce získá souřadnice polohy a orientaci nástroje robota a uloží je do již zmíněné proměnné *`Pozice_TCP`*. Souřadnice polohy a orientace nástroje jsou následně rozděleny na část translační a část orientační a jsou zobrazeny na obrazovku. Souřadnice translace jsou pro lepší představu zobrazovány v milimetrech. Kód pro zjištění a vypsání aktuálních souřadnic nástroje je následující:

Kód v Matlabu

```
%% Zjištění aktuální pozice TCP
Pozice_TCP = SoradniceTcp(Socket);
Translace = Pozice_TCP(1:3);
Orientace = Pozice_TCP(4:6);
str = sprintf('Aktuální souřadnice TCP jsou: %1.2f, %1.2f,
%1.2f, %1.2f, %1.2f, %1.2f', Pozice_TCP);
disp(str);
```

2.5.3 Translační pohyb robota

Další ukázková úloha prezentuje, jakým způsobem lze robotovi poslat požadavek pro translační pohyb. Potřebuje-li uživatel změnit polohu nástroje robota a umožnit tak jeho translační přesun na pozici novou, musí zadat nové souřadnice pro translační pohyb. Pro zadávání translačních souřadnic se dají použít dva způsoby. První z možností je zadávat nové souřadnice přímo a následně je poslat robotovi, který se do nich přesune. Tato volba zadávání má ale nevýhodou v nutnosti přesně znát pracovní prostor robota a současnou polohu robota.

Další možností změny translačních souřadnic je vzít souřadnice, které poslal robot a následně je jenom pozměnit. Tato volba má výhodu zejména v tom, že není nutné znát celý pracovní prostor robota, protože se k aktuální poloze jenom přičítá, nebo odečítá nějaká hodnota. Tato možnost zadávání je tudíž bezpečnější, nicméně při znalosti pracovního prostoru robota je první způsob účinnější, a proto je zde použit. Pro lepší představu pohybu se nové souřadnice pohybu zadávají v milimetrech a následně jsou před odesláním robotovi převedeny na metry. Kód pro zadávání translačních souřadnic je následující:

Kód v Matlabu

```
%% Zjištění aktuální pozice TCP a translační pohybová úloha
Pozice_TCP = SoradniceTcp(Socket);
Nova_rotace = Pozice_TCP(4:6);
x = input('Zadej souřadnice x: ');
y = input('Zadej souřadnice y: ');
z = input('Zadej souřadnice z: ');
Nova_translace = [x y z];
PohybRobota(Socket, Nova_translace, Nova_rotace)
```

Nejdříve je zapotřebí zjistit, kde se nástroj robota nachází. To je zařízeno stejným způsobem jako v předchozím případě. Po zadání nových translačních souřadnic se bude volat funkce pro pohyb robota, která byla popsána v předchozím pododdílu. Jak již ale bylo zmíněno, tato funkce pracuje i s orientací nástroje a je proto nutné definovat i ji. Abychom ale docílili změnu pouze translačních souřadnic, je z přijatých dat použita pouze orientační část souřadnic, která se uloží do proměnné *Nova_rotace* a nebude nijak změněna. Následně je uživatel vyzván, aby zadal nové souřadnice pro translační pohyb, které se uloží do proměnné zvané *Nova_translace*. Následně se zavolá funkce PohybRobota, která byla popsána v předchozím oddílu, která zařídí odeslání nových souřadnic robotovi. Tímto způsobem je

zachováno množství vstupních parametrů volané funkce a také je zaručeno, že se změní pouze souřadnice translačního pohybu a orientace nástroje zůstává nezměněna.

2.5.4 Rotační pohyb robota

Jak již z teoretické části této práce plyne, rotační pohyb je mnohonásobně složitější než pohyb translační. Také již bylo zmíněno, že rameno UR3 prezentuje rotaci nástroje jako trojici čísel v radiánech zvanou jako rotační vektor. Následující ukázková aplikace prezentuje, jakým způsobem lze docílit rotaci nástroje z programu Matlab. Při změně souřadnic rotačního pohybu se uživatel musí rozhodnout, v jaké ose otáčení bude nástroj rotovat. Zde zmíněné řešení rotačního pohybu je prováděno pomocí rotačních matic, které slouží k rotaci kolem nějaké osy souřadného systému, viz pododdíl 1.3.2 o rotačním pohybu robota v teoretické části této práce. Kód pro rotaci nástroje v Matlabu je následující:

Kód v Matlabu

```
%% Zjištění aktuální pozice TCP a rotační pohybová úloha
Pozice_TCP = SoradniceTcp(Socket);
Nova_translace = Pozice_TCP(1:3);
Aktualni_rotace = Pozice_TCP(4:6);
Rotacni_matice = vrrotvec2mat([Aktualni_rotace, ...
norm(Aktualni_rotace)]);
uhel = input('zadej velikost uhlu ve stupních: ');
fi = ((uhel)*pi/180);

% Rotační matice
rx = [1, 0, 0; 0, cos(fi), sin(fi); 0, -sin(fi), cos(fi)];
ry = [cos(fi), 0, -sin(fi); 0, 1, 0; sin(fi), 0, cos(fi)];
rz = [cos(fi), sin(fi), 0; -sin(fi), cos(fi), 0; 0, 0, 1];
natoceni = input('Kolem které osy chceš nastroj rotovat [rx,
ry, rz]? ');
Rotace = Rotacni_matice * natoceni;
Vector_rotace = vrrotmat2vec(Rotace (1:3,1:3));
Nova_rotace = Vector_rotace (4) *Vector_rotace (1:3);
PohybRobota(Socket, Nova_translace, Nova_rotace)
```

