UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

MONTÁŽNÍ OPERACE S ŠROUBOVÝM SPOJENÍM S VYUŽITÍM ROBOTA ABB YUMI

Bc. Petr Čepelák

Diplomová práce

2020

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Petr Čepelák**
Osobní číslo:	**I18228**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Řízení procesů**
Téma práce:	**Montážní operace s šroubovým spojením s využitím robota ABB YuMi**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování

Cílem práce je realizovat montážní operaci s využitím robota ABB YuMi, kde dojde k šroubovému spojení několika dílů. Operace bude zahrnovat podání dílů a spojovacích prvků, spojení do celku a odložení výrobku. K tomu bude využito obou paží robota a mechanických přípravků, které budou k tomu účelu navrženy a sestrojeny. V rámci práce bude ověřena i možnost automatické lokalizace spojovacích prvků s využitím modulu strojového vidění robota ABB YuMi. Pro off-line programování a odladění programu robota bude využito software ABB RobotStudio.

Práce bude obsahovat rozbor jednotlivých kroků řešení a návrh přípravků potřebných pro jejich realizaci. Dále bude v přiměřeném rozsahu popsáno využité zařízení a použité programovací prostředky a softwarové nástroje. Bude popsána struktura vytvořeného programu a komentovány jednotlivé procedury. Funkčnost programu bude v textu demonstrována vhodnou formou.

Rozsah pracovní zprávy: **min. 60 stran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

SICILIANO, B., SCIAVICCO, L., ORIOLLO, G. Robotics: Modelling, Planning and Control. Springer – Verlag, 2009.
CVEJN, J. Návod ke zpracování praktické úlohy ke zkoušce z předmětu Modelování, plánování pohybu a řízení robotů. Univerzita Pardubice, FEI, 2019.
Návod k použití IRC5 s jednotkou FlexPendant, RobotWare 6.06. ABB, 2017.
Technical reference manual RAPID overview, RobotWare 6.06. ABB, 2004-2017.
Technical reference manual RAPID Instructions, Functions and Data types, RobotWare 6.06. ABB, 2004-2017.
Product manual IRB 14000 gripper, IRC5. ABB, 2015-2017.
Application manual Integrated Vision, RobotWare 6.06. ABB, 2013-2017.

Vedoucí diplomové práce: **doc. Ing. Jan Cvejn, Ph.D.**
Katedra řízení procesů

Datum zadání diplomové práce: **7. listopadu 2019**
Termín odevzdání diplomové práce: **15. května 2020**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 7. listopadu 2019

Prohlášení

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 02. 05. 2020

Bc. Petr Čepelák

Poděkování

Tímto bych chtěl poděkovat doc. Ing. Janu Cvejnovi, Ph.D., vedoucímu diplomové práce, za vstřícný přístup a cenné rady při zpracování. Poděkovat bych chtěl také svému kamarádovi za zapůjčení 3D tiskárny a své rodině za neustálou podporu a motivaci.

V Pardubicích dne 02. 05. 2020

Bc. Petr Čepelák

ANOTACE

Diplomová práce se zabývá realizací robotické aplikace. Jde o montážní operaci, jenž zahrnuje podání dílů a spojovacího materiálu, šroubové spojení dílů do jednoho celku a odložení kompletu. To s využitím dvouramenného kooperujícího robota ABB YuMi a software ABB RobotStudio. Řešení zahrnuje i návrh mechanických přípravků potřebných pro realizaci jednotlivých operací.

KLÍČOVÁ SLOVA

robotika, průmyslový robot, kooperující robot, programování robotů, jazyk RAPID.

TITLE

ASSEMBLY OPERATION WITH SCREWED CONNECTION BY USING THE ABB YUMI ROBOT

ANNOTATION

The diploma thesis deals with a realization of a robotic application. It is an assembly operation that involves moving parts and fasteners, screw connection of parts, and putting the product away. For this purpose the cooperative robot ABB YuMi and the software ABB RobotStudio were used. The solution includes the design of mechanical jigs that are needed for realization of particular operations.

KEYWORDS

Robotics, industrial robot, cooperative robot, robot programming, language RAPID.

OBSAH

SEZNAM ZKRATEK A ZNAČEK

3D	trojrozměrný
ABB	Asea Brown Boveri
CR	kooperující robot
IIWA	inteligentní průmyslový pracovní asistent
KMR	KUKA mobile robotics
LBR	označení pro lehkou konstrukci robota společnosti KUKA
R.U.R.	Rossumovi univerzální roboti
SW	software
TCP	středový bod nástroje
TX2	označení řady kooperujících robotů společnosti Stäubli
UR	označení řady kooperujících robotů společnosti Universal Robots

SEZNAM SYMBOLŮ PROMĚNNÝCH VELIČIN A FUNKCÍ

x složka vektoru polohy v ose x

y složka vektoru polohy v ose y

z složka vektoru polohy v ose z

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

ÚVOD

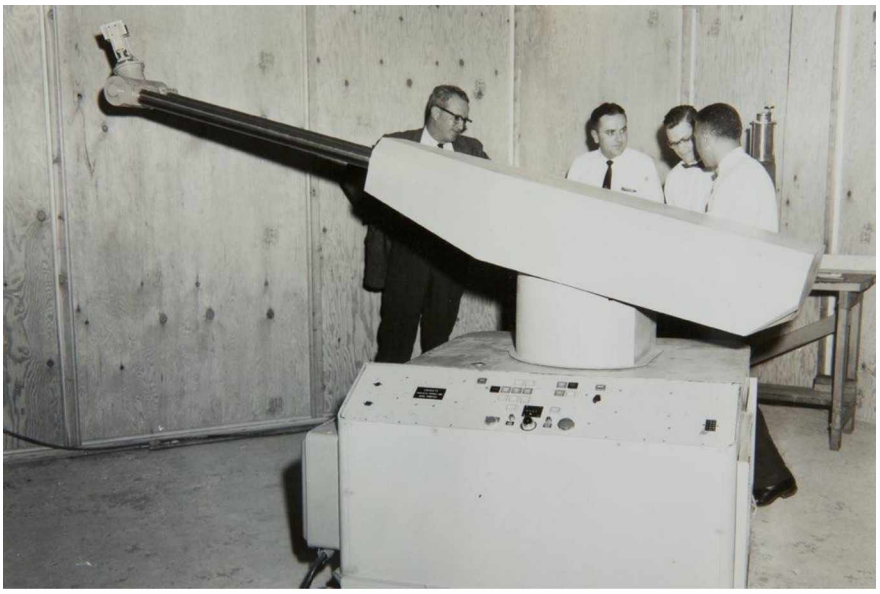
Hlavním cílem diplomové práce je realizace montážní operace, která bude zahrnovat podání dílů a spojovacího materiálu, šroubové spojení dílů do jednoho celku a odložení zkompletovaného výrobku. Je rovněž diskutována možnost využití strojového vidění pro lokalizaci spojovacího materiálu. To vše s využitím dvouramenného kooperujícího robota YuMi od společnosti ABB. Součástí řešení je i návrh a výroba mechanických přípravků potřebných pro montážní operaci. Pro tvorbu aplikace je využito software ABB RobotStudio.

Úvodní část práce je věnována nastínění aktuální situace ohledně robotizace v průmyslu. Následuje popis vlastností kooperujících robotů a přehled jejich běžně používaných zástupců. Další část se zabývá problematikou spojenou s programováním robotů a použitím programovacího jazyka RAPID. Obsahem textu je také rozbor jednotlivých kroků řešení zadané úlohy. V rámci toho jsou mimo jiné popsány i navržené mechanické přípravky a struktura vytvořeného programu, včetně komentovaného kódu jednotlivých procedur.

1 ROBOTIZACE V PRŮMYSLU

Slovo „robot“ poprvé použil spisovatel Karel Čapek ve své divadelní hře R.U.R. z roku 1920. Tento tvar slova mu poradil jeho bratr Josef, který jej odvodil ze slova „robota“. Zde roboty zastávají funkci umělých dělníků vykonávajících namáhavou činnost. V dnešním světě si pod tímto pojmem lze představit opravdu mnoho. Od kuchyňských a domácích spotřebičů, až po složitá zařízení, která jsou využívána v armádě, zdravotnictví, vesmírné technice atp. Roboty jsou také základním stavebním kamenem automatizace výrobních procesů. Robot využívaný pro podobné účely je v odborné praxi nazýván jako průmyslový robot, či manipulátor.

Již v roce 1954 začali američtí inženýři Goerge Devol a Joseph Engelberger pracovat na vývoji prvního průmyslového robota s názvem Unimate, jenž je ukázán na obrázku 1.1. Hned o pár let později byl Unimate nasazen do reálné výroby (Žáček, 2014).



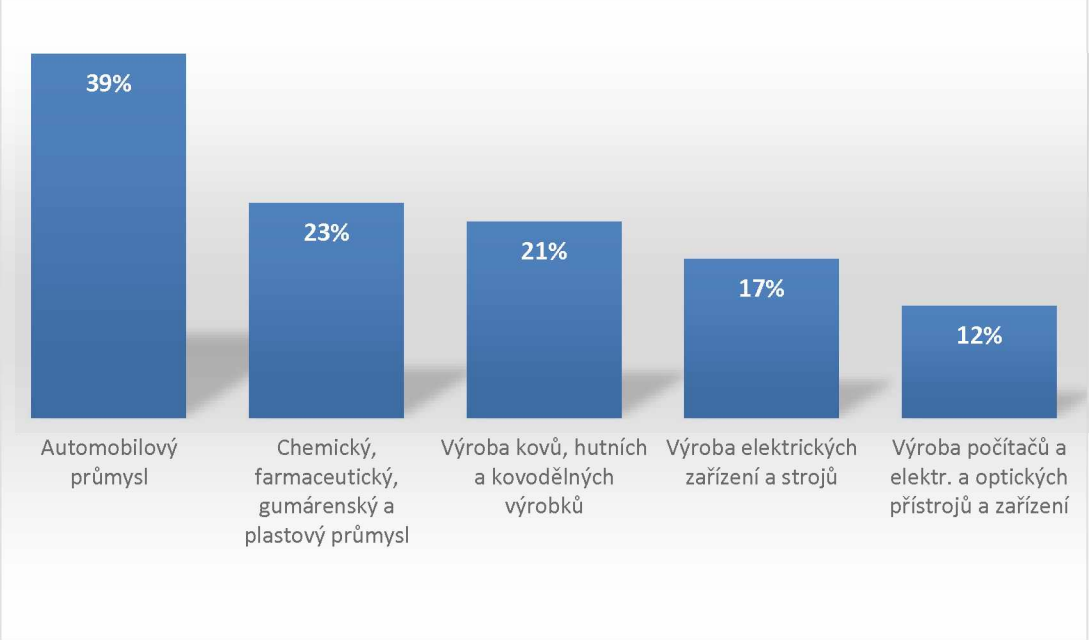
Obrázek 1.1 – První průmyslový robot Unimate (History, 2018)

Průmyslový robot je konstruován pro nepřetržité vykonávání zadané práce. Dokáže pracovat jako samostatná jednotka na základě vytvořeného programu, bez ohledu na zaměstnance, kteří okolo něj pracují. Z tohoto důvodu jsou roboty umístěny do bezpečnostních klecí. Vstup do pracovního prostoru robota je ošetřen řadou bezpečnostních prvků. Při jakémkoli narušení pracovního prostoru robota, např. vstupem obsluhy, je pohyb robotického ramene okamžitě zastaven. Při běžném provozu se tedy předpokládá, že robot

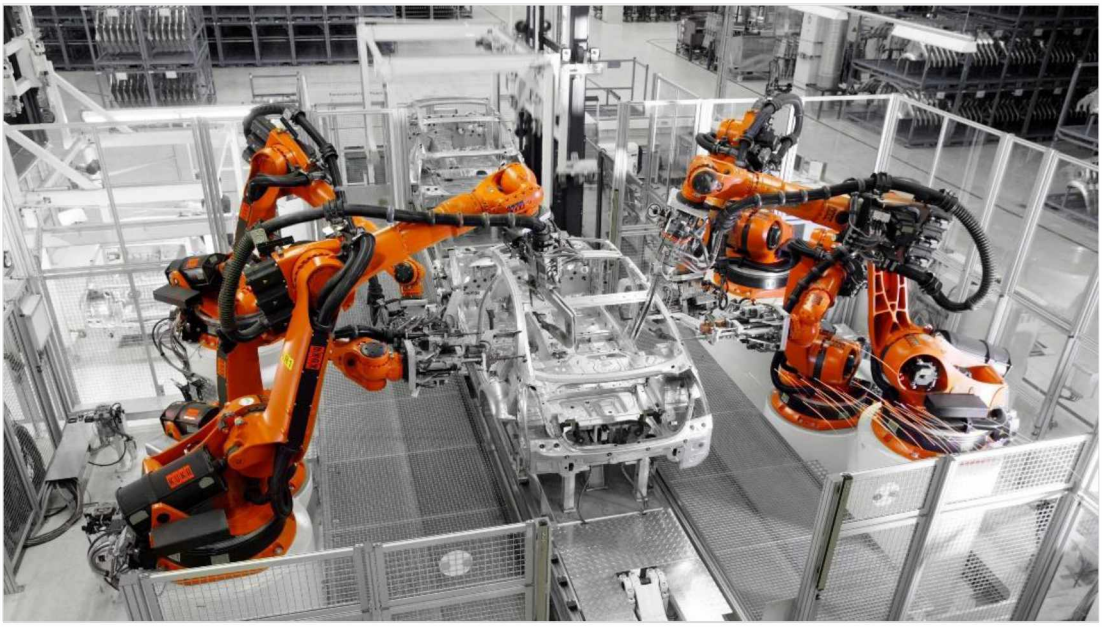
svým pohybem neohrožuje bezpečnost okolních zaměstnanců, proto mu je umožněno pracovat ve vysokých rychlostech, s vysokým užitečným zatížením a velkým dosahem.

Právě ve spojení s průmyslovými roboty je definován pojem robotizace. Jde o proces implementace průmyslových robotů do výroby, a to například pro nahrazení pracovníků, kteří zde vykonávali manuální činnost. Robotizace je dnes významnou oblastí pro zvýšení produktivity, efektivity a kvality výroby. Z důvodu udržení konkurenceschopnosti musí podniky do robotizace stále více a více investovat. Kromě toho jsou investice poháněny nedostatkem pracovní síly a rychle rostoucími náklady na mzdy zaměstnanců. Zároveň jsou pracovníci zbaveni monotónní, mnohdy také těžké, zdraví a životu nebezpečné práce.

Podle českého statistického úřadu je u nás využití průmyslové robotiky typické zejména pro velké podniky s více než 250 zaměstnanci. Přibližně 32 % takových podniků roboty používá. V grafu na obrázku 1.2 jsou uvedeny oblasti využití průmyslových robotů na území České republiky. Mezi ty nejběžnější patří automobilový průmysl. Příklad robotického pracoviště na výrobní lince automobilů je uveden na obrázku 1.3. Průmyslový robot je zde využíván pro šroubování, sváření, lepení, manipulaci s předměty atp. Roboty také využívá téměř čtvrtina firem působících v chemickém, farmaceutickém, gumárenském a plastovém průmyslu. Na třetí pozici se nachází firmy zabývající se výrobou kovů. Uvedená data se vztahují k roku 2018 (Využití robotiky, 2019).



Obrázek 1.2 – Využití průmyslových robotů na území České republiky (Využití robotiky, 2019).



Obrázek 1.3 – Výrobní linka automobilů s průmyslovými roboty

Možnosti robotizace se neustále rozšiřují, proto lze předpokládat, že zanedlouho budou roboty patřit mezi standardní vybavení každého výrobního podniku. Lidé se ale nemusí obávat, že by kvůli robotům přišli o svá zaměstnání. Příkladem mohou být světoví lídři robotizace, jako je Singapur, Jižní Korea nebo Německo, kteří mají současně i minimální nezaměstnanost. Společnosti se díky robotizaci stávají více konkurenceschopní, získávají více zakázek a rostou. Souběžně s tím vznikají i nová pracovní místa, již nyní je například velký nedostatek techniků a programátorů (Kozlík, 2019).

Stále častěji se v průmyslu začíná mluvit o nástupu tzv. kooperujících robotů, které jsou schopny bezpečně spolupracovat s lidskými pracovníky. I přesto, že jde o relativně nový pojem v oblasti automatizace, tak během posledních několika málo let mnoho světových robotických společností postupně zařazuje tento typ robotů do své nabídky, a to za účelem získání co největšího podílu na trhu při očekávaném růstu. Kooperující roboty jsou aktuálně nejrychleji rostoucím odvětvím automatizace a začínají se řadit mezi běžné technologie. V roce 2017 byl podíl kooperujících robotů pouhé 4 % z celkového prodeje průmyslových robotů, ale podle předběžných odhadů se do roku 2025 očekává nárůst až na 34 %. Mezi hlavní důvody patří stále se snižující cena kooperujících robotů, jejich snadnější implementace do výroby a širší možnosti využití (Robotics Online Marketing Team, 2018).

2 KOOPERUJÍCÍ ROBOT

Kooperující robot, někdy také označován jako „cobot“, je speciální druh průmyslového robota. Jde o robota, který je navržen tak, aby byl schopný spolupracovat v přímém kontaktu s člověkem, a to za účelem vyšší kvality, přesnosti a produktivity odvedené práce. Stejně jako ten průmyslový může pracovat bez přestávky a prakticky do nekonečna opakovat požadované úkony, které by jinak plýtvaly lidským potenciálem – ten může být následně využit pro kreativnější práci (Duchoslav, 2017).

Koncept robota tohoto typu se zrodil kolem roku 1995, kdy byl součástí výzkumného projektu vedeného nadací General Motors Foundation. Cílem projektu bylo vytvořit právě takové roboty, které by byly schopny pracovat ruku v ruce s obsluhou (Vojáček, 2017).

Kooperující robot, na rozdíl od toho průmyslového, nemusí být při vykonávání svých úkonů oplocen. Naopak je navržen tak, aby byl bezpečný i při práci v těsné blízkosti člověka. Je vybaven řadou dalších bezpečnostních prvků, jako např. citlivými senzory, které v případě neočekáváné kolize s obsluhou zastaví pohyb ramene v řádech milisekund. Dalším zásadním rozdílem je podstatně nižší náročnost na programování. Klasický průmyslový robot vyžaduje pokročilé znalosti a zkušenosti v oblasti programování. Při programování kooperujícího robota lze například využít tzv. funkce učení. Díky tomu může být rameno robota ručně vedeno po požadované trajektorii. Robot si tuto sekvenci bodů a pohybů zapamatuje a následně je schopen ji samostatně vykonávat. Pomocí tohoto postupu dokáže změny v programu provádět i proškolená obsluha. Odpadá tak nutnost přítomnosti zkušeného programátora při každé drobné změně robotického pracoviště.

Konstrukce je vyrobena z lehkého a flexibilního materiálu. Obvykle se sice pohybuje pomaleji a má nižší nosnost než klasický průmyslový robot, ale to z něho dělá ideálního pomocníka při práci s materiály o menších rozměrech a nižší hmotností. Mezi příklady praktického využití patří šroubování, podávání předmětů, svařování, lepení, balení křehkých materiálů, míchání barev apod. Na obrázku 2.1 je uveden příklad kooperujícího robota od společnosti Universal Robots (Duchoslav, 2017).

Součástí popisu kooperujících robotů je i přehled jejich používaných zástupců. Pro tento záměr byl vybrán sortiment společností Fanuc, Stäubli, Universal Robots, Kuka a ABB. Jde o světově známé společnosti, které se i u nás řadí mezi ty nejběžnější. Jejich nabídka sice široce přesahuje odvětví kooperujících robotů, nicméně pro účely této práce je soustředěno právě na ně.

Obrázek 2.1 – Kooperující robot (Vojáček, 2017)

2.1 KOOPERUJÍCÍ ROBOTY SPOLEČNOSTI FANUC

Japonská společnosti Fanuc nabízí řadu kooperujících robotů s označením CR, jako „Collaborative Robot“. Jde o šestiosá ramena různých rozměrů. Mezi ty nejzajímavější patří robot s označením CR-35iA. Jde o kooperující rameno s hodnotou užitečného zatížení až 35 kg. Díky tomu je považován za nejsilnějšího kooperujícího robota na světě. Rameno disponuje dosahem až 1813 mm s přesností ±0,03 mm. To vše za cenu vysoké hmotnosti ±990 kg. Díky svým vlastnostem je v průmyslu nasazován pro manipulaci s těžkými předměty. Příkladem použití může být manipulace s pneumatikami automobilů ve výrobním závodě, kde robot uchytí pneumatiku a obsluha jej pak volně navádí na požadované místo. Tato činnost je znázorněna na obrázku 2.2 (Collaborative Robot, 2011-2019).

Obrázek 2.2 – Kooperující robot CR-35iA od společnosti Fanuc (Teaching a robot using hand guidance, 2019)

2.2 KOOPERUJÍCÍ ROBOTY SPOLEČNOSTI STÄUBLI

Řada kooperujících robotů TX2, od švýcarské společnosti Stäubli, nabízí šest typů robotických ramen. Jedno z nich je uvedeno na obrázku 2.3. Ve všech případech jde o šestiosá robotická ramena, která se od sebe liší především svými rozměry. S rozměry ramen se mění i jejich základní parametry, mezi které patří přesnost, hmotnost, nosnost a velikost pracovního prostoru. Konkrétní hodnoty zmíněných parametrů pro jednotlivé modely jsou uvedeny v tabulce 2.1. Stäubli se mimo jiné soustředí i na možnost použití svých kooperujících ramen v hygienicky citlivých prostředí, jako je potravinářství, medicína a farmacie, fotovoltaika apod. Dále také pro náročné činnosti, například obrábění (Řada robotů TX2, 2019).