V první řadě je nutné znát aktuální natočení nástroje, a proto se před samotnou rotací volá funkce pro vypsání souřadnic pozice TCP, která je stejná jako v případě samotného požadavku pro získání souřadnic pozice robota. Po obdržení výsledku se přijaté souřadnice translace ignorují a pracuje se pouze s informací o aktuální orientaci nástroje uložené v proměnné *Aktualni_rotace*. V dalším kroku je zapotřebí vzít souřadnice orientace obdržené od robota a převést vektor rotace na příslušnou rotační matici. Tento převod je zařízen příkazem `vrrotvec2mat` a nově vzniklá rotační matice je uložena do proměnné *Rotacni_matice*. Na dalším řádku je uživatel vyzván k zadání velikosti úhlu ve stupních a hodnota se uloží do proměnné *uhel*. Následně je proveden přepočet úhlu ze stupňů na radiány a hodnota je uložena do proměnné zvané *fi*.

Jak již bylo zmíněno, zde uvedené řešení využívá k rotaci nástroje o libovolný úhel rotační matice, a proto je nutné je definovat. Následující tři řádky kódu proto obsahují jednotlivé rotační matice kolem každé osy souřadného systému, které byly rovněž podrobněji popsány v pododdíle 1.3.2 o rotačním pohybu robota v teoretické části práce. Uživatel je tedy opět vyzván ke zvolení jedné z nabídnutých možností, zde konkrétně kolem které osy má nástroj rotovat. Po zvolení jedné z možností se násobí rotační matice představující aktuální orientaci nástroje maticí ***Rx***, ***Ry*** nebo ***Rz***, které představují nové matice rotace o daný úhel v proměnné *fi*. Do proměnné *Rotace* se tedy uloží nová požadovaná matice rotace o daný úhel φ kolem zvolené osy souřadného systému.

Před posláním souřadnic soketem je ještě zapotřebí převést nově vzniklou matici na rotační vektor, se kterým robot umí pracovat. Převod je zařízen pomocí příkazu `vrrotmat2vec` a nově vzniklý vektor je uložen do proměnné *Vector_rotace*. V předposledním kroku je ještě nutné vynásobit první tři prvky vektoru čtvrtým prvkem, čímž získáme příslušný vektor rotace ve tvaru, v jakém je vyžadováno robotem. Posledním úkolem je zavolat funkci `PohybRobota`, který použije proměnnou *Nova_rotace* jako vstupní parametr a pošle souřadnice robotovi. Stejně jako v předchozím případě jsou souřadnice translační části nezměněny a mění se pouze orientace nástroje robota.

2.5.5 Ovládání nástroje

Ukázková aplikace pro ovládání stavu nástroje z řídícího skriptu je jednoduchá a uživateli stačí jenom definovat, jestli chce nástroj sepnout nebo rozepnout. Kód pro definici stavu a následné volání příslušné funkce je následující:

Kód v Matlabu

```
%% Interakce s nástrojem robota
disp('1 - Sepnout nástroj, 2 - Rozepnout nástroj');
uloha_nastroje = input('Vyberte jednu z možností: ');

if uloha_nastroje == 1
nastav = ('(1)');
NastrojRobota(Socket, nastav)
Pause(0.5);
elseif uloha_nastroje == 2
nastav = ('(0)');
NastrojRobota(Socket, nastav)
pause (0.5);
end
```

Uživatel je nejdříve prostřednictvím příkazu `input` vyzván k zadání číselné hodnoty 1, nebo 2. Zadaná hodnota je uložená do proměnné *`uloha_nastroje`*, která slouží k určení příslušné operace s nástrojem robota. V případě zadané hodnoty 1 se do proměnné *`nastav`* zapíše hodnota 1 a je volána funkce `NastrojRobota`, která zařídí odeslání požadavku pro sepnutí nástroje robota. V opačném případě se do proměnné *`nastav`* zapíše hodnota 0 a volaná funkce odesílá požadavek pro rozepnutí nástroje robota.

2.5.6 Režim volného pohybu

Tato ukázková aplikace zasílá robotovi požadavek pro aktivaci volného režimu. Řešení je založené na stejném principu, jako tomu bylo v případě ovládání nástroje. Kód pro aktivaci příslušného režimu a následné volání příslušné funkce je následující:

Kód v Matlabu

```
%% Interakce s režimem volného pohybu
disp ('1 - Zapnout režim volného pohybu, 2 - Vypnout režim
volného pohybu');
uloha_volneho_pohybu = input('Vyberte jednu z moznosti: ');

if uloha_volneho_pohybu == 1
nastav = ('(1)');
RezimVolnehoPohybu(Socket, nastav)
```

```
pause(0.5);
elseif uloha_volneho_pohybu == 2
nastav = ('(0)');
RezimVolnehoPohybu(Socket, nastav)
pause(0.5);
end
```

Uživatel je stejně jako v případě ovládání nástroje vyzván k zadání číselné hodnoty 1 nebo 2, čímž se hodnota uloží do proměnné *uloha_volneho_pohybu* a určí se tak, jestli bude požadavek pro aktivaci, nebo deaktivaci volného režimu. Má-li se daný mód aktivovat, je zapotřebí zadat hodnotu 1 a tím se tato hodnota uloží i do proměnné *nastav*, čímž se nastaví vstupní parametr volané funkce `RezimVolnehoPohybu`, která následně pošle požadavek robotovi. V opačném případě je proměnná *nastav* rovna 0 a vstupní parametr volané funkce `RezimVolnehoPohybu` je iniciovaný k deaktivaci příslušného módu.

2.5.7 Vypnutí robota a PolyScope

Další možností řídícího skriptu je možnost odeslání příkazu pro vzdálené vypnutí robota a řídícího programu PolyScope. Tento příkaz volá pomocnou funkci `VypnoutRobota`, která byla popsaná v předchozím oddílu o volacích funkcích Matlabu. Příkaz pro volání funkce ke vzdálenému vypnutí robota je jednoduchý a následující:

Kód v Matlabu

```
VypnoutRobota(Socket)
```

2.5.8 Kalibrace kamery

Cílem tohoto řešení je detekovat kulaté objekty a určit tak jejich příslušné translační souřadnice x a y, za účelem manipulace s těmito objekty. Princip řešení je založen na pořízení snímku z webové kamery (Logitech Webcam C930e) a následné detekci kulatých objektů v pořízeném snímku. Kód pro detekci kulatých objektů je následující:

Kód v Matlabu

```
clear; clc;
camera = webcam('Logitech Webcam C930e');
while true
%% Oříznutí snímku a změna pixelů
```

```
xs=200; ys=500; xl=800; yl=1400;
Picture = camera.snapshot;
Picture = Picture(xs: xl, ys:yl, :);
Picture = imresize(Picture, [485, 740]);
image(Picture)
hold on; grid;

%% Hledej kulaté objekty
[centers, radii] = imfindcircles(Picture, [9 20],
'ObjectPolarity', 'dark', 'Sensitivity', 0.9)
h = viscircles(centers, radii);
x = (centers(1:5)-390.1);
y = (-centers(6:10)-110);
disp(x)
disp(y)

%% Úprava souřadných os pro vizuální kontrolu výsledku
xo = (-390: 20: 350); yo = (-590: 20: -105);
osa_x = linspace(1, size (Picture, 2), numel(xo));
osa_y = linspace(1, size (Picture, 1), numel(yo));
set(gca, 'XTick', osa_x, 'XTickLabel', xo)
set(gca, 'YTick', osa_y, 'YTickLabel', flipud (yo (:)))
hold off
end
```

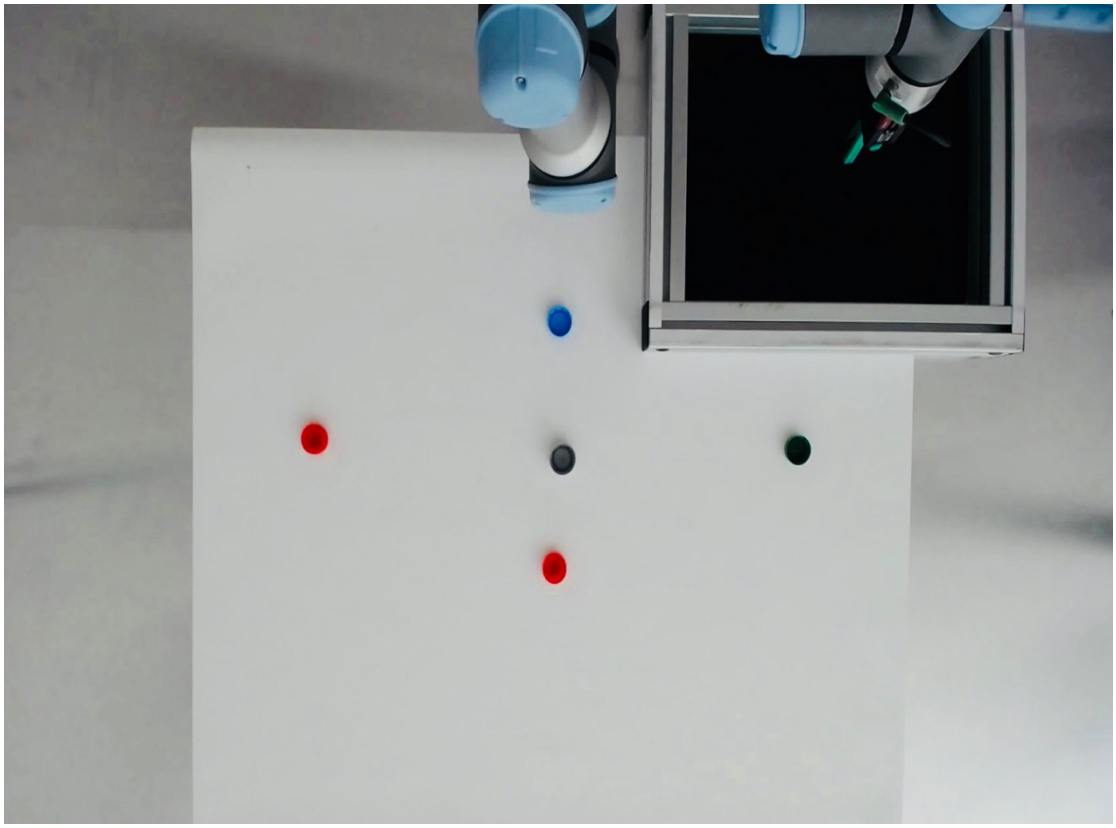
Do proměnné *camera* se definuje použitá kamera, která je v tomto případě (Logitech Webcam C930e) s rozlišením 1920×1080 pixelů. Následně je pomocí příkazu `camera.snapshot` pořízen snímek a je uložený do proměnné *Picture*. Jelikož je kamera umístěná nad robotem v takové vzdálenosti, aby nebránila při pohybu ramene, dochází k nasnímání velké plochy, což je nežádoucí. Řešení tohoto problému se nabízí jako fakt, že Matlab chápe snímek z kamery jako matici, a proto ji můžeme patřičně upravit a vymezit tak snímanou plochu. Za tímto účelem jsou zde proměnné *xs*, *xl*, *ys*, *yl*, které vymezují velikost nové matice neboli nově pořízeného snímku, viz kód. Dalším krokem je změna velikosti pixelů daného snímku a následně je snímek zobrazen na obrazovku.