Tabulka 2.1 – Vlastnosti robotů řady TX2 od společnosti Stäubli (Řada robotů TX2, 2019)

Model	TX2-40	TX2-60	TX2-60L	TX2-90	TX2-90L	TX2-90XL
Zatížení, kg	1,7	3,5	2	6	5	4
Dosah, mm	515	670	920	1000	1200	1450
Počet os	6	6	6	6	6	6
Přesnost, mm	±0,02	±0,02	±0,03	±0,03	±0,035	±0,04
Hmotnost, kg	29	52,2	52,5	114	117	119

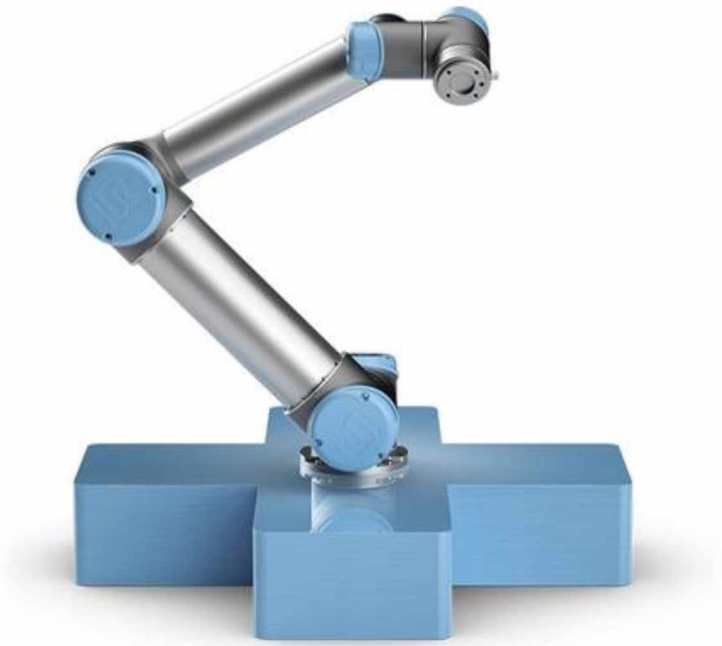
Obrázek 2.3 – Kooperující robot řady TX2 od společnosti Stäubli (Řada robotů TX2, 2019)

2.3 KOOPERUJÍCÍ ROBOTY SPOLEČNOSTI UNIVERSAL ROBOTS

Dánská společnost Universal Robots se jako jedna z mála soustředí výhradně na vývoj kooperujících robotů. Nabízí šestiosé roboty ve čtyřech různých provedeních, konkrétně UR3, ukázaný na obrázku 2.4, dále UR6, UR10 a UR16. Zkratka UR vystihuje název společnosti a číselná hodnota odpovídá velikosti manipulačního zatížení, tedy 3, 6, 10 a 16 kg. Při porovnání s ostatními výrobci jsou roboty této řady mimo jiné zajímavé poměrem hmotnosti a nosnosti jednotlivých ramen. Například nejmenší člen UR3, se zatížením až 3 kg, váží pouhých 11 kg. Zároveň umožňuje nekonečnou rotaci s koncovým nástrojem, což lze využít například u šroubovacích operací. Přehled základních vlastností všech typů je uveden v tabulce 2.2. Díky nízké váze se nabízí hned několik možností umístění robotů, jako například na stěnu či strop. Další z možností může být i implementace přímo na pracovní stůl zaměstnance (Kolaborativní roboti společnosti Universal Robots, 2020).

Tabulka 2.2 – Vlastnosti kooperujících robotů od společnosti Universal Robots (Kolaborativní roboti společnosti Universal Robots, 2020)

Model	UR3	UR5	UR10	UR16
Manipulační zatížení, kg	3	5	10	16
Hmotnost, kg	11,2	20,6	33,5	33,1
Dosah, mm	500	850	1300	900



Obrázek 2.4 – Kooperující robot UR3 od společnosti Universal Robots (UNIVERSAL ROBOT UR3e, 2020)

2.4 KOOPERUJÍCÍ ROBOTY SPOLEČNOSTI KUKA

Robot LBR iiwa, od německé společnosti Kuka, je jeden z prvních sériově vyráběných kooperujících robotů. Zkratka „LBR“ je označením pro lehkou konstrukci a „iiwa“ znamená „intelligent industrial work assistant“, což v doslovném překladu znamená „inteligentní průmyslový pracovní asistent“. LBR iiwa existuje ve dvou provedeních, které se od sebe liší svou nosností. Typ LBR iiwa 7 R800, uvedený na obrázku 2.5, disponuje nosností 7 kg s maximálním dosahem 800 mm. Druhým typem je LBR iiwa 14 R820, jenž umožňuje zátěž až 14 kg s maximálním dosahem 820 mm. Obě tyto ramena mohou být nainstalována na podlahu, stěnu či strop. Mezi vhodné oblasti použití patří například paletizace, montáž a demontáž, manipulace s předměty, nanášení lepidla, měření apod. (LBR iiwa, 2019).

Obrázek 2.5 – Kooperující robot LBR iiwa 7 R800 od společnosti Kuka (Collaborative robotics, 2016)

Dalším kooperujícím robotem této společnosti je KMR iiwa, kde „KMR“ znamená „KUKA mobile robotics“. Jde o kombinaci kooperujícího robota a autonomní plošiny. Výsledkem je samostatně se pohybující „vozítko“, znázorněné na obrázku 2.6, jehož součástí je právě již popsané kooperující rameno LBR iiwa. Autonomnost plošiny je zajištěna pomocí laserových skenerů, které nepřetržitě monitorují své okolí a na jakoukoliv překážku dokáží okamžitě reagovat. KMR iiwa se pohybuje na základě předem definované trasy s přesností polohování až ±5 mm. O zásobu elektrické energie se starají lithium-iontové baterie, umístěné v konstrukci pojezdu. Tento typ kooperujícího robota je v průmyslu využíván pro autonomní rozvoz materiálu po výrobní hale (KMR iiwa, 2019).

Obrázek 2.6 – Autonomní vozítko KMR iiwa od společnosti Kuka (KMR iiwa, 2019)

2.5 KOOPERUJÍCÍ ROBOTY SPOLEČNOSTI ABB

Základem nabídky kooperujících robotů švýcarské společnosti ABB jsou dva typy robotů řady YuMi. Marketingově chytře zvolený název vznikl sloučením dvou anglických slov. YuMi znamená „You and Me“, neboli „ty a já“, což přesně odpovídá účelu robotů této řady, tedy blízké spolupráci s člověkem. V prvním případě jde o dvouramenného robota, který je podrobněji popsán až v pododdílu 2.5.1, a to z důvodu jeho využití při realizaci zadání této diplomové práce. Druhým typem je jednoramenná verze uvedená na obrázku 2.7. Primárně jsou oba roboty navrženy pro přesnou montáž malých dílů (IRB 14000 YuMi, 2020).

Jednoramenný kooperující robot s modelovým označením IRB 14050 je nejnovějším přírůstkem řady YuMi. Oproti většině konkurenčních robotů nabízí pohyb v sedmi osách, což přináší ještě vyšší flexibilitu manipulátoru. Hmotnost ramene je 9,5 kg. S nízkou vahou zde souvisí i nízké užitečné zatížení, jehož hodnota činí pouhých 500 g. Díky své hmotnosti ho lze připevnit do libovolné polohy, ať už na podlahu, stěnu nebo strop. Jedním z příkladů využití může být i spolupráce se zmíněnou dvouramennou verzí při některé z montážních operací (IRB 14050 Single-arm YuMi Collaborative Robot, 2020).

Obrázek 2.7 – Jednoramenný kooperující robot YuMi od společnosti ABB (IRB 14050 Single-arm YuMi Collaborative Robot, 2020)

2.5.1 Dvouramenný kooperující robot YuMi

Jak již bylo nastíněno, jde o kooperující robot, který je součástí zadání této diplomové práce. Dvouramenný robot YuMi s označením IRB 14000, znázorněný na obrázku 2.8, se skládá ze základny a dvou ramen. K ramenům se přistupuje jako ke dvěma na sobě nezávislým robotům a každé z nich je programováno samostatně. Má lehkou, ale pevnou kostru vyrobenou z hořčíkové slitiny. Kostra je pokryta pružným plastovým pláštěm, zabaleným do měkkého polstrování. Díky tomu je robot schopen absorbovat sílu neočekávaných kolizí.

Obrázek 2.8 – Dvouramenný kooperující robot YuMi od společnosti ABB (IRB 14000 YuMi, 2020)

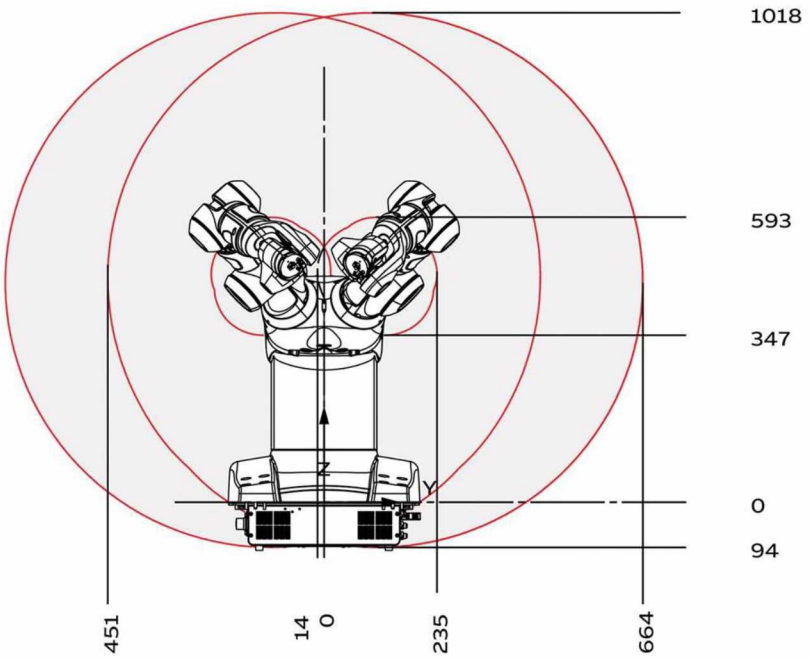
Robot obsahuje integrovaný řídicí systém IRC5, pomocí kterého je řízena přesnost a rychlost pohybu, doba cyklu, programovatelnost a synchronizace s externími zařízeními. K řídicímu systému je připojen operátorský panel FlexPendant, který lze využít například pro programování robotů, komunikaci s obsluhou apod. Na obrázku 2.9 je ukázán příklad skříně s řídicím systém IRC5, na které je položen panel FlexPendant (YuMi, 2015).

Obrázek 2.9 – Systém IRC5 s panelem FlexPendant (IRC5, 2020)

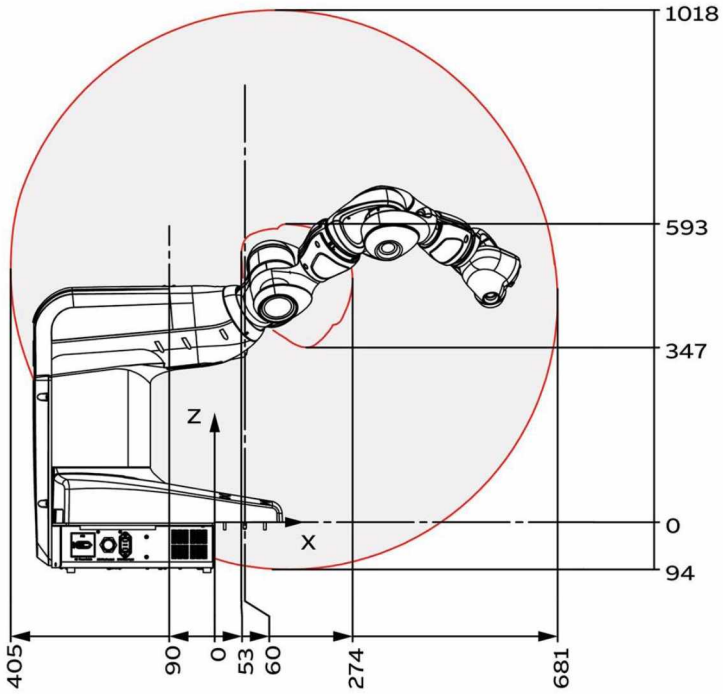
Hmotnost robota je 38 kg. Opakovaná přesnost polohy neboli opakované najetí nástroje do konkrétního bodu je ±0,02 mm s maximální rychlostí pohybu 1,5 m/s a užitečným zatížením 500 g. Každá z paží se dokáže ohýbat v sedmi osách. Pracovní rozsahy pohybů v jednotlivých osách, včetně maximálních rychlostí, jsou uvedeny v tabulce 2.3. Na obrázku 2.10 je znázorněn pracovní rozsah ramen při pohledu zepředu. Boční pohled je uveden na obrázku 2.11. Fyzické uspořádání os reálného robota je ukázáno na obrázku 2.12 (YuMi, 2015).

Tabulka 2.3 – Pohyby robota ABB YuMi (YuMi, 2015)

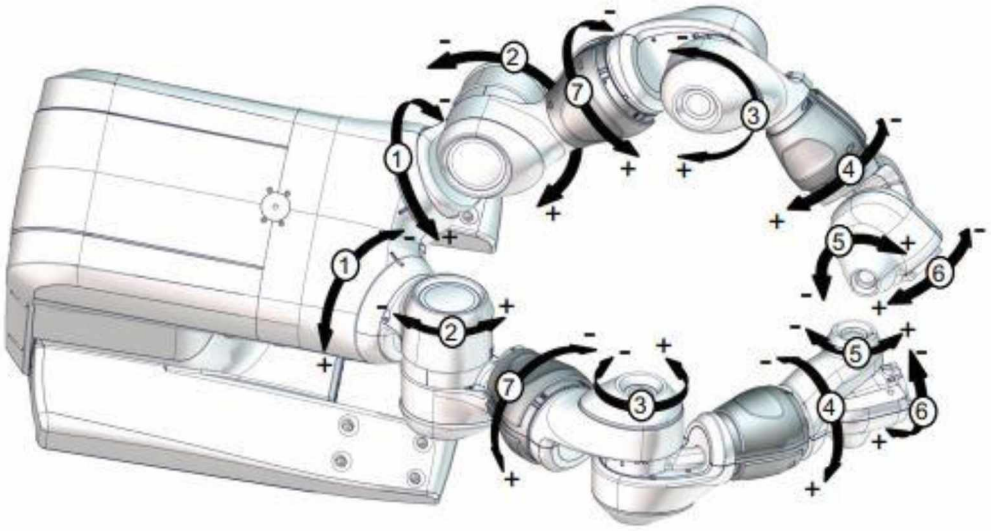
Pohyb na osách	**Pracovní rozsah**	**Maximální rychlost**
Osa 1 rotace	−168,5° až +168,5°	180°/s
Osa 2 paže	− 143,5° až +43,5°	180°/s
Osa 3 paže	−123,5° až +80,0°	180°/s
Osa 4 zápěstí	−290,0° až +290,0°	400°/s
Osa 5 ohýbání	−88,0° až +138,0°	400°/s
Osa 6 otáčení	−229,0° až +229,0°	400°/s
Osa 7 rotace	−168,5° až +168,5°	180°/s



Obrázek 2.11 – Pracovní rozsah dvouramenného robota YuMi při pohledu zepředu (Technical data IRB 14000 YuMi, 2020)



Obrázek 2.10 – Pracovní rozsah dvouramenného robota YuMi při pohledu z boku (Technical data IRB 14000 YuMi, 2020)



Obrázek 2.12 – Fyzické uspořádání os dvouramenného robota YuMi (IRB 14000 YuMi, 2020)

Průmyslové i kooperující roboty mohou být vybaveny různými druhy koncových nástrojů neboli efektorů, a to v závislosti na požadované činnosti. Konkrétně pro dvouramenného robota YuMi společnost ABB nabízí efektory, které jsou v dokumentaci označovány jako „IRB 14 000 gripper“. Ty se mohou skládat z různých kombinací uchopovacích prvků, mezi které patří dvojice mechanických prstů a pneumatická přísavka. Součástí efektoru může být i integrovaná kamera. Nabízené kombinace jsou uvedeny v tabulce 2.4, a některé z nich ukázány na obrázku 2.13. Veškeré doplňující informace lze dohledat ve volně přístupné dokumentaci na stránkách společnosti ABB.

Tabulka 2.4 – Kombinace prvků nástrojů IRB 14000

Varianta	Kombinace prvků
1.	Prsty
2.	Prsty + jedna pneumatická přísavka
3.	Prsty + dvě pneumatické přísavky
4.	Prsty + kamera
5.	Prsty + kamera + jedna pneumatická přísavka

Obrázek 2.13 – Efektory typu IRB 14 000 gripper (IRB 14000 YuMi, 2020)

3 PROGRAMOVÁNÍ ROBOTŮ

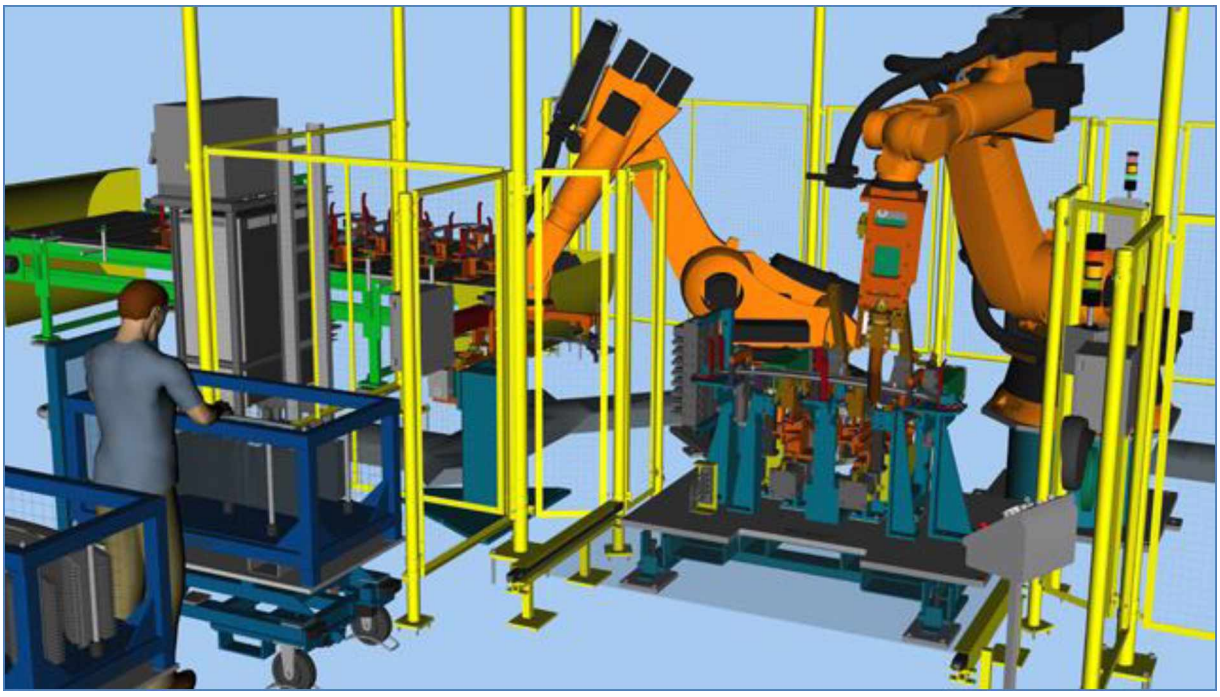
3.1 PŘÍSTUPY K PROGRAMOVÁNÍ ROBOTŮ

K programování robotů lze přistupovat pomocí dvou základních přístupů. Existují metody tak zvaného online a offline programování.

Metoda online je založena na přímé interakci programátora s robotem. Provádí se přímo na pracovišti prostřednictvím operátorského panelu teach-pendant. Jako příklad je na obrázku 3.1 znázorněn teach-pendant pro práci s roboty společnosti Kuka. Pomocí panelu je robot naváděn do požadovaných pozic, které jsou následně uloženy a zapsány do programu. Ze zaznamenaných pozic jsou vytvořeny pohybové instrukce, ke kterým jsou postupně přidávány i instrukce pro ovládání koncového efektoru nebo jiných periferních zařízení. Pomocí tohoto postupu je vytvořen celý program, který je rovnou zapsán do řídicího systému robota. Mezi výhody tohoto přístupu patří fakt, že programátor pracuje přímo v reálném prostředí. Tím se nabízí možnost okamžité kontroly funkčnosti programu, případnému odstranění nepřesností nebo změnám programu podle aktuálních požadavků koncového zákazníka. Nevýhodou je nutnost odstavení výroby na daném pracovišti. Z tohoto důvodu dochází ke ztrátám při výrobě a vysokému zatížení programátora. Tyto nevýhody znatelně eliminuje druhý z přístupů – offline programování.