Po zobrazení snímku je použit příkaz `imfindcircles`, který využívá princip „Houghovy transformace“ pro nalezení parametrického popisu jednoduchých objektů v obraze jako např. přímky, kružnice atd. Tento příkaz tedy pracuje s parametry, jako jsou citlivost snímání kružnice a má za úkol hledat kruhové objekty tmavší než pozadí. Výstupním parametrem této funkce je poloměr a střed každého nalezeného kulatého objektu, který slouží právě k nalezení souřadnic jednotlivých objektů, které jsou následně vypsány na obrazovku. Poté je ještě nutné přepočítat výsledné souřadnice získané ze snímku, k těm skutečným odpovídajícím pracovnímu prostoru robota. K tomuto účelu jsou zde proměnné *x* a *y*, které (viz kód) obsahují jednoduchý přepočet, abychom se alespoň dostali k co nejpřesnějšímu výsledku.

Jelikož ale není kamera dokonale vodorovně umístěná nad robotem a také zaostření kamery z programu Matlab není dokonalé, dochází ke zkreslení obrazu. U některých snímaných objektů dochází ke zkreslení, které se zvětšuje s rostoucí vzdáleností od čočky kamery. Jsou zde pro vizuální kontrolu ještě proměnné *xo* a *yo*, které slouží k úpravě souřadných os nového snímku za účelem přiblížení k reálným souřadnicím vymezeného pracovního prostoru robota, aniž by byla změněna velikost snímku.

2.6 ÚLOHA DETEKCE A PŘEMÍSTĚNÍ OBJEKTŮ

Jelikož se robotická ramena v průmyslu často využívají k přemísťování nejrůznějších dílů či materiálů, je zde ukázaná úloha založená na podobném principu. Úkolem této úlohy je najít, kde se předměty nachází a přemístit je pomocí robota do ohraničeného místa (viz obrázek 2.9). Veškerý pohyb robota v této úloze a jeho řízení se provádí pomocí již zmíněných a popsaných funkcí ze softwaru Matlab. V této jednoduché ukázkové úloze jsou využity téměř veškeré funkce, které byly podrobně popsány v oddílu 2.5 ukázkových aplikací skriptu v Matlabu. Jak je znázorněno na obrázku 2.9, objekty, se kterými bude robot pracovat a přemísťovat je z bodu A do bodu B jsou víčka od limonády, neboť mají vhodný kulatý tvar, jsou lehké a robot je může snadno a bez problémů uchopit a přemístit.