Obrázek 3.1 – Teach-pendant (WIRELESS TEACH PENDANTS FOR ROBOTS, 2019)

Principem offline metody je vytvoření virtuálního 3D modelu robotického pracoviště, který se v co nejvíce ohledech shoduje s reálným zařízením. To umožňuje programování na počítači bez nutnosti zastavení výroby. Na základě virtuální simulace procesu je možné ještě před fyzickou realizací připravit programy robotů, optimalizovat jejich pohyb a celkový pracovní čas, zamezit kolizím, ověřit dosažitelnost bodů v pracovním prostoru atp. Takto předem připravený program se následně nahraje do řídicího systému reálného robota. Díky tomu lze zkrátit dobu odstavení výrobní linky či jiného průmyslového zařízení na minimum a cílovému zákazníkovi tak ušetřit nemalé finanční prostředky. Příkladem může být výroba automobilů ve společnosti Škoda, kde každou minutu sjede z výrobní linky nový automobil. Při dlouhodobém odstavení linky by došlo k velkým ztrátám na výrobě. Právě to může být jeden z důvodů, kvůli kterému by měla být případná změna programu robota časově efektivní. Po nahrání programu do reálného řídicího systému je nutné ověřit jeho funkčnost a odladit případné neshody mezi virtuálním modelem a reálným zařízením. V zájmu programátora by tedy mělo být vytvořit co nejvěruhodnější model výrobního systému, a to z důvodu minimálního počtu oprav v reálné výrobě. Příklad virtuálního modelu robotického pracoviště je uveden na obrázku 3.2.



Obrázek 3.2 – Virtuální 3D model robotického pracoviště (Engineer automated production system using robotics and automation Simulation, 2020)

3.2 SOFTWARE PRO PROGRAMOVÁNÍ ROBOTŮ

V dnešní době je kladen důraz především na vývoj kvalitních simulačních software, která umožňují již popsané offline programování. Ve většině případů výrobci průmyslových robotů vyvíjí svůj vlastní software, který je kompatibilní pouze s roboty dané společnosti. To znamená, že například v software RoboDK od společnosti Universal Robots, lze pracovat pouze s roboty tohoto výrobce. Zde jsou uvedeny další příklady výrobců a jejich software:

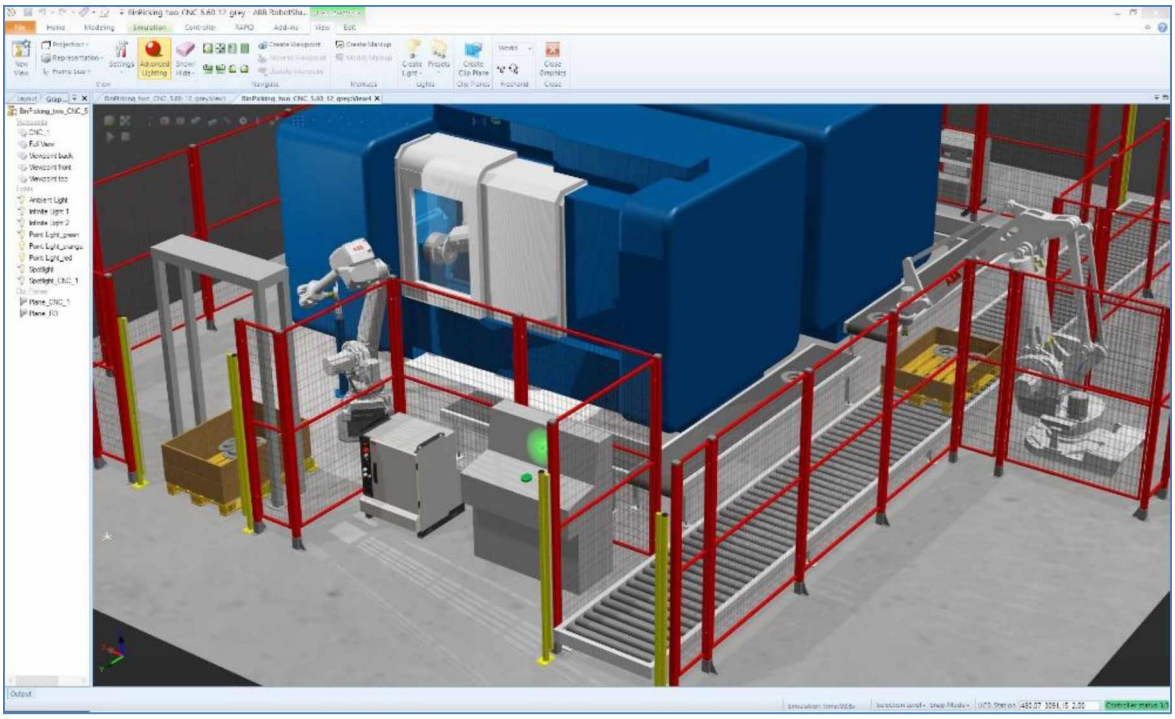
- KUKA – KUKA.Sim,
- Stäubli – Stäubli Robotics Suite,
- Mitsubishi – MELFA-Works,
- ABB – RobotStudio,
- FANUC – ROBOGUIDE,
- a další.

Mimo to existují i univerzální SW nástroje, které umožňují pracovat s roboty různých výrobců včetně jejich programovacích jazyků. Mezi příklady univerzálních software patří:

- WorkSpace,
- RobCad,
- RobotWorks,
- a další (AUTOMATIZACE PRŮMYSLOVÉHO BALENÍ: Simulace v robotizaci, 2011).

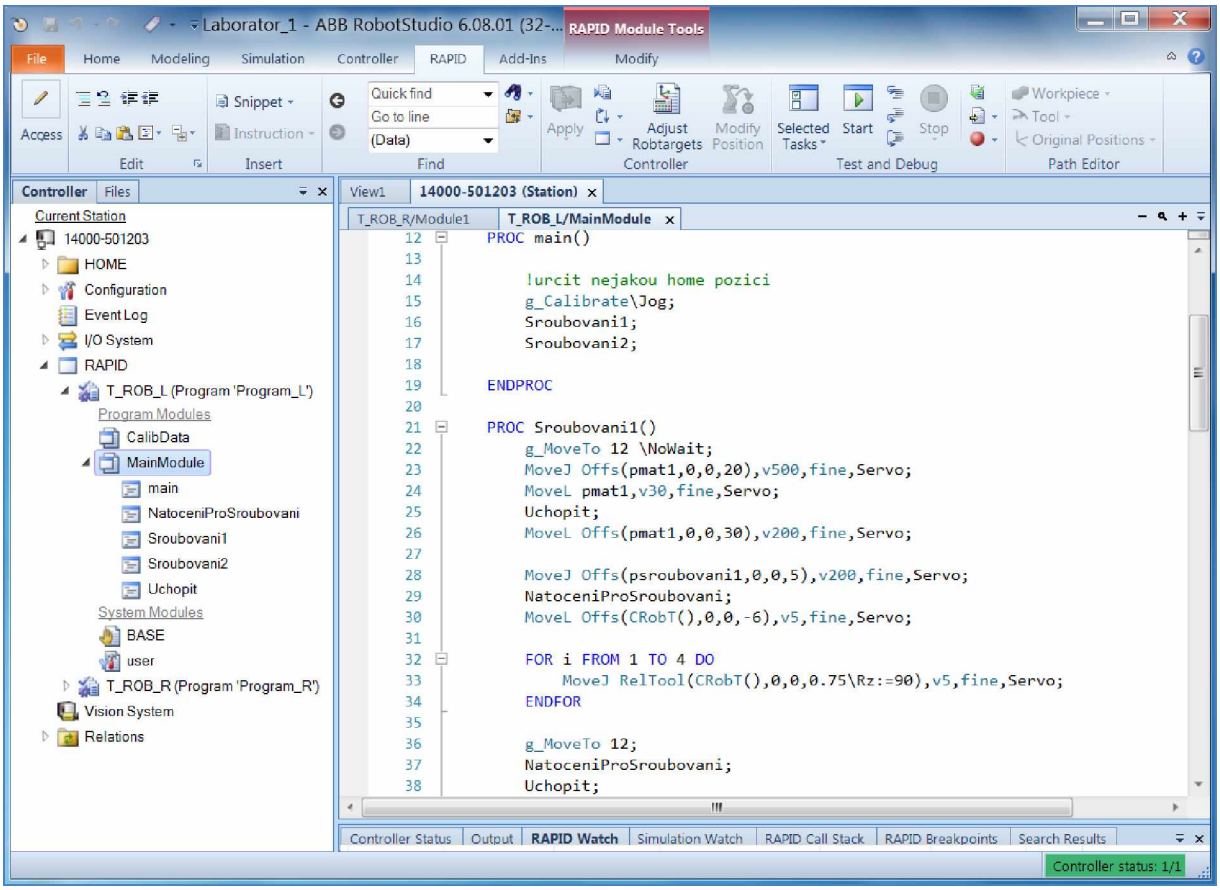
3.2.1 Software RobotStudio od společnosti ABB

Uživatelské programovací prostředí RobotStudio pracuje s přesnou kopií opravdového software, kterým je řízen reálný robot ve výrobě. Tím se nabízí možnost velmi realistické simulace výrobního procesu, a to s využitím robotických programů a konfiguračních souborů, totožnými s těmi, které se následně použijí i pro řízení reálného robota. Příklad virtuálního modelu robotického pracoviště, vytvořeného v software RobotStudio, je znázorněn na obrázku 3.3 (RobotStudio, 2020).

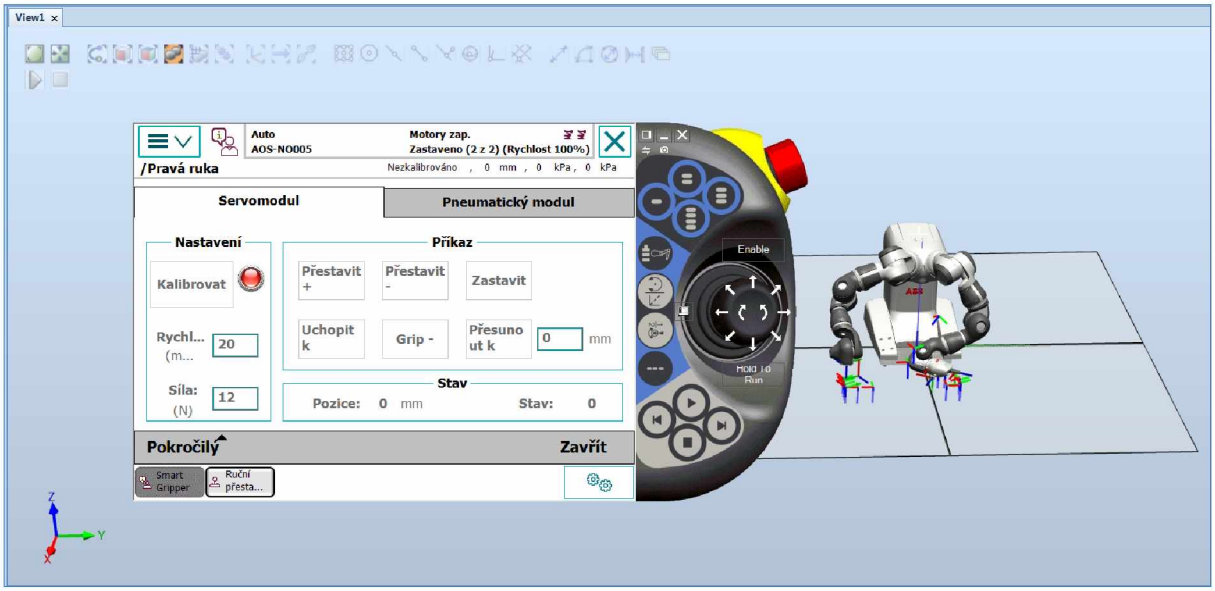


Obrázek 3.3 – Virtuální model robotického pracoviště v software RobotStudio

K tvorbě programů slouží vyšší programovací jazyk RAPID. Program je psán v integrovaném textovém editoru, ukázaném na obrázku 3.4. Editor nabízí funkce jako automatické doplňování argumentů, zvýraznění syntaxe a chyb, možnost ladění programu pomocí tzv. „breakpoints“ apod. Vytvořený program lze následně nahrát do kontroléru robota, a to buď reálného nebo virtuálního. Programovací jazyk RAPID je podrobněji popsán až v kapitole 4. Software dále nabízí řadu užitečných funkcí, pomocí kterých lze tvorbu programu zefektivnit. Například umožňuje import modelů různých CAD formátů (IGES, STEP, ACIS,…), které se následně jednoduše implementují do samotné simulace. S tím úzce souvisí i funkce *Smart Components*. Díky ní lze objektům přiřadit různé vlastnosti chování, a to za účelem co nejreálnější simulace procesu. Existují i funkce pro automatické generování trajektorie pohybu v závislosti na geometrii vloženého 3D modelu. Funkce *Virtual FlexPendant* umožňuje pracovat s virtuálním ovladačem robota, jak je tomu ukázáno na obrázku 3.5. To lze například využít pro školení obsluhy mimo reálné zařízení. Dalším příkladem zajímavé funkce je *Visual SafeMove*. Jde o grafický nástroj pro konfiguraci bezpečnostních zón v okolí robota, které lze v software vizualizovat ve 3D zobrazení (RobotStudio, 2020).

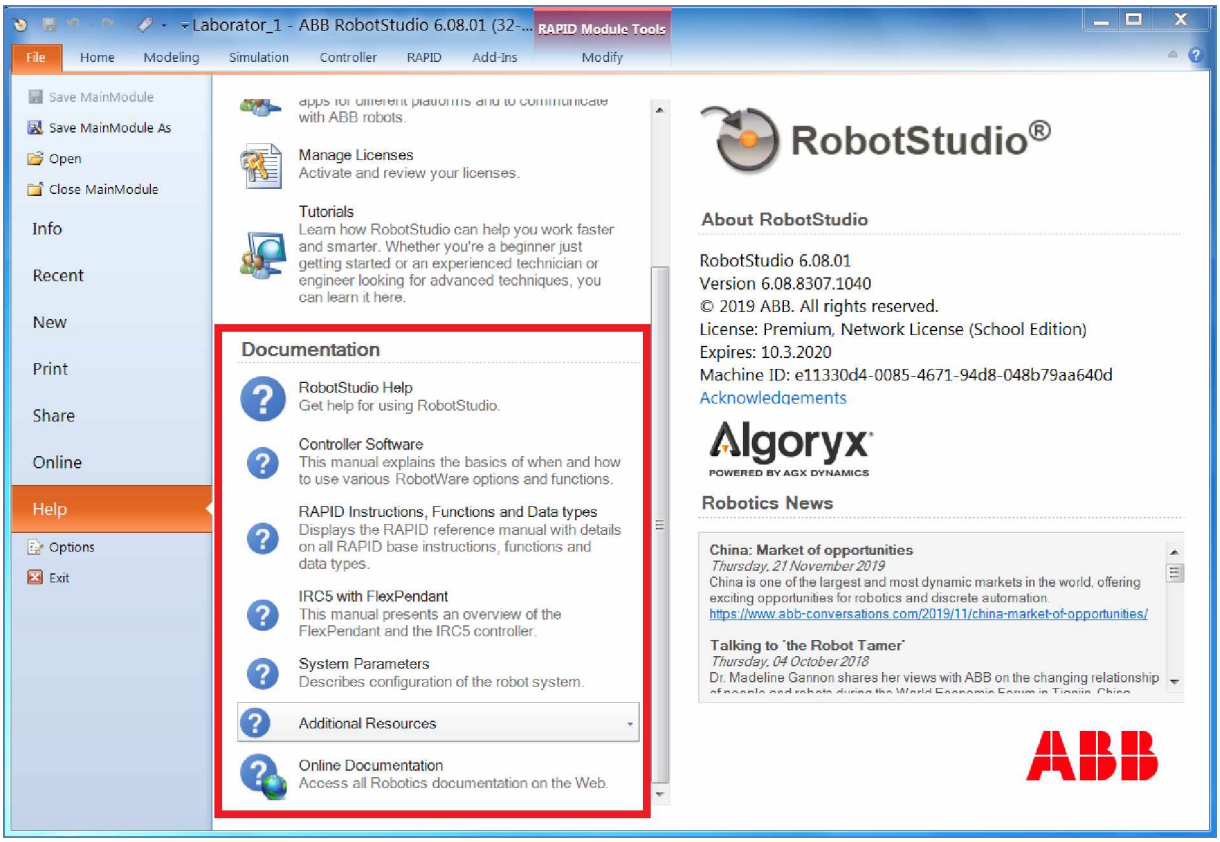


Obrázek 3.5 – Editor pro psaní programu v software RobotStudio



Obrázek 3.4 – Funkce Virtual FlexPendant v software RobotStudio

Součástí software je i rozsáhlá nápověda, jejíž dokumenty se nachází pod záložkou *File* a *Help*. Umístění manuálů v software RobotStudio je znázorněno na obrázku 3.6. Zde lze najít veškeré manuály potřebné pro tvorbu aplikací, práci s panelem FlexPendant, konfiguraci robotů atp. Totožné manuály jsou také volně dostupné na webových stránkách společnosti ABB.



Obrázek 3.6 – Manuály pro práci v software RobotStudio

Z webových stránek společnosti ABB lze zdarma získat i omezenou verzi software RobotStudio na dobu 30 dní. Plnohodnotná verze Premium je zpoplatněna, přičemž její cena se pohybuje okolo 30 tisíc korun na jeden rok. K této verzi lze dokoupit rozšiřující balíčky, jako například CAD Converter, nebo balíčky pro specifické aplikace (sváření, paletizaci, lakování apod.). Cena takového doplňku se pohybuje v rozmezí od 15 do 30 tisíc korun (Ceník RobotStudio 6, 2018).

4 PROGRAMOVACÍ JAZYK RAPID

Existuje velké množství výrobců robotů, přičemž každý z nich vyvíjí svůj vlastní software, pomocí kterého je robot od konkrétního výrobce programován. Různým software odpovídají i různé programovací jazyky. Zde jsou uvedeny příklady programovacích jazyků pro software vybraných společností:

- KUKA – Kuka Robot Language,
- Stäubli – VAL3,
- Universal Robots – URScript,
- FANUC – Karel,
- a další.

K programování robotů od společnosti ABB, ve vývojovém prostředí RobotStudio, slouží programovací jazyk RAPID. Jazyk RAPID je velmi rozsáhlý. S ohledem na potřeby diplomové práce je v této kapitole popsán pouze výběr základních funkcí a vlastností tohoto jazyka. Podrobnější informace lze dohledat ve volně přístupných dokumentacích, a to přímo na webových stránkách ABB.

4.1 STRUKTURA JAZYKA

Program se skládá z posloupnosti instrukcí a funkcí, které vedou robota k vykonání požadovaných úkonů. Příkladem mohou být instrukce pro ovládání nástroje, pohybů ramene, časovačů, vstupů a výstupů apod. Funkce fungují velice podobně, pouze s tím rozdílem, že vrací určitou hodnotu. Například jsou používány pro výpočty pozic robota, matematické operace atp. Instrukce i funkce se většinou skládají z řady argumentů, které definují jejich přesnou činnost. Argumenty je možné rozdělit do dvou základních skupin – povinné a volitelné. Ty od sebe lze odlišit hned při pohledu na syntaxi dané instrukce nebo funkce. Povinné argumenty nejsou ohraničeny hranatými závorkami a jejich definice je nutná pro funkčnost programu. Volitelné argumenty jsou uzavřeny do hranatých závorek a lze je při psaní programu vynechat. Existují i instrukce a funkce, u kterých se argumenty vzájemně vylučují a nemohou být definovány současně. Takové argumenty jsou od sebe v syntaxi odděleny svislou čarou „|“. Dále je uveden příklad syntaxe jedné z instrukcí, kde je vidět, že instrukce *TPWrite* se skládá z jednoho povinného argumentu *String* a pěti volitelných, vzájemně se vylučujících argumentů *\Num*, *\Bool*, *\Pos*, *\Orient* a *\Dnum*.

```
TPWrite String [\Num]|[\Bool]|[\Pos]|[\Orient]|[\Dnum];
```

Pro lepší orientaci v programu se doporučuje používat komentáře. Již v této kapitole jsou komentáře používány při demonstraci částí kódů programovacího jazyka RAPID. Jsou jednou z nejběžněji používaných instrukcí a žádným způsobem neovlivňují chod programu. Komentář začíná symbolem „!“ a končí prvním znakem nového řádku. V programu jsou zvýrazněny zelenou barvou. Použití komentáře je ukázáno v následující části kódu.

```
!Nastavení výchozí pozice
MoveL p10, v50, z10, tool1;
```

Program robota je dělen na programové moduly, přičemž každý z nich může obsahovat jednu či více procedur. Pouze jeden z modulů obsahuje proceduru *main,* která je automaticky volána při spuštění programu a vždy musí být jeho součástí. Z procedury *main* jsou následně volány další části kódu. Existuje i tzv. systémový modul, který je používán pro deklaraci dat a procedur společných pro celý systém – typickým příkladem může být definice nástrojů. Pro realizaci krátkých a jednoduchých programů si programátor vystačí pouze s jedním programovým modulem. Naopak při programování složitějších aplikací jich lze definovat libovolně mnoho. Rozdělením komplexnějších aplikací do více modulů lze dosáhnout vyšší přehlednosti programu. Modul je deklarován klíčovým slovem *MODULE*, za kterým následuje jeho název a ukončen příkazem *ENDMODULE*. Pomocí procedur lze kód rozdělit dle konkrétních činností. Procedura je deklarována klíčovým slovem *PROC*, za kterým následuje název procedury, vstupní argumenty a kód, který má být procedurou vykonán. Procedura je následně ukončena příkazem *ENDPROC*. Jednotlivé procedury mohou být volány z jiných procedur pomocí instrukce *ProcCall*. S proměnnou, která byla deklarována mimo proceduru, lze pracovat v rámci celého modulu. Naopak proměnné, deklarované uvnitř procedury, mohou být použity pouze danou procedurou. Zmíněné informace jsou demonstrovány v přiložené části kódu.

```
MODULE MainModule
     !Automaticky volaná procedura main
     PROC main()
          !Volání procedury s argumenty
          Start 5, "Ahoj";
          !Volání procedury bez argumentů
           Pohyb;
     ENDPROC
     !Deklarované procedury
     PROC Start(num pocet, string pozdrav)
          !Kód procedury
          …
```

```
    ENDPROC
    PROC Pohyb()
        !Kód procedury
        ...
    ENDPROC
ENDMODULE
```

4.2 ŘÍZENÍ TOKU PROGRAMU A DATOVÉ TYPY

Program je zpravidla vykonáván sekvenčně, a to příkaz po příkazu. Sekvenční chod může být narušen několika způsoby. Typickým příkladem je volání jiné procedury, jenž je potřebná pro řešení nastalé situace. Po vykonání této procedury program pokračuje instrukcí, která následuje instrukci pro volání. Pro řízení toku programu lze například použít i instrukce typické pro většinu programovacích jazyků, mezi které patří instrukce typu *IF*, *FOR*, *WHILE* apod. Jazyk RAPID také umožňuje obsluhu přerušení. To spočívá v definici podmínky pro přerušení a události, která má být v případě splnění zadané podmínky vykonána. Podmínkou pro přerušení může být například změna hodnoty binární proměnné vstupního nebo výstupního signálu. S touto proměnnou je spojena procedura pro obsluhu přerušení. V případě, kdy dojde ke splnění zadané podmínky, dojde k okamžitému zastavení programu a zahájí se vykonávání odpovídající procedury, pomocí které je nastalá situace vyřešena. Program poté pokračuje od místa, kde došlo k jeho přerušení. Užitečnou funkcí jazyka je i obsluha chyb. Velké množství chyb, které nastanou v průběhu vykonání programu, může být vyřešeno v rámci programu a nemusí tak dojít k jeho zastavení. Součástí programových procedur totiž může být i část kódu, která obsluhu nastalé chyby zajistí. Například pokud se uživatel pokusí o dělení nulou, může mu být díky této části kódu zobrazeno chybové hlášení a umožněno zadání nových hodnot. V případě, kdy podobná opatření nejsou definována, dojde k vyvolání systémové chyby a ukončení chodu programu. Pro zobrazení podobného chybového hlášení lze využít možnosti komunikace mezi systémem robota a obsluhou pomocí panelu FlexPendant. Zde se nachází operátorské okno, které lze použít jak pro zobrazení zpráv od robota, tak i pro zadávání pokynů robotovi – uživatel takto například může zadat požadovaný počet výrobků ke zkompletování.

RAPID umožňuje pracovat s mnoha různými datovými typy, které lze využít pro širokou škálu funkcí jazyka, jako například datové typy určené pro práci se strojovým viděním, připojenými senzory atp. V tabulce 4.1 je pro názornost uveden výběr pouze těch nejběžněji používaných datových typů, pomocí kterých jsou v níže uvedené části kódu demonstrovány

základní operace, jako je deklarace proměnné a přiřazení hodnoty. Ještě předtím jsou ale popsány způsoby uložení dat různých datových typů:

- Konstanta (*CONST*): Hodnotu přiřazenou při deklaraci konstanty již nelze dále v programu měnit.
- Proměnná (*VAR*): Hodnotu proměnné lze v průběhu vykonávání programu měnit. Po ukončení a opětovném nastartování programu, si nepamatuje poslední přiřazenou hodnotu.
- Persistentní proměnná (*PERS*): Persistentní proměnná je téměř totožná s klasickou proměnnou (*VAR*), ale liší se v jedné zásadní věci. Persistentní proměnná si i po ukončení a opětovném nastartování programu pamatuje poslední přiřazenou hodnotu. Například u dvouramenného robota YuMi se persistentní proměnná také využívá pro deklaraci dat společných pro obě ramena, kde každé z nich má svůj vlastní program.

Tabulka 4.1 – Základní datové typy jazyka RAPID

Datový typ	Popis
num	Číselná hodnota – celé i desetinné číslo.
string	Řetězec znaků neboli text s maximálním počtem znaků 80.
bool	Logická hodnota – nabývá pouze dvou hodnot – TRUE nebo FALSE.

Následující kód demonstruje základní operace s datovými typy.

```
!Deklarace konstanty datového typu num
CONST num Pocet;
!Deklarace proměnné datového typu string
VAR string Pozdrav;
!Deklarace persistentní proměnné logického typu
PERS bool Dokonceno;
!Přiřazení hodnot pomocí operátoru „:="
Pocet := 5;
Pozdrav := "Ahoj";
Dokonceno := TRUE;
```

Mimo zmíněné jednoduché datové typy jsou v jazyce RAPID často využívány i strukturované datové typy, které jsou složeny z více položek stejného nebo různého typu. Výsledkem je spojení několika elementů do jedné proměnné. Takto složená proměnná bývá argumentem běžně používaných instrukcí. Příkladem může být datový typ *pose*, jenž slouží k uložení informace o posunutí a rotaci souřadnicového systému vůči jinému. Proměnná

datového typu *pose* se skládá ze dvou položek různých datových typů, a to *pos* a *orient*. Typ *pos* definuje posunutí souřadnicového systému v osách *x*, *y* a *z*, které je vyjádřeno v milimetrech. Typ *orient* určuje jeho natočení pomocí kvaternionu o složkách *q1*, *q2*, *q3*, *q4*. Kvaterniony jsou určitým rozšířením oboru komplexních čísel v matematice – skládají se ze čtyř složek (jedna reálná složka a tři imaginární) a v robotice jsou využívány právě pro popis rotace v prostoru. V programu jsou od sebe položky různých typů odděleny hranatými závorkami, jak je tomu ukázáno v níže uvedeném kódu.

```
!Syntaxe
VAR pose NÁZEV := [[pos], [orient]];
!Složení položek
VAR pose NÁZEV := [[x, y, z], [q1, q2, q3, q4]];
```

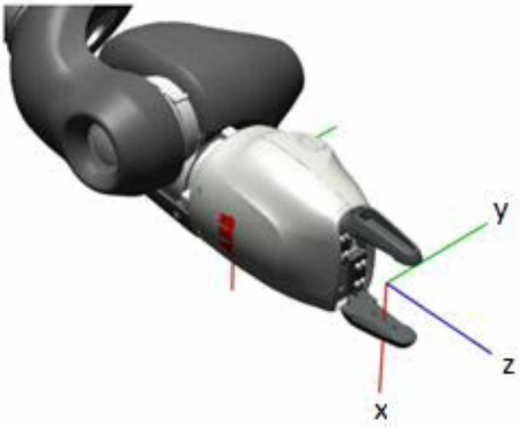
Dalším vhodným příkladem je datový typ *robtarget*, který se skládá ze čtyř položek odlišného typu. Typ *robtarget* je úzce spjat s pohybem robota, proto je podrobněji popsán až v oddílu 4.3, který je na tuto problematiku zaměřen. Právě tam lze dohledat i informace ohledně složení tohoto datového typu.

U strukturovaných datových typů lze pracovat i s konkrétními parametry vybraných položek. Příkladem může být již popsaný typ *pos*, u kterého je možné definovat hodnotu posunutí pouze jedné z os, jak je ukázáno v přiloženém kódu.

```
!Příklad
VAR pos pos01
pos01.x := 50;
pos01.y := pos01.y + 10;
```

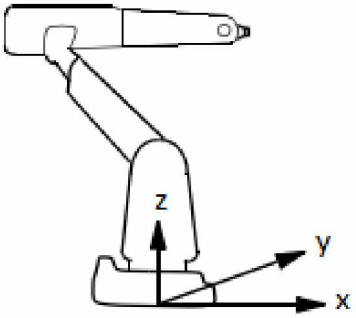
4.3 INSTRUKCE PRO POHYB ROBOTA

Pozice robota a jeho pohyby se vždy vztahují ke středu nástroje, který je z pravidla označován jako TCP („tool center point“). Jde bod, ve kterém je definován počátek souřadnicového systému nástroje. TCP se může nacházet kdekoliv na nástroji v závislosti na typu konkrétního nástroje. Příkladem může být TCP umístěný ve středu uchopovacího efektoru, jak je tomu ukázáno na obrázku 4.1. V programu může být definován libovolný počet takovýchto nástrojů a jejich TCP, nicméně nově vytvářené pozice robota se vždy vztahují pouze aktuálně zvolenému nástroji. Následně se právě TCP vybraného nástroje pohybuje podél trajektorie, definované pohybovými instrukcemi.



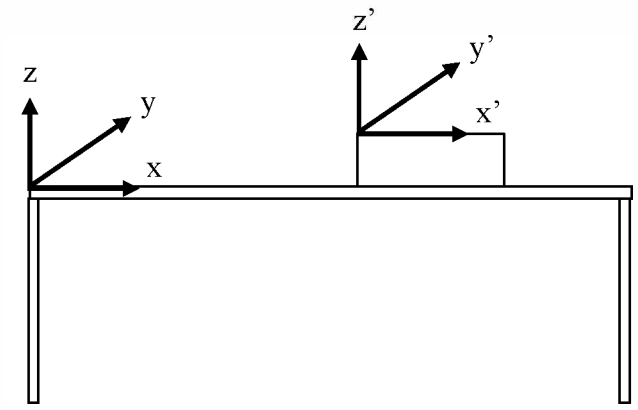
Obrázek 4.1 – TCP uchopovacího efektoru

Poloha nástroje (TCP) může být definována v různých souřadnicových systémech robota. V případě, že uživatel neučiní jinak, tak se veškeré pozice vztahují k základnímu souřadnicovému systému (*base coordinate system*). Základní souřadnicový systém, znázorněný na obrázku 4.2, se nachází v základně robota a lze s ním vystačit při programování jednoduchých aplikací.



Obrázek 4.2 – Základní souřadnicový systém robota

Uživateli je ale umožněno definovat si libovolné souřadnicové systémy, a to kdekoliv v pracovním prostoru robota. K tomuto účelu slouží datový typ *wobjdata*. Běžná je situace, kdy si uživatel v základním souřadnicovém systému robota definuje nový uživatelský souřadnicový sytém (*user frame*). V tomto uživatelském souřadnicovém systému může být dále definován i podřadný souřadnicový systém objektu (*object frame*). Příkladem je situace uvedená na obrázku 4.3, kde je uživatelský souřadnicový systém umístěn v rohu pracovního stolu robota a jemu podřadný souřadnicový systém objektu se nachází přímo na objektu, na kterém bude robot vykonávat pracovní činnost. Tím lze docílit vyšší přehlednosti při tvorbě složitějších aplikací.



Obrázek 4.3 – Uživatelské souřadnicové systémy

Pozice robota jsou definované strukturovanými datovými typy *robtarget* nebo *jointtarget*. Typ *robtarget* určuje pozici bodu pomocí souřadnic v aktuálně zvoleném souřadnicovém systému. Robot je ale schopný dosáhnout jedné a té samé pozice několika různými způsoby. Z toho důvodu jsou jeho součástí i konfigurace natočení os. Strukturovaný datový typ *robtarget* se skládá z těchto položek:

- *Pos*… definuje pozici (*x*, *y* a *z*) středu nástroje (TCP) vyjádřenou v milimetrech.
- *Orient*… definuje otočení nástroje pomocí kvaternionu (*q1*, *q2*, *q3* a *q4*).
- *Confdata*… definuje konfigurace os robota (*cf1*, *cf4*, *cf6* a *cfx*).
- *Extjoint*… definuje pozice až šesti externích os (*eax_a* až *eax_f*).

Dále je ukázána syntaxe datového typu *robtarget* včetně složení jeho položek.

```
!Syntaxe
robtarget NÁZEV:=[[pos],[orient],[confdata],[extjoint]];
!Složení položek
robtarget NÁZEV:=[[x,y,z],[q1,q2,q3,q4],[cf1,cf4,cf6,cfx],
[eax_a,eax_b,eax_c,eax_d,eax_e,eax_f]];
```

Další možností je použití datového typu *jointtarget*, který slouží k určení pozice pomocí úhlu natočení jednotlivých os robota a externích os. Ten se skládá z následujících položek:

- *Robjoint*… definuje úhly natočení jednotlivých os robota vyjádřených ve stupních (*rax_1* až *rax_6*). V případě, kdy má robot více než 6 os (například YuMi od společnosti ABB), jsou úhly natočení dalších os definovány na pozicích externích os.
- *Extjoint*… definuje pozice externích os (*eax_a* až *eax_f*).

Níže je ukázána syntaxe datového typu *jointtarget* včetně složení jeho položek.

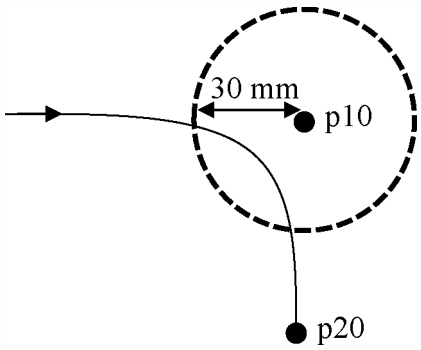
```
!Syntaxe
jointtarget NÁZEV:=[robjoint],[extjoint]];
!Složení položek
jointtarget NÁZEV:=[[rax_1,rax_2,rax_3,rax_4,rax_5,rax_6],
[eax_a,eax_b,eax_c,eax_d,eax_e,eax_f]];
```

Samotný pohyb robota je vykonáván na základě pohybových instrukcí. Tento pohyb lze chápat jako přesun z aktuální pozice do nové, přičemž cestu mezi těmito dvěma body si robot automaticky vypočítá. Instrukce *MoveL* a *MoveJ* zajistí přesun nástroje (TCP) z aktuální pozice do definované, která musí být deklarována pomocí datového typu r*obtarget*. Navzájem se od sebe liší způsobem vykonání tohoto pohybu. Při použití instrukce *MoveL* nástroj provede lineární pohyb do definované pozice. *MoveJ* se naopak použije v případě, kdy tento pohyb nemusí být lineární. Součástí obou instrukcí jsou totožné argumenty, mezi které patří:

- *ToPoint*… definuje cílový bod, do kterého se má nástroj přesunout.
- *Speed*… definuje rychlost nástroje při pohybu. Kde například *v100* odpovídá rychlosti 100 mm/s.
- *Zone*… určuje přesnost projetí definovaného bodu v trajektorii pohybu. Kde například *z30* znamená, že nástroj musí projet oblastí o poloměru 30 mm v okolí zadaného bodu. Pokud uživatel zadá hodnotu *fine*, tak nástroj najede přesně do definovaného bodu. Tím lze ovlivnit i způsob vykonání pohybu v případě, kdy se v programu nachází sekvence několika pohybových instrukcí. Pokud není zadaná hodnota *fine*, dojde k plynulému spojení pohybů po sobě jdoucích instrukcí. Naopak v případě, kdy je u některé z instrukcí použita hodnota *fine*, tak robot musí najet přímo do definovaného bodu, zde pohyb přerušit a až poté zahájit vykonávání následující instrukce. Funkce tohoto argumentu je demonstrována pomocí níže uvedeného příkladu a obrázku 4.4.
- *Tool*… definuje nástroj, jehož TCP se přesune do cílového bodu.

V následujícím kódu je uvedena syntaxe instrukcí *MoveL* a *MoveJ*. Součástí je i příklad použití, pomocí něj a obrázku 4.4 je také ukázána funkce argumentu *Zone*.

```
!Syntaxe
MoveL ToPoint, Speed, Zone, Tool;
MoveJ ToPoint, Speed, Zone, Tool;
!Příklad
MoveJ p10, v100, z30, tool0;
MoveJ p20, v100, fine, tool0;
```



Obrázek 4.4 – Funkce argumentu Zone

Další možností je vykonání pohybu na základě pozice určené datovým typem *jointtarget*, který definuje pozici robota pomocí úhlů natočení jednotlivých os. K realizaci takového pohybu slouží instrukce *MoveAbsJ*. Její syntaxe je téměř totožná s již popsanými instrukcemi *MoveL* a *MoveJ*. Jediným rozdílem je jiný datový typ cílového bodu – *ToJointPos*. Syntaxi a příklad použití instrukce *MoveAbsJ* demonstruje následující kód.

```
!Syntaxe
MoveAbsJ ToJointPos, Speed, Zone, Tool;
!Příklad
MoveAbsJ JointPos10, v500, z50, tool0;
```

Popsané pohybové instrukce *MoveL* a *MoveJ* lze rozšířit o funkci *RelTool*. Ta zajistí vykonání posuvného pohybu a rotace o požadovanou hodnotu vůči zadané pozici robota. Tento posuvný a rotační pohyb se vztahuje k souřadnicovému systému nástroje (TCP). Funkce *RelTool* se skládá z těchto argumentů:

- *Point*… definuje bod, vůči kterému se nástroj posune o zadanou hodnotu.
- *Dx, Dy, Dz*… definují posun TCP v jednotlivých osách (v milimetrech).
- *Rx, Ry, Rz*… definují rotaci TCP v jednotlivých osách (ve stupních). Pokud uživatel zadá dvě nebo tři rotace najednou, budou vykonávány postupně, a to v pořadí *x*, *y* a *z*.

V přiloženém kódu je ukázána syntaxe a příklady použití funkce *RelTool*.

```
!Syntaxe
RelTool (Point, Dx, Dy, Dz [\Rx] [\Ry] [\Rz]);
!Příklady
!TCP se posune o 5 mm v ose y od bodu p1
MoveJ RelTool(p1,0,5,0), v5, z30, tool0;
!TCP provede rotaci o 90 stupňů v ose z
MoveL RelTool(p1,0,0,0 \Rz:=90), v5, z30, tool0;
```

Velice často se při programování robotu využívá i funkce *Offs*. Její použití je podobné funkci *RelTool*. Slouží k vykonání posuvného pohybu vůči zadanému bodu. Zásadním rozdílem je, že tento pohyb je vykonán v aktuálně zvoleném souřadnicovém systému robota, nikoli v souřadnicovém systému nástroje. Funkci *Offs* tvoří tyto argumenty:

- *Point*… definuje bod, vůči kterému se nástroj posune o zadanou hodnotu.
- *XOffSet*, *YOffSet*, *ZOffSet*… definují posun v osách *x*, *y*, *z* (v milimetrech).

V následujícím kódu je demonstrována syntaxe a příklady použití funkce *Offs*.

```
!Syntaxe
MoveL Offs(Point, XOffSet, YOffSet, ZOffSet);
!Příklady
!Robot se přesune do pozice vzdálené 15 mm v ose z od bodu p2
MoveL Offs(p2,0,0,15), v30, z10, tool0;
!Robot se přesune do pozice vzdálené 5 mm v ose x a 2 mm
v ose y od bodu p3
MoveJ Offs(p3,5,2,0), v500, z100, tool0;
```

V průběhu vykonání programu může nastat situace, kdy je potřeba zjistit aktuální pozici robota. K tomuto účelu lze použít funkce *CrobT* nebo *CjoinT*. *CrobT* získá aktuální pozici datového typu *robtarget*. Naopak funkce *CjoinT* získá pozici datového typu *jointtarget*. Takto získané pozice lze jednoduše uložit a využít dále v programu. Příklady uložení aktuálních pozic jsou demonstrovány pomocí níže uvedeného kódu.

```
!Příklad
VAR robtarget p1;
VAR jointtarget p2;
p1 := CrobT();
p2 := CjoinT();
```

4.4 INSTRUKCE PRO PRÁCI SE VSTUPNÍMI/VÝSTUPNÍMI SINGÁLY A INSTRUKCE PRO ČEKÁNÍ

Robot může být vybaven sadou digitálních a analogových vstupů a výstupů, které lze v rámci aplikace volně využít. Pro práci s těmito signály jsou v jazyku RAPID vyhrazeny zvláštní instrukce. Často používané jsou i instrukce pro čekání, které slouží k pozdržení chodu programu. Instrukce například čekají na splnění zadané podmínky, změnu hodnoty vstupního nebo výstupního signálu, uběhnutí časového intervalu atp., a až poté pokračuje vykonávání programu. Součástí popisu instrukcí jsou i části kódu, pomocí kterých je ukázána jejich syntaxe a demonstrovány příklady použití.