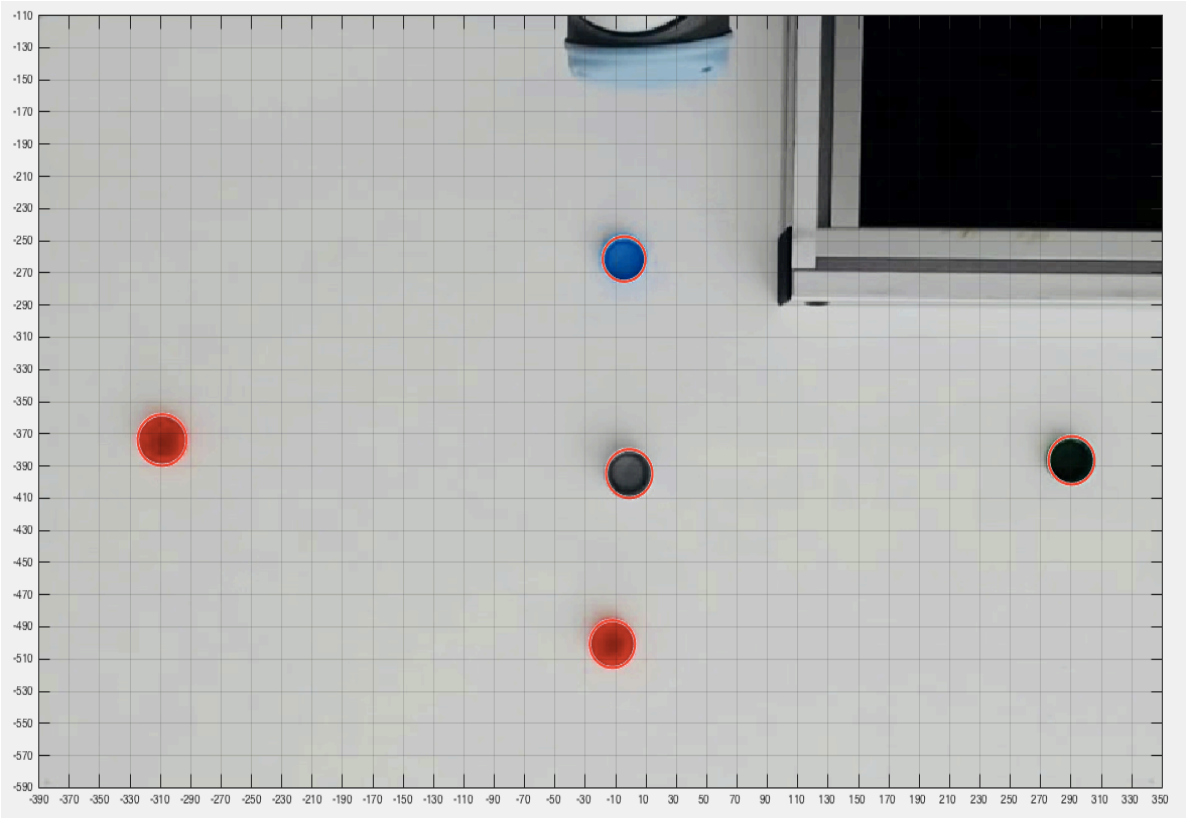
Obrázek 2.9 – Pracovní prostor robota s rozmístěnými objekty

2.6.1 Detekce

Prvním a nejdůležitějším úkolem, než bude možné s objekty manipulovat, je zjistit, kde se dané objekty, tedy víčka, nachází. Je tedy nutné zjistit příslušné translační souřadnice x a y jednotlivých víček. Pro detekci objektů a následné určení jejich translačních souřadnic je využito již zmíněné kamerové řešení pro detekci objektů, které je popsané v pododdílu 2.5.8. Toto řešení tedy hledá kulaté objekty, které následně zvýrazní tím, že je ohraničí červeně a vypíše souřadnice středu jednotlivých objektů.

Objekty použité v této úloze vyhovují nejenom hmotností, ale především tvarem, a proto je (viz obrázek 2.10) metoda pro detekci kulatých objektů vhodná a víčka bez problémů detekuje, zvýrazní a nalezne příslušné souřadnice. Obrázek 2.10 také ukazuje, že velikost pracovního prostoru robota, která je kamerou nasnímána za účelem detekce objektů, je pro souřadnou osu x od -390 mm do 350 mm a pro souřadnou osu y od -105 mm do -590 mm. Dále je zde také použita již zmíněná mřížka, která slouží jako vizuální kontrola nalezených a vypsaných souřadnic jednotlivých objektů, neboť přesnost řešení detekce víček pracuje u některých objektů s odchylkou až $\pm$ 5 mm. Hodnoty souřadnic nalezených objektů dle obrázku

2.10 jsou uvedeny v tabulce 2.2 a při srovnání obdrženého výsledku metodou detekce a vizuální kontrolou pomocí mřížky je vidět, že výsledky se pohybují dost přesně.



Obrázek 2.10 – Nalezené a zvýrazněné objekty

Tabulka 2.2 – Nalezené translační souřadnice objektů dle obrázku 2.10

Pozice objektu	*x*, mm	*y*, mm
Levé víčko	-306	-379
Pravé víčko	295	-374
Horní víčko	-6	-249
Dolní víčko	-8,2	-500
Prostřední víčko	-1	-392

2.6.2 Přesun objektů

Jakmile jsou zjištěné souřadnice jednotlivých objektů, je možné začít s přesouváním těchto objektů na požadované místo. Dalším krokem tedy je zvolit jednu z možností pohybu robota, které byly popsány v oddílu 2.5 o ukázkových aplikacích. Pro pohodlí je možné rovněž použít speciální režim volného pohybu a dostat se nad objekt ručně, což zaručí velmi přesnou

manipulaci s objektem. Tuto volbu je vhodné použít za účelem kontroly souřadnic polohy jednotlivých objektů, nebo např. ke zjištění polohy objektů, které by kamera nedetekovala.