- *SetAO*... slouží k nastavení hodnoty analogového výstupu. Přičemž argument *Signal* definuje název konkrétního analogového výstupu a argument *Value* jeho požadovanou hodnotu.

```
!Syntaxe
SetAO Signal Value;
!Příklad
SetAO ao3, 6.5;
```

- *SetDO*... slouží k nastavení hodnoty digitálního výstupu. Přičemž argument *Signal* definuje název konkrétního digitálního výstupu a argument *Value* jeho požadovanou hodnotu (logická 0 nebo 1).

```
!Syntaxe
SetDO Signal Value;
!Příklad
SetDO do5, 1;
```

- *PulseDO*... slouží k vygenerování pulzu na digitálním výstupu robota. Volitelným argumentem *PLength* lze nastavit délku tohoto pulzu v rozsahu od 0,001 do 2000 sekund. Pokud je tento argument vynechán, je použita jeho výchozí hodnota 0,2 sekundy. Argument *Signal* definuje název konkrétního digitálního výstupu.

```
!Syntaxe
PulseDO [\PLength] Signal;
!Příklad
PulseDO \PLength := 0.5, do8;
```

- *InvertDO*… invertuje hodnotu digitální výstupu (změna hodnoty logického signálu z 0 na 1 nebo z 1 na 0). Argumentem *Signal* je definován název výstupu, jehož hodnota má být invertována.

```
!Syntaxe
InvertDO Signal;
!Příklad
InvertDO do10;
```

- *WaitUntil*… slouží k pozastavení programu, dokud není zadaná logická podmínka splněna. Tu lze definovat pomocí argumentu *Cond*.

```
!Syntaxe
WaitUntil Cond;
!Příklad
VAR bool Hotovo;
WaitUntil Hotovo = TRUE;
```

- *WaitTime*… slouží k pozastavení chodu programu na definovanou hodnotu času, kterou lze nastavit pomocí argumentu *Time*. Časový údaj je zadáván v sekundách. Při použití volitelného argumentu *\InPos*, se čeká na úplné zastavení robotu a až poté je zahájeno počítání.

```
!Syntaxe
WaitTime [\InPos] Time;
!Příklad
WaitTime \InPos, 5;
```

- *WaitRob*… čeká na zastavení pohybu robota. Jednou z možností použití této instrukce je zadání volitelného argumentu *\ZeroSpeed*, kdy program pokračuje až po dosažení nulové rychlosti ramene.

```
!Syntaxe
WaitRob [\InPos] | [\ZeroSpeed];
!Příklad
!Až robot zastaví, nastav digitální výstup do5 na 0.
WaitRob \ZeroSpeed;
SetDO do5, 0;
```

- *WaitDI*… čeká, dokud nepřijde signál z digitálního vstupu. Přičemž argument *Signal* definuje název sledovaného vstupu a *Value* hodnotu, na kterou se čeká (logická 0 nebo 1). Totožným způsobem lze použít i instrukci čekající na digitální výstup – *WaitDO*.

```
!Syntaxe
WaitDI Signal, Value;
WaitDO Signal, Value;
!Příklad
WaitDI di3, 0;
WaitDO do2, 1;
```

- *WaitAI…* čeká, dokud hodnota analogového vstupu nedosáhne požadované hodnoty. Přičemž argument *Signal* definuje název konkrétního analogového vstupu a argument *Value* jeho požadovanou hodnotu. Volitelnými parametry lze také určit, zda instrukce čeká, dokud hodnota analogového signálu bude větší (*\GT*) než požadovaná hodnota *Value*, nebo menší (*\LT*). Stejným způsobem lze použít i instrukci čekající na hodnotu analogového výstupu – *WaitAO*.

```
!Syntaxe
WaitAI Signal [\LT] | [\GT] Value;
WaitAO Signal [\LT] | [\GT] Value;
!Příklady
WaitAI ai1, \LT, 3;
WaitAO ao1, \GT, 4;
```

4.5 INSTRUKCE PRO OVLÁDÁNÍ UCHOPOVACÍCH PRVKŮ A INSTRUKCE PRO KOMUNIKACI S PANELEM

Zde je uveden výběr běžně používaných instrukcí pro ovládání mechanických prstů a pneumatických přísavek. Součástí popisu instrukcí jsou i části kódu, pomocí kterých je demonstrována syntaxe a příklady použití. Jako první jsou popsány instrukce pro práci s mechanickými prsty.

- *g_Init…* slouží k inicializaci prstů efektoru, při které mohou být pomocí volitelných argumentů nastaveny parametry rychlosti (*\maxSpd*) a síly úchopu (*\holdForce)*. Pokud žádný z těchto parametrů není definován, jsou použity jejich výchozí hodnoty (rychlost 25 mm/s a uchopovací síla 20 N).

```
!Syntaxe
g_Init [\maxSpd] [\holdForce];
!Příklad
g_Init \maxSpd := 15 \holdForce := 10;
```

- *g_Calibrate*… slouží ke kalibraci prstů efektoru. Ta musí být provedena ještě před použitím pohybových nebo uchopovacích instrukcí efektoru.

```
!Příklad
g_Calibrate \Jog;
```

- *g_GripIn*… slouží k vykonání svíracího pohybu prstů efektoru. Pomocí volitelného argumentu *\holdForce* lze nastavit sílu uchopení. Instrukci je také možné rozšířit o argument *\NoWait*. V takovém případě program nečeká na úplné sevření prstů a již při svíracím pohybu začne vykonávat následující instrukce.

```
!Syntaxe
g_GripIn [\holdForce] [\NoWait];
!Příklady
g_GripIn \holdForce := 15;
g_GripIn \NoWait;
```

- *g_GripOut*… slouží k vykonání rozevíracího pohybu prstů efektoru. Podobně jako předchozí instrukce *g_GripIn* jde rozšířit o volitelné argumenty *\holdForce* a *\NoWait*.

```
!Syntaxe
g_GripOut [\holdForce] [\NoWait];
!Příklad
g_GripOut;
```

- *g_MoveTo*… slouží k nastavení definované vzdálenosti rozevření prstů efektoru. Požadovanou hodnotu lze nastavit argumentem *TargetPos*. Hodnota se musí pohybovat v intervalu od 0 do 25 mm. Jako předchozí instrukce lze doplnit o argument *\NoWait*.

```
!Syntaxe
g_MoveTo TargetPos [\NoWait];
!Příklad
g_MoveTo 15;
```

- *g_GetPos*… slouží k určení aktuální pozice rozevření prstů efektoru. Výstupní hodnota je udávána v milimetrech. V níže uvedeném příkladu je demonstrováno uložení aktuální pozice rozevření do proměnné *L_Pozice*.

```
!Příklad
VAR num L_Pozice;
L_Pozice := g_GetPos();
```

Následují instrukce pro ovládání pneumatických přísavek, které jsou rozšířeny o číslo přísavky. Existuje totiž typ nástroje, který je vybaven dvěma pneumatickými přísavkami. Pro správnou funkčnost programu je potřeba je od sebe odlišit.

- *g_VacuumOn[1/2]*… slouží k zapnutí sání vzduchu pneumatickou přísavkou.
- *g_VacuumOff[1/2]*… slouží k vypnutí sání vzduchu pneumatickou přísavkou.
- *g_BlowOn[1/2]*… slouží k zapnutí vyfukování vzduchu z pneumatické přísavky.
- *g_BlowOff[1/2*]… slouží k vypnutí vyfukování vzduchu z pneumatické přísavky.

```
!Příklady
g_VacuumOn1;
g_VacuumOff1;
g_BlowOn2;
g_BlowOff2;
```

Dále jsou popsány základní instrukce umožňující komunikaci mezi systémem robota a obsluhou, pomocí panelu FlexPendant. Součástí jsou i části kódu, ve kterých je ukázána syntaxe instrukcí a příklady jejich použití.

- *TPWrite*… zajistí zobrazení textu v operátorském okně panelu FlexPendant. Součástí textu mohou být i data různého datového typu, těm odpovídají volitelné argumenty *\Num*, *\Bool*, *\Pos*, *\Orient*, *\Dnum*.

```
!Syntaxe
TPWrite String[\Num]|[\Bool]|[\Pos]|[\Orient]|[\Dnum];
!Příklad
VAR num PocetKusu;
PocetKusu := 5;
TPWrite „Počet vyrobených kusů="\Num:=PocetKusu;
```

- *TPReadNum*… slouží k přečtení a uložení číselné hodnoty, kterou obsluha zadala v operátorském okně FlexPendant. Argumentem *TPAnswer* je proměnná typu *num*, do které se zadaná hodnota uloží. Pomocí argumentu *TPText* je v operátorském okně zobrazen doplňují text, například otázka, na kterou uživatel odpoví číselnou hodnotou.

```
!Syntaxe
TPReadNum TPAnswer TPText;
!Příklad
VAR num Pocet;
TPReadNum Pocet, „Kolik kusů má být vyrobeno?“;
```

- *TPErase…* vymaže text z operátorského okna.

```
!Syntaxe
TPErase;
```

5 REALIZACE ZADANÉ ÚLOHY

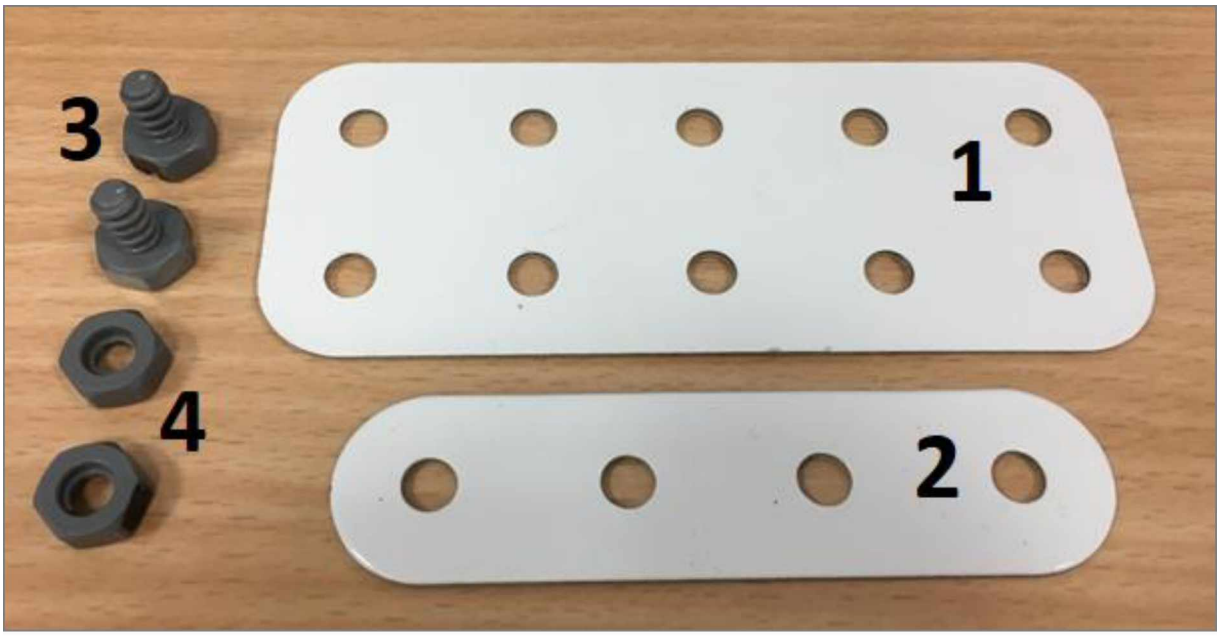
Jak již bylo v úvodu práce zmíněno, hlavním cílem diplomové práce je realizace montážní operace. Ta se skládá z činností pro podání dílů a spojovacího materiálu, sešroubování dílů do jednoho kompletu a odložení hotového výrobku. Součástí je i návrh a výroba mechanických přípravků potřebných pro montáž. V rámci úlohy se také mají ověřit možnosti využití strojového vidění pro lokalizaci spojovacích prvků. Pro tyto účely práce byl využit dvouramenný kooperující robot ABB YuMi.

Řešení práce bylo realizováno na robotickém pracovišti, nacházejícím se v jedné z laboratoří fakulty Elektrotechniky a informatiky na univerzitě v Pardubicích. Robotické pracoviště je ukázáno na obrázku 5.1.

Obrázek 5.1 – Robotické pracoviště

5.1 VYUŽITÉ SOUČÁSTKY A NÁVRH MECHANICKÝCH PŘÍPRAVKŮ

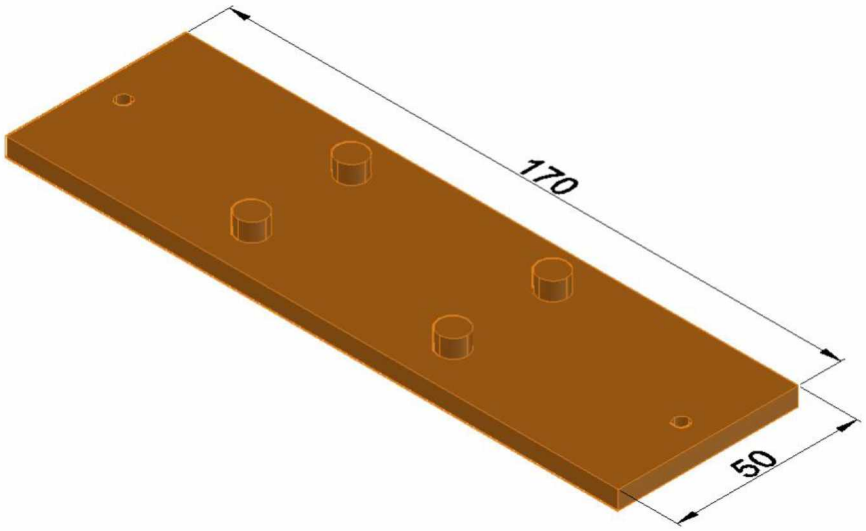
Výrobek byl zvolen tak, aby jeho součástky byly z lehkých materiálů a bez ostrých hran. Bylo využito dílů stavebnice ROTO Abc, které jsou vyrobeny z tvrdého kartonu a plastových šroubů a matic. Na obrázku 5.2 jsou vybrané součástky ukázány a popsány pomocí následujícího číselného označení: velký díl (1), malý díl (2), šrouby (3), matice (4). V rámci montážní operace dojde ke spojení malého a velkého dílu pomocí dvou šroubových spojení.



Obrázek 5.2 – Součástky výrobku

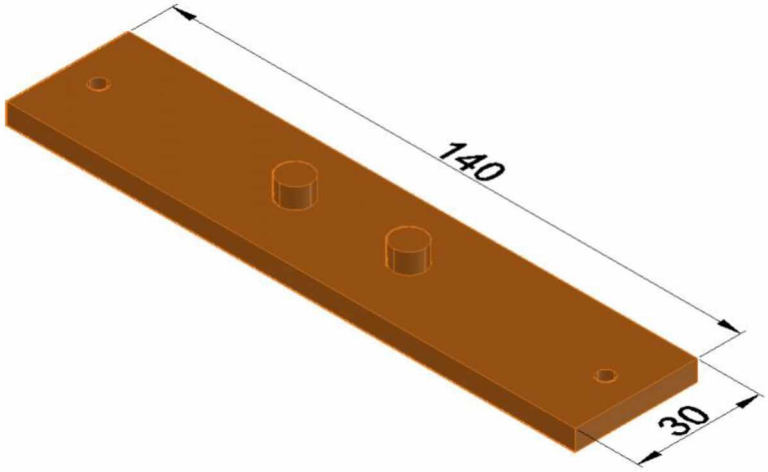
Na základě rozměrů vybraných dílů a spojovacího materiálu byly v software Autodesk AutoCAD navrženy modely mechanických přípravků. Ty byly následně vytištěny pomocí 3D tiskárny a připevněny na dvě plechové paletky. Navržené mechanické přípravky v první řadě slouží k určení stálých pozic všech použitých součástek potřebných pro sestavení jednoho výrobku. Tyto pozice musí být určeny z důvodu přesného uchopení pomocí nástroje robota. Zvláštní mechanické přípravky musely být vyrobeny i pro realizaci samotné šroubovací operace.

Na obrázku 5.3 je znázorněn model mechanického přípravku pro uložení velkého dílu. Díl je nasazen na čtyři stabilizační kolíky, jejichž rozměry odpovídají otvorům v dílu. Uložením velkého dílu na stabilizační kolíky je zajištěna jeho přesná pozice pro odebrání nástrojem robota.



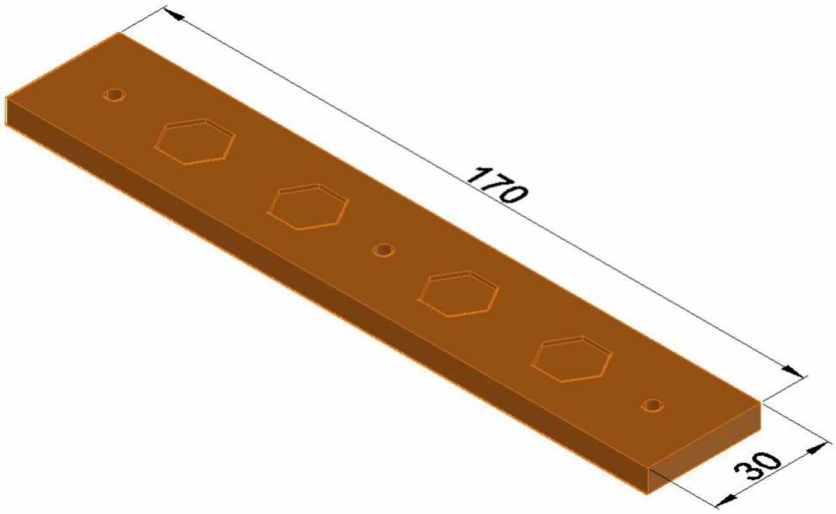
Obrázek 5.3 – Model mechanického přípravku pro založení velkého dílu

Model mechanického přípravku pro založení malého dílu je ukázán na obrázku 5.4. Přesná pozice dílu, pro odebrání nástrojem robota, je opět určena pomocí stabilizačních kolíků, jejichž rozměry jsou shodné s otvory v dílu.



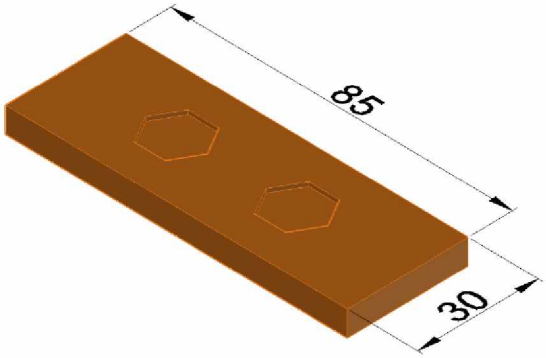
Obrázek 5.4 – Model mechanického přípravku pro založení malého dílu

Pro založení spojovacího materiálu, tedy šroubů a matic, byl navržen mechanický přípravek uvedený na obrázku 5.5. Pozice jsou určeny pomocí 1 mm hlubokých drážek ve tvaru matice a hlavy šroubu. Po umístění spojovacího materiálu do těchto drážek je určena jejich přesná pozice pro odebrání nástrojem robota.



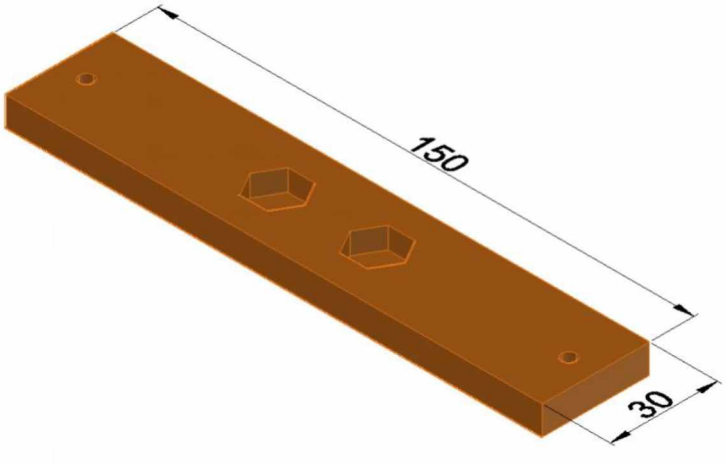
Obrázek 5.5 – Model mechanického přípravku pro založení matic a šroubů

Na stejném principu funguje i mechanický přípravek pro předání matic z pracovního prostoru pravého ramene do pracovního prostoru levého ramene. Přesné pozice jsou znovu určeny díky 1 mm hlubokým drážkám o rozměrech matice. Model tohoto mechanického přípravku je znázorněn na obrázku 5.6.



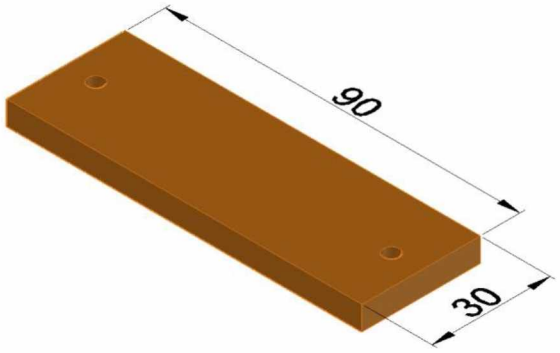
Obrázek 5.6 – Model mechanického přípravku pro předání matic

Na obrázku 5.7 je ukázán model mechanického přípravku, jenž slouží ke stabilizaci šroubů při šroubovací operaci. Šrouby jsou zde založeny do 5 mm hlubokých prohlubní ve tvaru hlavy šroubu. Těmito prohlubněmi je zajištěna stálá pozice šroubů při šroubování matic a nedochází tak k jejich protáčení.



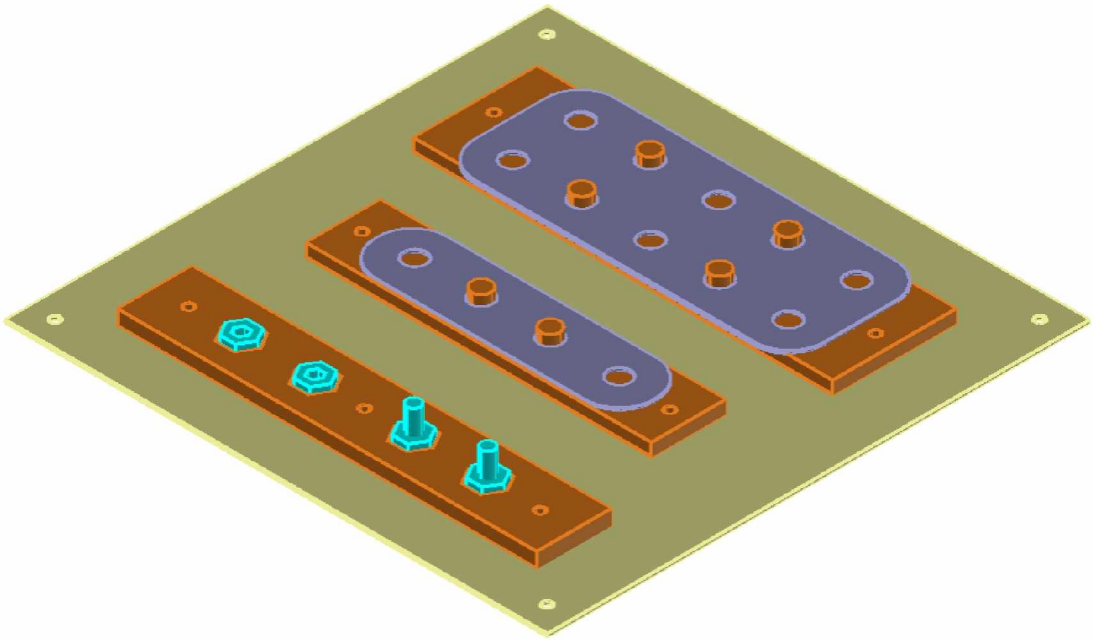
Obrázek 5.7 – Model mechanického přípravku pro stabilizaci šroubů

Posledním navrženým modelem je model uvedený na obrázku 5.8. Jde o distanční podložku, na kterou je pro vykonání šroubovací operace položen velký díl. Díky tomu se malý i velký díl nachází ve stejné rovině a může tak dojít k jejich korektnímu spojení.



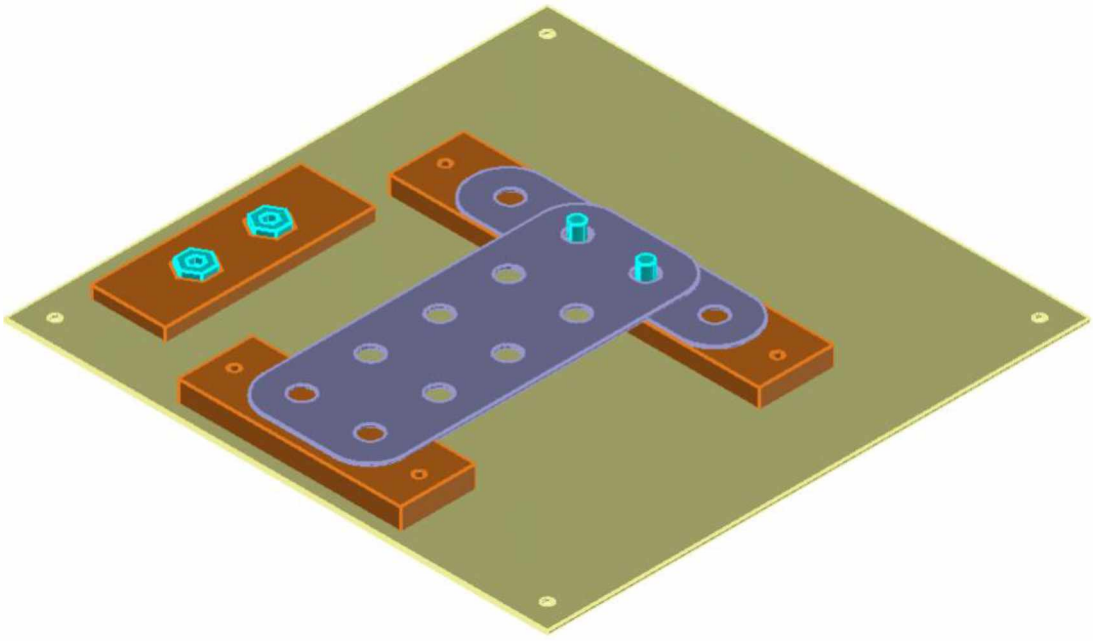
Obrázek 5.8 – Model mechanického přípravku pro podložení dílů

Jak již bylo zmíněno, tak popsané mechanické přípravky jsou připevněny na dvě plechové paletky o rozměrech 220x220 mm, které jsou přimontovány ke stolu v pracovním prostoru robota. Pro demonstraci umístění mechanických přípravků na paletkách, včetně založených součástek, byly vytvořeny modely obou paletek. Z důvodu srozumitelnějšího popisu jsou paletky, zde i dále v textu, označovány jako „paletka č.1“ a „paletka č. 2“. Na paletku č. 1 jsou přišroubovány tři mechanické přípravky, které slouží k založení součástek obsluhou a určení jejich stálých pozic pro odebrání robotem. Jde o přípravky pro založení spojovacího materiálu, a malého a velkého dílu. Model paletky č. 1, včetně založených součástek, je znázorněn na obrázku 5.9.



Obrázek 5.9 – Model paletky č. 1 včetně založených součástek

Součástky z paletky č. 1 jsou v rámci vykonávání úlohy robotem přemístěny na paletku č. 2, kde je následně realizována samotná šroubovací operace. Cílové pozice součástek pro přemístění jsou určeny pomocí zbylých mechanických přípravků. Jde o přípravky pro předání matic, uložení šroubů pro jejich stabilizaci při šroubování matic a distanční podložku pro podložení velkého dílu. Paletka č. 2 včetně založených součástek, po jejich přemístění z paletky č. 1, je ukázána na obrázku 5.10.



Obrázek 5.10 – Model paletky č.2 včetně založených součástek

5.2 ŠROUBOVÉ SPOJENÍ S VYUŽITÍM ROBOTA ABB YUMI

Stěžejním krokem celé aplikace je činnost šroubování, pro kterou bylo při tvorbě programu vyzkoušeno několik různých přístupů, které jsou v následujícím textu komentovány. Sama společnost ABB tvrdí, že dvouramenný robot YuMi je navržen především pro vykonávání lehkých montážních operací. Při tvorbě části programu pro šroubovací operaci se ukázalo jako nepraktické, že rameno neumožňuje otáčení koncovým nástrojem v neomezeném rozsahu, jako tomu je například u kolaborativního robota UR3 od společnosti Universal Robots. Pracovní rozsah šesté osy, tedy osy, která zajišťuje rotační pohyb nástroje, je od −229° do +229°. To znamená, že při šroubování dlouhých závitů je nutné, aby rameno třeba i několikrát pozastavilo proces šroubování, vrátilo se zpět na začátek pracovního rozsahu a až poté znovu pokračovalo v rotačním pohybu. Zároveň je ale potřeba zmínit, že se nabízí možnost spolupráce obou ramen, kdy například jedno rameno uchopí šroub, druhé matici a šroubovací pohyb vykonávají proti sobě, čímž je dosaženo dvojnásobného pracovního rozsahu. Ne vždy je ale tento způsob řešení možný. Jako například v řešení této diplomové práce, kde jsou šrouby založeny do připevněného mechanického přípravku.

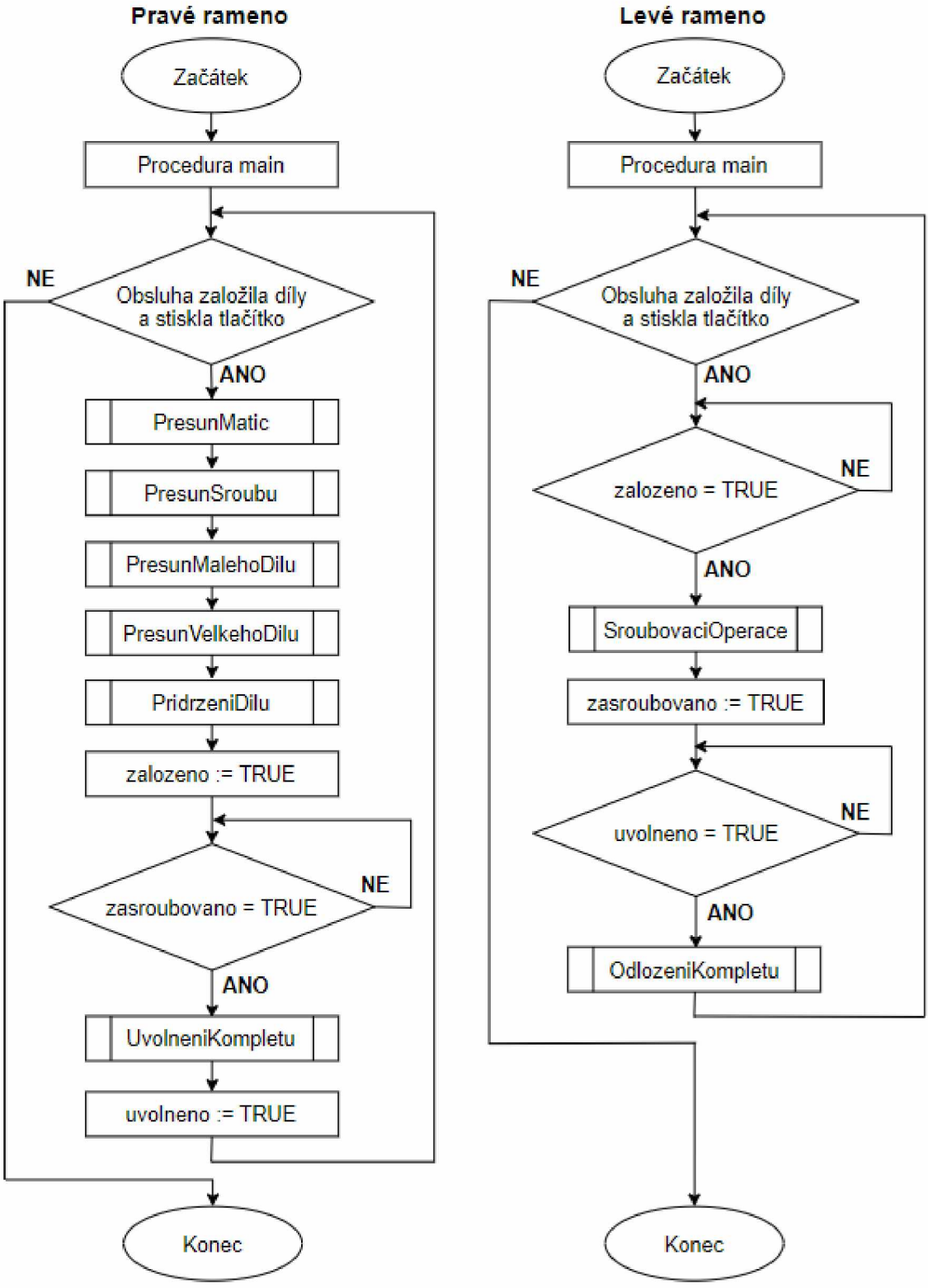
Při tvorbě programu šroubovací operace bylo jako první otestováno použití softwarového prostředku *SoftMove*, který slouží ke snížení tuhosti ramene robota v definovaném směru, zatím co v ostatních směrech jsou zachovány původní vlastnosti. Předpokládalo se, že *SoftMove* zajistí vyšší vůli nástroje pro správné najetí matice na šroub a následné chycení závitu při rotačním pohybu. Nicméně ukázalo se, že s funkcí *SoftMove* se znatelně zhorší přesnost pohybů nástroje, která je pro tuto operaci nezbytnou a šroubové spojení tak nebylo možné spolehlivě realizovat. Dále bylo testováno využití instrukce *ContactL*, pomocí které je nástroji umožněno tlačit na předmět při vykonávání lineárního pohybu a vyhnout se tak chybovému hlášení spojeného s kolizí. Při tlačení je měřena aktuální velikost kroutícího momentu motorů a porovnávána s požadovanou hodnotou uživatele. Po dosažení požadované hodnoty, tedy požadované míry přitlačení, je činnost instrukce ukončena. Tento proces byl v rámci úlohy vyzkoušen pro najetí nástroje s maticí na závit šroubu, kde se nástroj s maticí přesunul nad pozici šroubu a začal proti němu vykonávat lineární pohyb. V okamžik, kdy matice narazila na šroub, tak byla požadovanou silou natlačena na jeho závit a mohlo se zahájit vykonávání rotačního pohybu pro šroubování. Tento postup by ale našel spíše uplatnění u aplikace, kde se mění délka použitých šroubů. Díky tomu by bylo zaručeno, že nástroj s maticí najede na závit, jakkoliv dlouhého šroubu. Nicméně v úloze této diplomové je v každém cyklu použit stejný typ šroubu a zároveň je díky navrženým mechanickým přípravkům zajištěno, že

jsou založeny vždy stejným způsobem. Proto se jako nejspolehlivější ukázalo řešení s použitím pevně definované pozice pro nasazení matice na závit šroubu, odkud je následně zahájeno samotné šroubování. Pro vykonání šroubovacího pohybu byla vybrána funkce *RelTool*, jenž je pro podobné účely přímo určena. S jejím využitím je totiž možné realizovat posuvný a rotační pohyb zároveň. Parametry posuvu a rotace byly určeny na základě velikosti stoupání závitu použitých šroubů. Použití této funkce sebou ale může přinést jeden nežádoucí problém. V případě, kdy je zadán vyšší rozsah rotačního pohybu, může nastat situace, kdy robot neví, jakým směrem má cílové pozice dosáhnout. Výsledkem toho je chybové hlášení, které sděluje, že požadované pozice není možné dosáhnout a chod programu je ukončen. Problém spočívá v odlišné konfiguraci os aktuální a cílové pozice. Jedním ze způsobů řešení je rozdělení rotačního pohybu na více dílčích úseků – tento postup byl využit i při realizaci úlohy této diplomové práce. Další možností je použití instrukce *ConfL*, pomocí které lze dočasně vypnout monitorování konfigurace os při vykonávání pohybu ramene.

Při tvorbě programu bylo uvažováno i nad přístupy pro utahování závitu. V případě jako tento, kdy je známá délka použitých šroubů, by bylo možné dotahování realizovat na základě pevně stanoveného počtu otáček nástroje pro dostatečné dotažení. I přesto, že výsledky tohoto postupu nebyly špatné, byla nakonec vybrána jiná z možností. Jako spolehlivější se ukázalo řešení s použitím funkce *GetMotorTorque*, pomocí které lze získat aktuální hodnotu kroutícího momentu motoru libovolné osy ramene a tu následně porovnávat s požadovanou hodnotou pro dotažení. Pomocí toho je zajištěno, že je šroubový spoj vždy dostatečně dotažen.

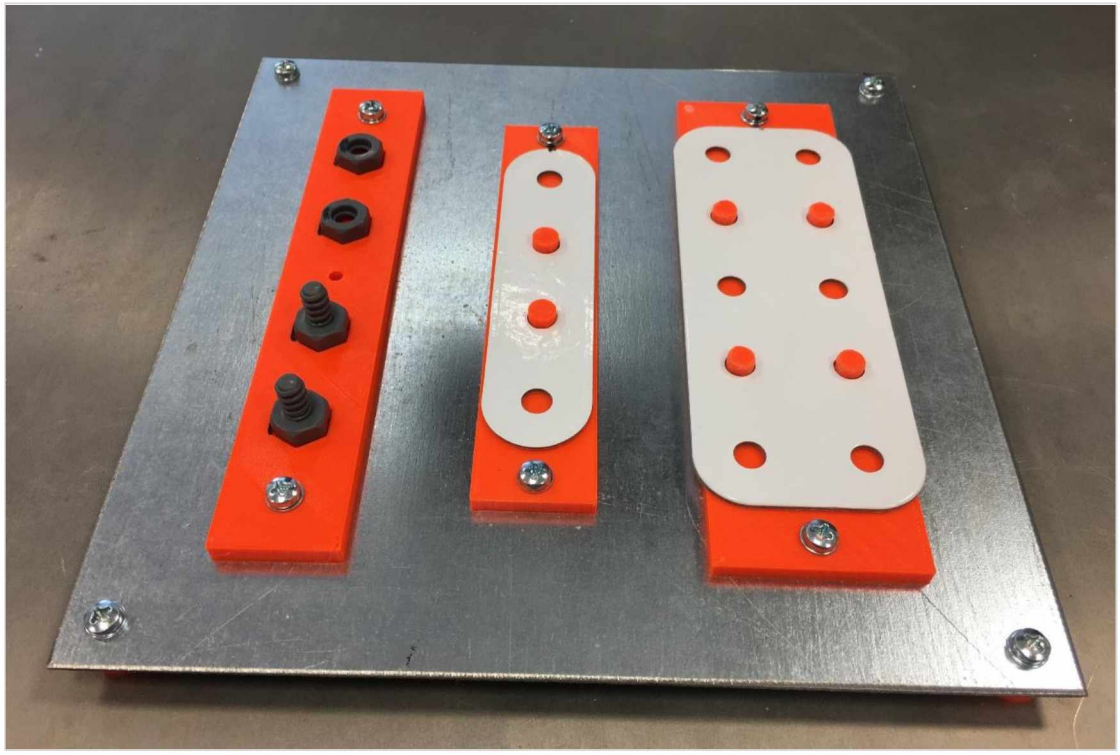
5.3 POPIS MONTÁŽNÍ OPERACE A STRUKTURY PROGRAMU

Jak již bylo v pododdílu 2.5.1 vysvětleno, ramena kooperujícího robota YuMi pracují nezávisle na sobě a při programování je k nim přistupováno jako ke dvěma samostatným robotům. Z toho důvodu byly v rámci zadané úlohy vytvořeny dva programy – pro levé a pravé rameno. Každému programu odpovídá jeden programový modul. Moduly jsou rozděleny do několika procedur, přičemž každá z nich zajišťuje konkrétní činnost ramene. Na obrázku 5.11 jsou uvedeny vývojové diagramy, které demonstrují posloupnost volání procedur obou robotických ramen. V následujícím textu jsou na základě těchto diagramů komentovány kroky montážní operace. Souběžně s tím je popsána i struktura vytvořeného programu.



Obrázek 5.11 – Vývojové diagramy programů pravého a levého ramene

Po spuštění se jako první volají procedury *main* obou robotických ramen, ze kterých je řízeno volání podřadných procedur. Program nejdříve čeká, než obsluha založí potřebné díly na paletku č. 1, jak je tomu ukázáno na obrázku 5.12 a stiskne tlačítko pro zahájení montážního cyklu. Po jeho stisknutí ramena najedou do výchozích pozic, kde dojde k inicializaci a kalibraci nástrojů a resetování hodnot níže popsaných proměnných.

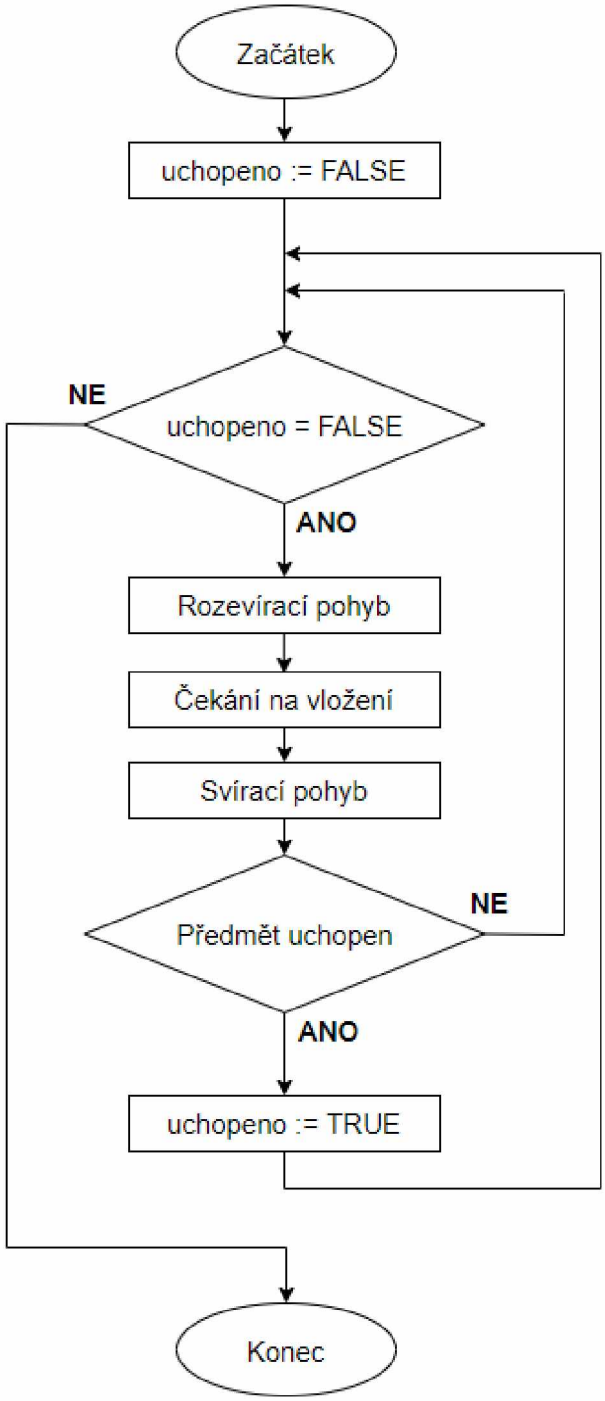


Obrázek 5.12 – Paletka č. 1 se založenými díly

Z procedur *main* jsou dále postupně volány dílčí procedury, které již zajišťují konkrétní činnosti obou ramen. Posloupnost volání těchto procedur je synchronizována pomocí persistentních proměnných, které byly definovány pro oba programy společně. Mezi ty patří logické proměnné *zalozeno*, *zasroubovano*, *uvolneno*. Právě těmto proměnným je na začátku každého cyklu programu přiřazena hodnota *FALSE*. Jejich konkrétní souvislost se synchronizací chodu programu je vysvětlena dále v textu.