Jelikož ale máme k dispozici souřadnice (viz tabulka 2.2) jednotlivých objektů, pak se také jako vhodná metoda nabízí možnost přesunout robota s využitím translačních pohybových možností z ukázkových aplikací. Jak je z obrázku 2.10 zřejmé, všechny objekty jsou na podložce umístěny tak, že je lze zvednout a přesunout bez nutnosti natočení nástroje robota. Není zde tedy nutné rotovat nástroj robota, a to naši úlohu značně zjednodušuje, neboť se používá jenom translační pohyb, který, jak z teoretické části práce plyne, je mnohonásobně jednodušší. Část kódu pohybové úlohy pro prostřední víčko dle obrázku 2.10, je následující:

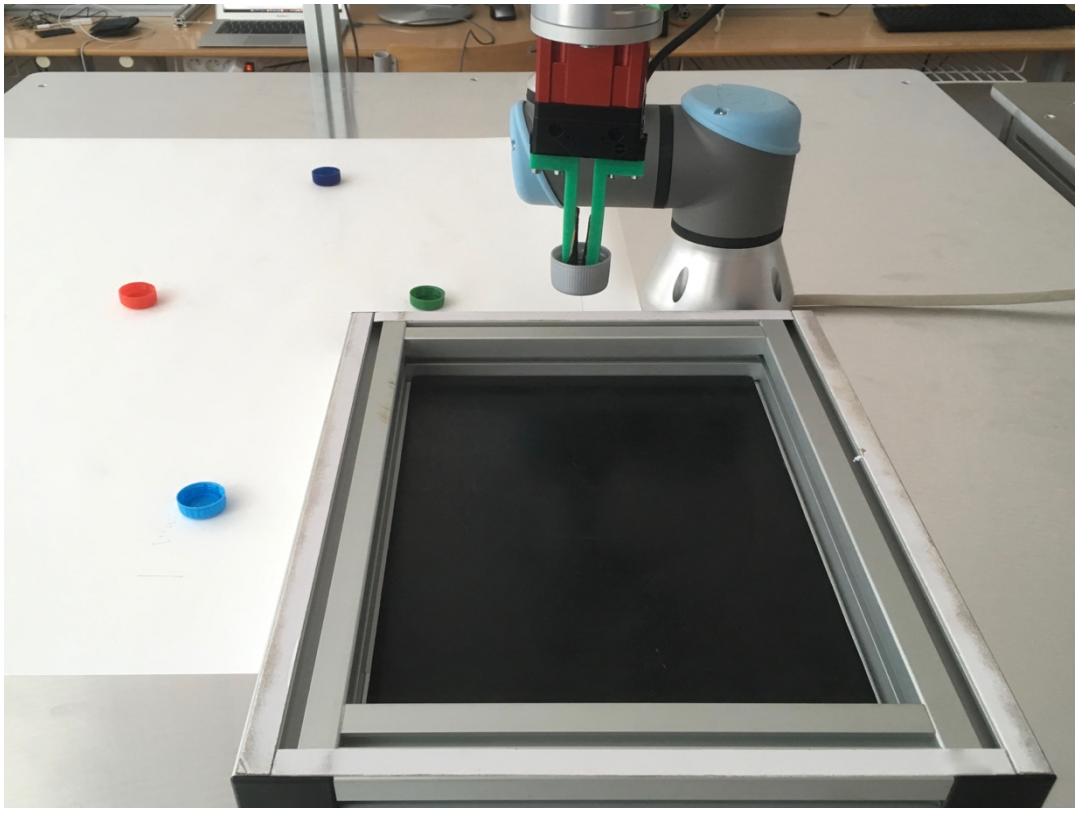
Kód v Matlabu

```
clc; clear;
%% TCP/IP spojení
disp ('Vítejte v programu pro vzdálené ovládaní robotického
ramene UR3, Prosím počkejte na spojeni s robotem!');
Socket = tcpip ('0.0.0.0', 30000, 'NetworkRole', 'server');
Fclose(Socket);
disp ('Prosím stlačte tlačítko play v PolyScope')
fopen (Socket);
disp ('Spojeni s robotem je navázáno');

%% Zjištění aktuální pozice TCP a přesun prostředního víčka
nastav = ('(1)');
NastrojRobota(Socket, nastav)
Pozice_TCP = SoradniceTcp(Socket);
Nova_rotace = Pozice_TCP (4:6);
Nova_translace = [-1 (-392) 100];
PohybRobota(Socket, Nova_translace, Nova_rotace)
Pause(0.5);
Nova_translace = [-1 (-392) 2.5];
PohybRobota(Socket, Nova_translace, Nova_rotace)
nastav = ('(0)');
NastrojRobota(Socket, nastav)
pause(1);
Nova_translace = [306 (-92) 150];
```

```
PohybRobota(Socket, Nova_translace, Nova_rotace)
pause(0.5);
Nova_translace = [306 (-92) 15];
PohybRobota (Socket, Nova_translace, Nova_rotace)
%% Sepnutí nástroje za účelem puštění objektu
nastav = ('(1)');
NastrojRobota(Socket, nastav)
pause(0.5);
Nova_translace = [306 (-92) 150];
PohybRobota(Socket, Nova_translace, Nova_rotace)
pause(0.5);
```

Nejdříve tedy dojde ke spojení s ramenem pomocí TCP/IP spojení stejně, jak tomu bylo při popisu ukázkových aplikací v předchozím oddílu 2.5. Rovněž struktura kódu a použité funkce jsou stejné jako při ukázkových úlohách a průběh přesunu objektu znázorňuje obrázek 2.11, kde je zachycený moment, kdy robot přesouvá prostřední šedé víčko na určenou pozici. Zde uvedený kód je jenom část celkového kódu navrženého pro tuto úlohu, určený k přesunu jednoho objektu.



Obrázek 2.11 – Přesun objektu

Pro přesun zbylých objektů, tedy víček je kód pořád stejný a opakuje se pro každé víčko. Jediné, co se v kódu pořád mění jsou příslušné souřadnice každého objektu a souřadnice pro finální pozici těchto objektů. Po dokončení úlohy jsou všechny objekty přesunuté a srovnané na požadované místo a výsledek znázorňuje obrázek 2.12.

Obrázek 2.12 – Finální pozice objektů

3 ZÁVĚR

Bakalářská práce je věnována problematice robotických ramen a tvorbě softwarové komunikace, která probíhá mezi programem Matlab a řídícím programem robotického ramene UR3.

V teoretické části práce je jednoduché rozdělení robotů, ze kterého vyplívá, že kooperující robotická ramena mají jednoznačnou převahu nad rameny klasickými, a to je důvod, proč jsou v dnešní době tolik žádanými pomocníky v každé oblasti průmyslu. Dále jsou zde uvedeny jednotlivé druhy pohybů a také pohyb robota v prostoru, a to jak pro část translační, tak i část rotační.

Praktická část práce obsahuje podrobný popis robotického ramene UR3 a také programu PolyScope, se kterým probíhá softwarová komunikace s programem Matlab. Řídicí program robota PolyScope je opravdu všestranný nástroj, který umožnuje řídit robotické rameno a jeho pohyby precizně a plynule. Navržené řešení softwarové komunikace umožnuje řídit robota tak, že mu posíláme jednotlivé příkazy. Tento způsob řízení je méně efektivní z pohledu plynulosti výsledného pohybu, ale na druhou stranu se tím otevírá možnost ovládání ramene na dálku z programu Matlab, čímž máme k dispozici nepřeberné množství dalších funkcionalit včetně moderních nástrojů strojového učení a počítačového vidění. Funkčnost řešení softwarové komunikace mezi programy byla postupně ověřována pro jednotlivé funkce a také jako celek v ukázkových aplikacích, kde se ukázala jako bezproblémová.

Předmětem dalšího bádání a práce na softwarové komunikaci za účelem zdokonalení dosavadního řešení by tedy mohla být lepší a efektivnější metoda pro rotaci koncového efektoru robota, založená například na kvaternionech. Dále by bylo možné přidávat další významné funkce robota, jako např. volbu jiného typu pohybu, nebo zdokonalit kamerovou detekci objektů za účelem získání souřadnic manipulujících objektů a rozšířit možnost snímání i na jiný druh objektů.

LITERATURA

ABB YuMi® - IRB 14000. In: ABB [online]. Praha 4: ABB, ©2018 [cit. 2018-02-27]. Dostupné z: http://new.abb.com/products/robotics/cs/prumyslove-roboty/yumi

Collaborative robotics: KUKA LBR IIWA. In: KUKA [online]. Augsburg: KUKA Aktiengesellschaft, ©2018 [cit. 2018-02-27]. Dostupné z: https://www.kuka.com/cs-cz/tisk/news/2016/03/collaborative-robotics/

Collaborative Robot: FANUC Robot CR-35iA. FANUC [online]. FANUC CORPORATION, ©2011-2017 [cit. 2018-04-07]. Dostupné z: https://www.fanuc.co.jp/en/product/robot/f_r_collabo.html

CR-35iA. In: FANUC [online]. Praha 8 - Libeň: FANUC Czech, 2018 [cit. 2018-02-27]. Dostupné z: http://www.fanuc.eu/cz/cs/roboty/str%C3%A1nka-filtru-robot%C5%AF/spolupracuj%C3%ADc%C3%AD-roboty/collaborative-cr35ia

DUŠEK, F.; HONC D. Matlab a Simulink: úvod do používání. Pardubice: Univerzita Pardubice, 2005. ISBN 80-719-4776-8.