Jak je ve vývojových diagramech znázorněno, pravé rameno může s vykonáváním svých procedur začít ihned po stisknutí tlačítka. Naopak levé rameno čeká na změnu proměnné *zalozeno*. Volá se tedy procedura pravého ramene *PresunMatic*. V rámci ní dojde k přemístění obou matic z paletky č. 1 na paletku č. 2, kde jsou pro ně určené pozice na mechanickém přípravku, který je určen pro odložení matic v pracovním prostoru levého ramene. K přemístění matic jsou použity prsty efektoru.

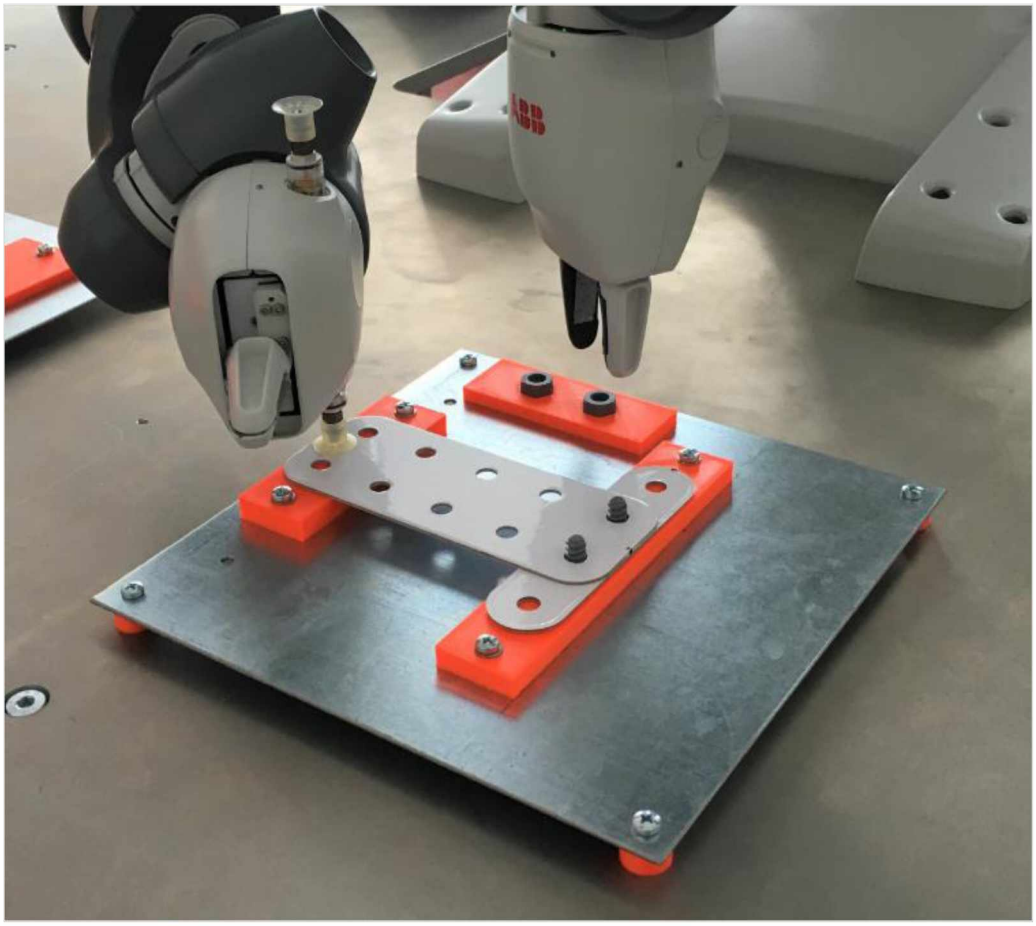
Pro každou uchopovací činnost prstů, kdekoliv v programu, se volá procedura *Uchopit*. Procedura je zajímavá především proto, že je v rámci ní ošetřeno i neúspěšné uchopení předmětu. Použití této procedury je v programu poměrně časté, z toho důvodu není součástí vývojových diagramů na obrázku 5.11, čímž je zajištěna jejich vyšší přehlednost. Její funkce je ale popsána pomocí samostatného vývojového diagramu na obrázku 5.13. a následujícího textu.



Obrázek 5.13 – Vývojový diagram procedury Uchopit

Z diagramu je zřejmé, že pokud prsty při svíracím pohybu neuchopí žádný předmět, tak se znovu otevřou, po definovanou dobu se vyčkává na vložení předmětu obsluhou a následně opět dochází ke svíracímu pohybu. Tímto způsobem se v cyklu čeká na korektní uchopení předmětu. Po úspěšném uchopení se mění hodnota proměnné *uchopeno* a činnost procedury končí. Kód programu pokračuje instrukcí, která následuje volání této procedury.

Po přesunu matic je jako další volána procedura *PresunSroubu*, pomocí které jsou přemístěny oba šrouby z paletky č. 1 na paletku č. 2. Na paletce č. 2 jsou odloženy na mechanický přípravek, zajišťující jejich stabilitu při montážní operaci. Pro přesun matic jsou znovu použity prsty efektoru. Následují procedury *PresunMalehoDilu* a *PresunVelkehoDilu*. V rámci nich dojde k postupnému přemístění obou dílů z paletky č. 1 na paletku č. 2, kde jsou nasazeny na již uložené šrouby. Jako první je přesunut malý díl a na něj je poté umístěn díl velký. K uchopení dílů je použita pneumatická přísavka. Další úlohou pravého ramene je procedura *PridrzeniDilu*. V rámci ní je s využitím pneumatické přísavky velký díl přitlačen k distanční podložce, kterou je díl podložen – tím je dosaženo vyšší stability obou dílů pro šroubovací operaci. Pro přehlednost je tato situace znázorněna na obrázku 5.14.

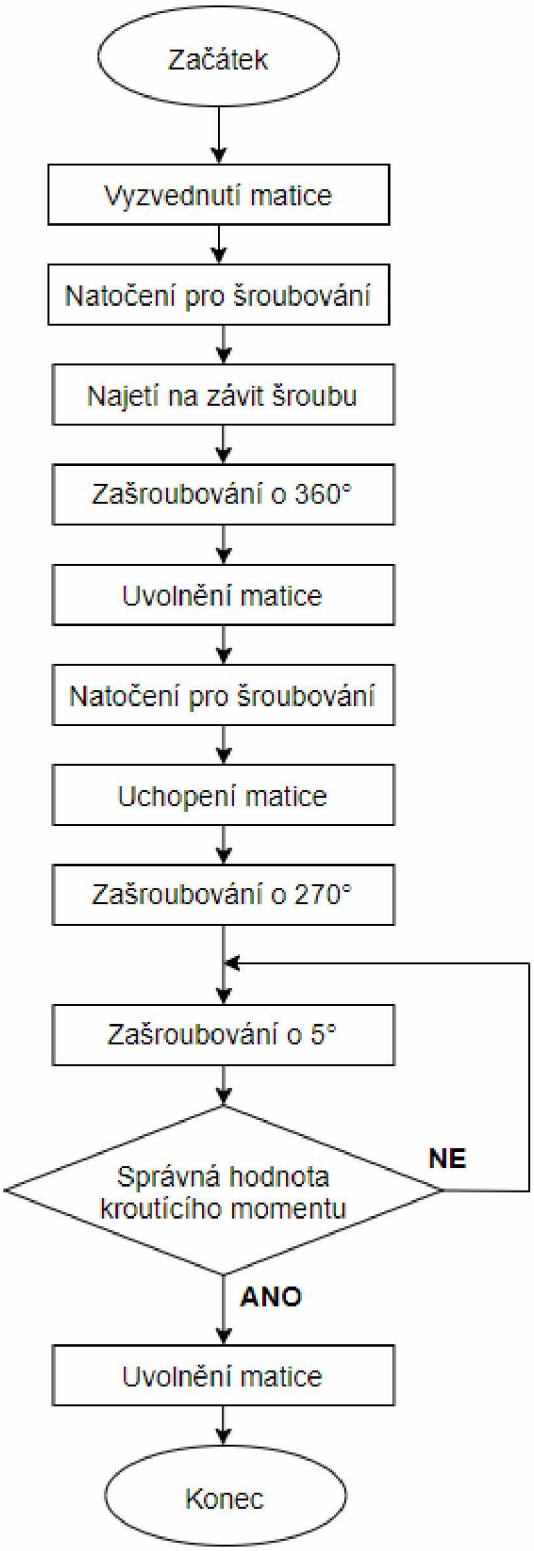


Obrázek 5.14 – Situace na paletce č. 2 před šroubovací operací

V momentě, kdy se pravé rameno nachází v této pozici, je změněna hodnota proměnné *zalozeno*. Tím je poskytnuta informace o tom, že veškeré díly jsou založeny a pravé rameno se nachází v pozici pro přidržení dílů. Levé rameno tak může zahájit svou činnost. Naopak pravé rameno v pozici pro přidržení setrvává, dokud není změněna hodnota proměnné *zasroubovano*.

Jak je ve vývojovém diagramu na obrázku 5.11 ukázáno, první činností levého ramene je *SroubovaciOperace*. V rámci ní jsou matice s využitím prstů efektoru zašroubovány na již připravené šrouby, čímž dojde ke spojení malého a velkého dílu. Tato procedura je v porovnání s ostatními poněkud obsáhlejší a skládá se z více dílčích úkonů. Z toho důvodu pro ni byl vytvořen samostatný vývojový diagram, uvedený na obrázku 5.15. Pomocí něj a následujícího textu jsou popsány kroky pro zašroubování jedné z matic, pro druhou matici je postup totožný.

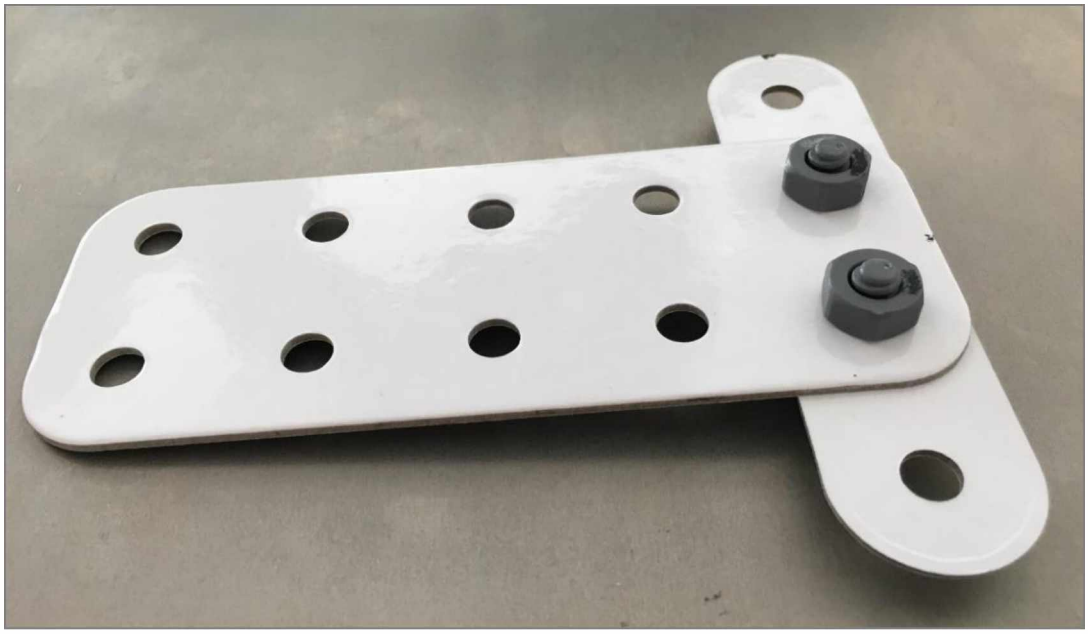
Činnost procedury začíná vyzvednutím matice z mechanického přípravku, na který byla přesunuta pravým ramenem. Následně dojde k natočení nástroje do pozice vhodné pro šroubování. Robot YuMi totiž umí vykonat rotační pohyb nástroje pouze v rozsahu od −229° do +229°. Z toho důvodu je potřeba nástroj natočit tak, aby se vytvořil dostatečný prostor pro vykonání rotačního pohybu. Matice je poté umístěna na začátek závitu šroubu. Tato pozice je neměnná, a proto může být pevně definována. Pomocí mechanických přípravků je totiž zajištěno vždy stejné založení obou šroubů. V tento okamžik se zahajuje samotné šroubování, které je rozděleno do několika dílčích kroků. V prvním z nich se provede zašroubování matice o 360°. Poté je šroubování pozastaveno, přičemž efektor uvolní matici a vrátí se zpět do pozice, ze které mu bude umožněno znovu vykonávat rotační pohyb pro šroubování, aby se nedostal na konec rozsahu otočení. Následně je matice znovu uchopena a zašroubována o dalších 270°. Tím se matice dostane téměř ke konci závitu šroubu a zahájí se proces utahování. Při něm se měří hodnota aktuálního kroutícího momentu nástroje a porovnává se s definovanou hodnotou pro utažení. Pokud je aktuální hodnota nižší než požadovaná, dojde k utažení matice o 5°. Tento úkon se v cyklu opakuje, dokud není dosaženo požadované hodnoty kroutícího momentu. Matice je poté považována za dostatečně dotaženou a může být uvolněna. Stejným způsobem je následně zašroubována i druhá matice.



Obrázek 5.15 – Vývojový diagram procedury SroubovaciOperace

Z hlavního vývojového diagramu na obrázku 5.11 je zřejmé, že po dokončení šroubovací operace se změní hodnota proměnné *zasroubovano*. Tím je poskytnuta informace

o tom, že díly byly sešroubovány a pravé rameno tak může zkompletovaný výrobek uvolnit. Po jeho uvolnění se mění hodnota proměnné *uvolneno*. To je naopak signál pro levé rameno, že zkompletovaný výrobek může pomocí pneumatické přísavky vyjmout a odložit na určené místo, jak je tomu ukázáno na obrázku 5.16. K tomuto účelu slouží procedura *OdlozeniKompletu*. Program se poté vrací znovu na začátek, kde čeká na opětovné založení dílů a stisknutí tlačítka.



Obrázek 5.16 – Zkompletovaný výrobek

5.4 PROCEDURY PROGRAMU PRAVÉHO RAMENE

Částečně byly funkce a činnosti procedur vysvětleny již v oddílu 5.3, který popisuje strukturu vytvořeného programu. V oddílech 5.4 a 5.5 je uveden rozbor kódů procedur pravého a levého ramene. Díky tomu je ukázáno programové řešení již popsaných činností zadané úlohy. Součástí rozboru nejsou detailní informace ke všem použitým instrukcím a nástrojům jazyka. Pro tento účel byla napsána kapitola 4, která se zaměřuje výhradně na problematiku spojenou s programovacím jazykem RAPID a lze tam veškeré doplňující informace dohledat. Následuje rozbor procedur z programu pravého ramene.

Základním kamenem programu je procedura *main*, jejíž kód je uveden níže v textu. V ní se jako první instrukcí *SetDO* nastaví hodnota digitálního výstupu *do0* na logickou 1 a instrukcí *WaitDI* se čeká na logickou 1 digitálního vstupu *di0*. Mezi tento digitální výstup a vstup robota je totiž zapojeno tlačítko, které slouží k zahájení chodu cyklu programu. Po jeho stisknutí rameno, s využitím pohybové instrukce *MoveJ*, najede do výchozí pozice *pstart*. Důležitým

argumentem této pohybové instrukce je mimo jiné i typ zvoleného nástroje. V tomto případě jde o nástroj s označením *Servo*. Tímto způsobem jsou v technické dokumentaci i v software RobotStudio označovány mechanické prsty. Pro přehlednost byl tento název zachován. Následně dojde k přiřazení hodnoty *FALSE* persistentním proměnným *zalozeno*, *zasroubovano* a *uvolneno*. Jak již bylo u popisu struktury programu zmíněno, tyto proměnné jsou definovány společně pro oba programy a je pomocí nich řízena posloupnost vykonávání činností obou ramen. Instrukce *g_Init* poté zajistí inicializaci prstů efektoru, kde je argumentem *\holdForce* nastavena síla pro uchopení předmětů a instrukcí *g_Calibrate \Jog* je provedena jejich kalibrace. Následuje postupné volání procedur *PresunMatic*, *PresunSroubu*, *PresunMalehoDilu*, *PresunVelkehoDilu*, *PridrzeniDilu*. Po vykonání procedury *PridrzeniDilu* je změněna hodnota proměnné *zalozeno*. Tím je poskytnuta informace o tom, že všechny předchozí procedury byly úspěšně vykonány. Na tuto informaci ve svém programu čeká levé rameno, které následně zahajuje některou ze svých činností. Pravé rameno nyní pomocí instrukce *WaitUntil* čeká na změnu hodnoty proměnné *zasroubovano*, ta je změněna v programu levého ramene. V okamžik, kdy se proměnná *zasroubovano* rovná hodnotě *TRUE*, pravé rameno pokračuje procedurou *UvolneniKompletu*. Poté, co je procedura *UvolneniKompletu* vykonána, rameno najíždí zpět do výchozí pozice *pstart* a mění se hodnota proměnné *uvolneno*. Tím je znovu poskytnuta informace levému rameni o tom, že procedura *UvolneniKompletu* byla vykonána, a to na to reaguje svou činností.

```
PROC main()
     SetDO do0, 1;
     WaitDI di0, 1;
     MoveJ pstart,v500,z50,Servo;
     zalozeno:=FALSE;
     zasroubovano:=FALSE;
     uvolneno:=FALSE;
     g_Init \holdForce:=8;
     g_Calibrate \Jog;
     PresunMatic;
     PresunSroubu;
     PresunMalehoDilu;
     PresunVelkehoDilu;
     PridrzeniDilu;
     zalozeno:=TRUE;
     WaitUntil zasroubovano=TRUE;
     UvolneniKompletu;
     MoveJ pstart,v500,z50,Servo;
     uvolneno:=TRUE;
ENDPROC
```

Následuje popis procedury *PresunMatic*. V rámci této procedury jsou s využitím prstů efektoru přesunuty obě matice z paletky č. 1 na paletku č. 2. V přiložené části kódu je demonstrováno přesunutí jedné z nich. Pro druhou matici je kód téměř totožný, liší se pouze názvy pozic. Pro realizaci přesunu byly deklarovány pozice pro vyzvednutí matice *pmat1*, odložení matice *pmat1k* a pomocná pozice *pp*, kterou nástroj musí projet při pohybu z jedné paletky na druhou. Pohyb je uskutečněn sekvencí několika pohybových instrukcí *MoveJ* a *MoveL*, které jsou v některých případech rozšířeny o funkci *Offs*. Pro rozevření prstů je použita instrukce *g_MoveTo*, pomocí které jsou prsty rozevřeny do definované pozice. Pro uchopovací pohyb je volána procedura *Uchopit*, jejíž kód je podrobněji popsán až dále v textu.

```
PROC PresunMatic()
     !První matice
     g_MoveTo 12,\NoWait;
     MoveJ Offs(pmat1,0,0,30),v500,fine,Servo;
     MoveL pmat1,v100,fine,Servo;
     Uchopit;
     MoveL Offs(pmat1,0,0,20),v300,z50,Servo;
     MoveJ pp,v500,z100,Servo;
     MoveJ Offs(pmat1k,0,0,20),v500,fine,Servo;
     MoveL pmat1k,v30,fine,Servo;
     g_MoveTo 12;
     MoveL Offs(pmat1k,0,0,20),v500,fine,Servo;
     MoveJ pp,v500,z50,Servo;
     !Druhá matice
     …
ENDPROC
```

Pro každý uchopovací pohyb prstů efektoru je volána procedura *Uchopit*, v rámci ni je navíc ošetřena i situace neúspěšného uchopení předmětu. Její kód je přiložen níže v textu. Funkce této procedury byla již popsána v oddílu 5.3, a to pomocí vývojového diagramu na obrázku 5.13. Procedura pomocí *WHILE* cyklu čeká na správné vložení předmětu obsluhou a jeho uchopení. Pro rozevření prstů je znovu použita instrukce *g_MoveTo* a pro sevření instrukce *g_GripIn*. Časovou prodlevu mezi těmito dvěma pohyby zajišťuje instrukce *WaitTime*. Uchopení předmětu je detekováno funkcí *g_GetPos*. V případě, kdy je předmět správně uchopen, je změněna hodnota proměnné *uchopeno* a činnost procedury končí.

```
PROC uchopit()
     uchopeno:=FALSE;
     WHILE uchopeno=FALSE DO
          g_MoveTo 12;
          WaitTime 1;
          g_GripIn;
```

```
            IF g_GetPos()>=1 THEN
                uchopeno:=TRUE;
            ENDIF
        ENDWHILE
ENDPROC
```