Fixed perimeter guarding with weld curtains. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001, 9 July 2008 [cit. 2018-03-07]. Dostupné z: https://en.wikipedia.org/wiki/Machine_guarding#/media/File:Fixed_Perimeter_Guarding.jpg

GUPTA, A. K.; ARORA S. K.; WESTCOTT J. R. Industrial Automation and Robotics [online]. Dulles, Virginia: Mercury Learning and Information, 2017 [cit. 2018-02-01]. ISBN 978-1-5231-0916-6. Dostupné z: https://app.knovel.com/hotlink/toc/id:kpIAR00001/industrial-automation/industrial-automation

IRB 1520ID. In: ABB [online]. Praha 4: ABB, ©2018 [cit. 2018-03-02]. Dostupné z: http://www.abb.com/cawp/seitp202/c1c09378f9da858dc1257a2f00401904.aspx

IRB 360 FlexPicker™: Robot FlexPicker® druhé generace pracuje s ještě vyšší produktivitou. In: ABB [online]. Praha4: ABB, ©2018 [cit. 2018-03-07]. Dostupné z: http://new.abb.com/products/robotics/cs/prumyslove-roboty/irb-360

JEFFUS, L. Welding and metal fabrication [online]. Clifton Park, NY: Delmar, 2012, 597–603 [cit. 2018-02-01]. ISBN 9781621986904. Dostupné z: https://app.knovel.com/hotlink/toc/id:kpWMFE0002/welding-metal-fabrication/welding-metal-fabrication

KUKA LBR iiwa. KUKA: SÍLA AUTOMATIZACE [online]. Augsburg: KUKA Aktiengesellschaft, ©2018 [cit. 2018-03-27]. Dostupné z: https://www.kuka.com/cs-cz/produkty,-služby/robotické-systémy/průmyslové-roboty/lbr%C2%A0iiwa

Kuka Robotics. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001, 2 July 2015 [cit. 2018-03-20]. Dostupné z: https://commons.wikimedia.org/wiki/File:Innorobo_2015_-_Kuka_Robotics.JPG

LONESCU, H. 6 degrees of freedom. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001, 6 June 2010 [cit. 2018-03-07]. Dostupné z: https://commons.wikimedia.org/wiki/File:6DOF_en.jpg

MOSTÝN, V.; KRYS V. Mechatronika průmyslových robotů: učební text [online]. Ostrava: Vysoká škola báňská – Technická univerzita Ostrava, 2012 [cit. 2018-03-07]. ISBN 978-80-248-2610-3. Dostupné z: http://www.person.vsb.cz/archivcd/FS/MePR/texty/Mechatronika%20prumyslovych%20robotu.pdf

NOF, S. Y. Handbook of industrial robotics. New York: Wiley, 1985. ISBN 04-718-9684-5.

Programmable Universal Machine for Assembly. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2018-03-07]. Dostupné z: https://en.wikipedia.org/wiki/Programmable_Universal_Machine_for_Assembly

SKAŘUPA, J. Průmyslové roboty a manipulátory [online], 2007. Ostrava: Vysoká škola báňská – Technická univerzita [cit. 2018-02-05]. ISBN 978-80-248-1522-0. Dostupné z: http://www.elearn.vsb.cz/archivcd/FS/PRM/Text/Skripta_PRaM.pdf

ŠVEJDA, M. Kinematika robotických architektur [online]. Plzeň, 2011 [cit. 2017-11-02]. Dostupné z: http://home.zcu.cz/~msvejda/URM/materialy/KinematikaRobotArchitektur.pdf. Práce ke státní doktorské zkoušce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky.

TAUFER, I.; KOTYK J.; JAVŮREK M. Jak psát a obhajovat závěrečnou práci: bakalářskou, diplomovou, rigorózní, disertační, habilitační. 2., dopl. a opr. vyd. Pardubice: Univerzita Pardubice, 2014. ISBN 978-80-7395-746-9.

UNIVERSAL ROBOTS: Uživatelská příručka – UR3/CB3 Překlad originálního návodu. Verze 3.5.2. 2018. Dostupné také z: https://s3-eu-west-1.amazonaws.com/ur-support-site/32331/UR3_User_Manual_cs_Global-3.5.2.pdf

Universal Robots. In: ATN [online]. Oppach: ATN HÖLZEL, ©2018 [cit. 2018-03-07]. Dostupné z: http://atngmbh.com/en/universal-robots/

VOJÁČEK, A. Robot vs. Cobot. Automatizace.hw.cz: rady a poslední novinky z oboru [online]. Praha 4: HW server, c1997-2014, 08. září 2017 [cit. 2018-02-24]. Dostupné z: https://automatizace.hw.cz/robot-vs-cobot.html

Yaw Axis Corrected. In: Wikipedia: the free encyclopedia [online]. san Francisco (CA): wikimedia Foundation, 2001, 10 February 2010 [cit. 2018-03-20]. Dostupné z: https://commons.wikimedia.org/wiki/File:Yaw_Axis_Corrected.svg

YuMi® – společně tvoříme budoucnost automatizace. ABB [online]. Praha 4: ABB, ©2015 [cit. 2018-02-27]. Dostupné z: https://library.e.abb.com/public/5662b754739545899518eea9ea7e3781/yumi_datasheet_cz.pdf

PŘÍLOHY

A – CD

Příloha A

Příloha k bakalářské práci

SW PRO OVLÁDÁNÍ ROBOTICKÉHO RAMENE Z PROSTŘEDÍ MATLAB

Dominik Varga

CD

Obsah