Další procedurou pravého ramene je procedura *PresunSroubu*, pomocí které jsou oba šrouby přesunuty z paletky č. 1 na paletku č. 2. Kód pro přesunutí jednoho ze šroubů je uveden níže v textu, pro přesun druhého šroubu je kód téměř totožný, liší se pouze názvy pozic. K přemístění jsou znovu použity prsty efektoru, k jejichž ovládání slouží instrukce *g_MoveTo* a již představená procedura *Uchopit*. Přesun je realizován sekvencí pohybových instrukcí *MoveJ* a *MoveL* s doplněním o funkci *Offs*, pomocí kterých je šroub vyzvednut z pozice *psroub1* a odložen na pozici *psroub1k*. Pro kontrolovaný pohyb mezi paletkami je znovu použita pomocná pozice *pp*.

```
PROC PresunSroubu()
    !První šroub
    g_MoveTo 12,\NoWait;
    MoveJ Offs(psroub1,0,0,30),v500,fine,Servo;
    MoveL psroub1,v100,fine,Servo;
    Uchopit;
    MoveL Offs(psroub1,0,0,10),v300,z50,Servo;
    MoveJ pp,v500,z100,Servo;
    MoveJ Offs(psroub1k,0,0,30),v500,fine,Servo;
    MoveL psroub1k,v30,fine,Servo;
    g_MoveTo 12;
    MoveJ Offs(psroub1k,0,0,30),v300,z50,Servo;
    MoveJ pp,v500,z50,Servo;
    !Druhý šroub
    …
ENDPROC
```

V posloupnosti programu následují procedury pro přesun malého a velkého dílu z paletky č. 1 na paletku č. 2. V rámci tohoto popisu je komentován pouze kód procedury pro přesun malého dílu (*PresunMalehoDilu*), který je níže uveden. Kód procedury pro přesun velkého dílu je téměř totožný a opět se liší pouze názvy pozic. Pro sekvenci pohybových instrukcí *MoveJ* a *MoveL*, díky kterým je přesun realizován, byly deklarovány pozice pro vyzvednutí dílu *pdil1*, odložení dílu *pdil1k* a pomocná pozice *ppdil*, která opět slouží k zajištění kontrolovaného pohybu mezi paletkami. Díl je uchopen pneumatickou přísavkou, která je v technické dokumentaci a software RobotStudio označována jako *VaccumOne*. Spuštění nasávání vzduchu, a tedy i uchopení dílu, zajišťuje instrukce *g_VacuumOn1*. Naopak k vypnutí

sání a uvolnění předmětu slouží instrukce *g_VacuumOff1*. Tyto činnosti jsou doplněny o instrukci pro čekání *WaitTime*, pomocí které je zajištěna dostatečná časová prodleva pro spolehlivé uchopení a odložení dílu, ještě před tím, než se zahájí vykonávání pohybu ramene.

```
PROC PresunMalehoDilu()
     MoveJ Offs(pdil1,0,0,40),v300,z10,VaccumOne;
     MoveL pdil1,v20,fine,VaccumOne;
     g_VacuumOn1;
     WaitTime 0.5;
     MoveL Offs(pdil1,0,0,30),v20,z10,VaccumOne;
     MoveJ ppdil,v500,z100,VaccumOne;
     MoveJ Offs(pdil1k,0,0,40),v300,z10,VaccumOne;
     MoveL pdil1k,v20,fine,VaccumOne;
     g_VacuumOff1;
     WaitTime 0.5;
     MoveL Offs(pdil1k,0,0,40),v100,z10,VaccumOne;
     MoveJ ppdil,v500,z100,VaccumOne;
ENDPROC
```

Následuje procedura *PridrzeniDilu*, jejíž kód je ukázán dále v textu. Úkolem procedury je zajistit stabilitu obou dílů při montážní operaci levého ramene. Toho je dosaženo přitlačením velkého dílu k podložce pomocí pneumatické přísavky. Nejdřív jsou instrukcí *g_MoveTo* sevřeny prsty efektoru, aby nepřekážely v pohybu. Poté nástroj pomocí pohybových instrukcí *MoveJ* a *MoveL* najede do pevně definované pozice *ppridrz*, ze které jsou díly přitlačeny.

```
PROC PridrzeniDilu()
     g_MoveTo 0,\NoWait;
     MoveJ Offs(ppridrz,0,0,15),v300,z30,VaccumOne;
     MoveL ppridrz,v50,fine,VaccumOne;
ENDPROC
```

Poslední procedurou pravého ramene je procedura *UvolneniKompletu*, která je volána potom, co dojde k sešroubování dílů. Cílem procedury je uvolnění hotového výrobku. To je realizováno pohybovou instrukcí *MoveJ* a funkcí *Offs*, díky které pneumatická přísavka vyjede z pozice *ppridrz* a tím díl uvolní.

```
PROC UvolneniKompletu()
     MoveJ Offs(ppridrz,0,0,30),v100,z30,VaccumOne;
ENDPROC
```

5.5 PROCEDURY PROGRAMU LEVÉHO RAMENE

Jako tomu bylo u pravého ramene, tak i v programu levého ramene je výchozím bodem procedura *main*, jejíž kód je uveden níže v textu. I v proceduře *main* levého ramene se nejdříve instrukcí *WaitDI* čeká na stisknutí tlačítka pro start cyklu. Po jeho stisknutí rameno, s využitím pohybové instrukce *MoveJ*, najíždí do výchozí pozice *pstart*. Následuje resetování hodnot persistentních proměnných (*zalozeno*, *zasroubovano*, *uvolneno*), inicializace a kalibrace prstů efektoru (instrukce *g_Init* a *g_Calibrate*). Ani po stisknutí tlačítka nemůže procedura *main* bezprostředně zahájit volání dílčích procedur, jelikož instrukcí *WaitUntil* čeká na změnu proměnné *zalozeno*. Hodnota této proměnné je změněna v rámci programu pravého ramene. V okamžik, kdy se proměnná *zalozeno* rovná hodnotě *TRUE*, je volána procedura *SroubovaciOperace*. Po jejím vykonání, je změněna hodnota proměnné *zasroubovano*, na což pravé rameno reaguje svou činností. Levé rameno tak opět čeká na změnu hodnoty proměnné *uvolneno*. Poté, co se hodnota proměnné *uvolneno* rovná *TRUE*, volá se procedura *OdlozeniKompletu*. Po jejím dokončení se rameno vrací do výchozí pozice *pstart*.

```
PROC main()
     WaitDI di0, 1;
     MoveJ pstart,v500,z100,Servo;
     zalozeno:=FALSE;
     zasroubovano:=FALSE;
     uvolneno:=FALSE;
     g_Init \holdForce:=8;
     g_Calibrate \Jog;
     WaitUntil zalozeno=TRUE;
     SroubovaciOperace;
     zasroubovano:=TRUE;
     WaitUntil uvolneno:=TRUE;
     OdlozeniKompletu;
     MoveJ pstart,v500,z100,Servo;
ENDPROC
```

Z procedury *main* se jako první volá procedura *SroubovaciOperace*. Ta zajistí zašroubování obou matic na již připravené šrouby s díly. Níže v textu je demonstrován kód pro zašroubování jedné matice, pro šroubování druhé matice je kód téměř totožný, liší se pouze názvy pozic. Struktura a funkce procedury byla již také popsána pomocí vývojového diagramu na obrázku 5.15 v oddílu 5.3. Z důvodu vyšší přehlednosti je kód procedury, s využitím komentářů, rozdělen podle dílčích úkonů. V první části je pomocí prstů efektoru vyzvednuta matice z mechanického přípravku (pozice *pmat1*) a přesunuta těsně nad závit šroubu (pozice *psroubovani1*). Pro rozevření prstů je použita instrukce *g_MoveTo* a pro uchopení procedura

Uchopit, jejíž funkce byla vysvětlena již v rámci popisu procedur pravého ramene, tedy v oddílu 5.4. Přesun matice je realizován základními pohybovými instrukcemi *MoveL*, *MoveJ* a funkcí *Offs*. V pozici těsně nad šroubem dojde k natočení nástroje do pozice vhodné pro šroubování. Pro tento účel byla vytvořena proměnná *pnatoceni* datového typu *jointtarget*. Do ní je nejdříve, po použití funkce *CrobT*, uložena aktuální pozice úhlů natočení kloubů ramene. Příkazem *pnatoceni.robax.rax_6:=−180* je přiřazena nová hodnota úhlu natočení šesté osy. Pohybovou instrukcí *MoveAbsJ* se rameno do změněné pozice natočí. Takto orientovaný nástroj poté usadí matici na závit šroubu a zahájí se vykonávání části pro zašroubování matice o 360°. To je realizováno pomocí cyklu *FOR* se čtyřmi opakováními. Proces šroubování je totiž rozdělen do čtyř dílčích pootočení po 90°. Pro robota je pak snazší vypočítat cílové pozice a není tak nutné řešit konfiguraci os – tento problém byl podrobněji komentován v oddílu 5.2, jenž se zabývá problematikou spojenou se šroubováním s pomocí robota YuMi. Šroubovací pohyb zajistí instrukce *MoveL* s rozšířením o funkci *RelTool*. Díky tomu rameno vykoná rotační a posuvný pohyb zároveň. Hodnota lineárního posunu v ose *z* o 0,75 mm při rotaci o 90° byla určena na základě velikosti stoupání závitu. Po vykonání rotačního pohybu o 360° se šestá osa ramene přiblíží k limitu jejího rozsahu pro otočení. Z toho důvodu musí být matice uvolněna a nástroj se vrátí zpět do pozice vhodné pro vykonání šroubovacího pohybu. Toho je znovu docíleno již výše popsanou částí kódu pro natočení šesté osy. Následně je matice znovu uchopena a může pokračovat proces šroubování. Tentokrát je matice zašroubována o 270°, čímž se přiblíží ke konci závitu šroubu a zahájí se proces dotahování. Dotahování probíhá pomocí cyklu *WHILE*, ve kterém se porovnává aktuální hodnota kroutícího momentu (*KrMom_aktualni*) s hodnotou definovanou pro dostatečné dotažení (*KrMom_dotazeno*). V každém cyklu se s využitím pohybové instrukce *MoveL* a funkce *RelTool* zašroubuje matice o 5° a funkcí *GetMotorTorque(6)* se změří hodnota kroutícího momentu šesté osy, jenž se uloží právě do proměnné *KrMom_aktualni*. Cyklus končí v okamžik, kdy je hodnota proměnné *KrMom_aktualni* větší než hodnota proměnné *KrMom_dotazeno*. Matice je pak považována za dostatečně dotaženou a může být instrukcí *g_MoveTo* uvolněna.

```
PROC SroubovaciOperace()
     !První Šroub
     !Vyzvednutí matice a najetí ke šroubu
     g_MoveTo 12\NoWait;
     MoveJ Offs(pmat1,0,0,20),v500,fine,Servo;
     MoveL pmat1,v30,fine,Servo;
     Uchopit;
     MoveL Offs(pmat1,0,0,30),v200,fine,Servo;
     MoveJ Offs(psroubovani1,0,0,5),v200,fine,Servo;
     !Natočení pro šroubování
     pnatoceni:=CJointT();
     pnatoceni.robax.rax_6:=-180;
     MoveAbsJ pnatoceni,v10,fine,Servo;
     !Usazení matice na závit šroubu
     MoveL Offs(CRobT(),0,0,-5.5),v5,fine,Servo;
     !Šroubování o 360 stupňů
     FOR i FROM 1 TO 4 DO
       MoveL RelTool(CRobT(),0,0,0.75\Rz:=90),v5,fine,Servo;
     ENDFOR
     !Natočení pro šroubování
     g_MoveTo 12;
     pnatoceni:=CJointT();
     pnatoceni.robax.rax_6:=-180;
     MoveAbsJ pnatoceni,v10,fine,Servo;
     Uchopit;
    !Šroubování o 270 stupňů
     FOR i FROM 1 TO 3 DO
       MoveL RelTool(CRobT(),0,0,0.75\Rz:=90),v5,fine,Servo;
     ENDFOR
    !Dotažení matice
     KrMom_dotazeno:=0.0016;
     KrMom_aktualni:=0;
     WHILE KrMom_dotazeno>KrMom_aktualni DO
       MoveJ RelTool(CRobT(),0,0,0.04\Rz:=5),v5,fine,Servo;
       KrMom_aktualni:=GetMotorTorque(6);
     ENDWHILE
     !Uvolnění matice
     g_MoveTo 12;
     MoveL Offs(CRobT(),0,0,40),v100,fine,Servo;

     !Druhý Šroub
     …
ENDPROC
```

Další volanou procedurou je procedura *OdlozeniKompletu*, jejíž kód je ukázán dále v textu. V rámci této procedury je sešroubovaný výrobek, s využitím pneumatické přísavky, vyjmut z mechanického přípravku (pozice *podloz*) a odložen na určené místo (pozice *podlozk*). Přesun výrobku je realizován posloupností pohybových instrukcí *MoveL* a *MoveJ*, doplněné

o funkci *Offs*. Spuštění a vypnutí sání vzduchu pneumatické přísavky je řízeno instrukcemi *g_VacuumOn1* a *g_VacuumOff*. Tyto instrukce jsou znovu doplněny o instrukci *WaitTime*, která zajistí potřebnou časovou prodlevu pro dostatečné uchopení a uvolnění výrobku před zahájením vykonávání navazujících pohybových instrukcí.

```
PROC OdlozeniKompletu()
     MoveJ Offs(podloz,0,0,50),v200,z20,VaccumOne;
     MoveL podloz,v50,fine,VaccumOne;
     g_VacuumOn1;
     WaitTime 0.5;
     MoveJ Offs(podloz,0,0,60),v200,z20,VaccumOne;
     MoveJ Offs(podlozk,0,0,40),v200,z20,VaccumOne;
     MoveL podlozk,v50,fine,VaccumOne;
     g_VacuumOff1;
     WaitTime 0.5;
     MoveJ Offs(podlozk,0,0,50),v300,z20,VaccumOne;
ENDPROC
```

5.6 MOŽNOST VYUŽITÍ STROJOVÉHO VIDĚNÍ ROBOTA ABB YUMI

Rozšiřujícím úkolem diplomové práce bylo ověřit možnost lokalizace součástek výrobku s využitím modulu strojového vidění robota ABB YuMi. Tento krok vzhledem k nastalé situaci nebylo možné plně realizovat. Z toho důvodu je zde popsán pouze základní postup práce pro použití strojového vidění. Součástí je i rozbor způsobů, jakými by bylo možné strojové vidění do vytvořené aplikace implementovat.

Pro práci s kamerou, v rámci software RobotStudio, slouží integrované prostředí *Integrated Vision*, které lze otevřít ze záložky *Controller*. Zobrazí se nová záložka *Vision*, kde se nachází panel nástrojů pro konfiguraci strojového vidění. Funkce panelu jsou rozděleny do několika záložek, a to v posloupnosti, ve které se k nim přistupuje při tvorbě úlohy. Jde o záložky *Camera*, *File*, *Image* a *Configure Job*. Prvním krokem je připojení ke kameře. To je možné realizovat pomocí záložky *Camera* a její funkce *Connect*, kde si uživatel vybere požadovanou kameru a jednoduše se k ní připojí. Dále je potřeba vytvořit nový konfigurační soubor s příponou názvu *.job*, do kterého je možné ukládat veškerá nastavení kamery a kdykoliv se k nim znovu vracet. K tomuto účelu slouží funkce záložky *File*, odkud lze nový konfigurační soubor vytvořit, načíst nebo uložit. Následuje záložka *Image*, kde se nachází funkce pro pořízení snímku a videa, jejich uložení a načtení. Důležitou záložkou je záložka *Configure Job*, odkud se realizují všechny důležité konfigurace strojového vidění. V první řadě je zde možné pomocí funkce *Setup Image* nastavit základní parametry obrazu pro pořizování snímků, jako je

například kontrast a jas. Poté je na základě pořízeného snímku funkcí *Calibrate* provedena kalibrace kamery. Kalibrace spočívá v převedení pixelů snímku na skutečné jednotky, například milimetry. Pomocí kalibrace je také určen počátek souřadnicového systému kamery. Uživatel si může vybrat z několika různých postupů. Nejpřesnější metodou je kalibrace pomocí vytištěné čtverečkované mřížky, kde uživatel definuje rozměry mřížky a kamera ji detekuje. Dále je také možné využít například metody kalibrace na základě detekce dvou hran, kde je známá jejich vzdálenost, či detekce krajů kruhu, kde je znám průměr tohoto kruhu atp. Software následně vyhodnotí kvalitu provedené kalibrace. Na základě toho se uživatel může rozhodnout, zda chce kalibraci opakovat, například s jiným nastavením obrazu, či za jiných světelných podmínek, nebo je s kvalitou kalibrace spokojen a pokračuje dalším krokem konfigurace. Tím je výběr nástroje pro lokalizaci předmětu v obrazu. Software nabízí řadu takovýchto nástrojů, jejichž seznam je uveden v položce *Add Part Location Tool*. Mezi nejběžněji používané patří *PatMax Pattern* a *Blob*. Prvním krokem nastavení nástroje *PatMax Pattern* je definice předmětu, který má být v obraze lokalizován. Toho může být docíleno pořízením snímku předmětu připojenou kamerou, přičemž konkrétní předmět uživatel označí speciálním nástrojem. Funkce nástroje *PatMax Pattern* poté spočívá v určení přesné pozice definovaného předmětu pomocí souřadnic x a y, úhlu natočení a počtu nalezených předmětů, kdekoliv v obrazu. Funkce nástroje *Blob* je založena na lokalizaci skupiny tmavých nebo světlých pixelů a určení polohy této skupiny. Tento nástroj se běžně používá jako doplněk dalších nástrojů strojového vidění. Existují i inspekční nástroje, které lze nelézt v položce *Add Part Inspection Tool*. Ty slouží ke zkoumání předmětů nalezených lokalizačními nástroji. Inspekční nástroje mohou být využity například pro kontrolu přítomnosti předmětů, měření, počítání apod. Uživatel s využitím lokalizačních a inspekčních nástrojů získá potřebná data, která je poté možné pomocí funkce *Output to RAPID* převést do programu RAPID vytvořené aplikace. Například může jít právě o data definující polohu nalezeného předmětu, úhel natočení, počet, rozměry atp. Tyto výstupní data z kamery jsou uloženy do proměnných programu RAPID a uživatel zde s nimi může dále pracovat.

V jazyku RAPID jsou pro práci s kamerou určené speciální instrukce, díky kterým lze implementovat strojové vidění do aplikace robota. Pro tento záměr jsou ale také velice užitečné i šablony s již předpřipravenými částmi kódu, které je možné jednoduše vložit do vytvořeného programu. Takto předpřipravené úryvky kódu jsou v software RobotStudio označovány jako *Snippets* a lze je nalézt v záložce *RAPID*. Každá šablona obsahuje proceduru zajišťující vybranou činnost kamery. Po jejím vložení do programu je potřeba doplnit pouze základní parametry, jako je například název připojené kamery, typ použitého nástroje atp. Zároveň je ale

nutné program rozšířit o volání vložené procedury. Příkladem může být šablona *Move to detected object*, jejíž procedura slouží k detekci předmětu pomocí pořízeného snímku, určení jeho pozice a následnému přesunutí nástroje robota k nalezenému předmětu. Hlavní část kódu této šablony, po její implementaci do programu, je ukázána dále v textu.

```
…
CONST string myjob := "myjob.job";
TASK PERS wobjdata mywobj:=[…];
CONST robtarget myrobtarget:= […];
VAR cameratarget mycameratarget;

PROC MoveToDetectedObject()
    CamSetProgramMode mycamera;
    CamLoadJob mycamera, myjob;
    CamSetRunMode mycamera;
    CamReqImage mycamera;
    CamGetResult mycamera, mycameratarget;
    mywobj.oframe := mycameratarget.cframe;
    MoveL myrobtarget, v100, fine, mytool \WObj:=mywobj;
ENDPROC
…
```

Úvodní část kódu slouží k deklaraci dat, kde jako první uživatel doplní název vytvořeného konfiguračního souboru (*"myjob.job"*). Dále zde jsou vytvořeny proměnné datových typů *wobjdata* a *robtarget*. Proměnná typu *wobjdata* slouží k definici nových souřadnicových systémů (*user frame* a *object frame*). Počátek souřadnicového systému *user frame* se musí shodovat s počátkem souřadnicového systému kamery, jenž byl určen při její kalibraci. Počátek souřadnicového systému *object frame* se poté přemisťuje podle polohy detekovaného předmětu – to ještě bude blíže vysvětleno dále v textu. Pozice určená proměnnou typu *robtarget* bude využita pro přesun nástroje k nalezenému předmětu. Tato pozice musí být definována v počátku vytvořeného souřadnicového systému objektu (*object frame*). Poslední deklarovanou proměnnou je proměnná typu *cameratarget*, do které se ukládají data z kamery. U instrukcí procedury *MoveToDetectedObject* musí uživatel v první řadě upravit název připojené kamery (*mycamera*) a v pohybové instrukci *MoveL* (poslední řádek kódu procedury) definovat název nástroje (*mytool*), jehož TCP se přesune k nalezenému předmětu.

V průběhu vykonávání programu je pak instrukcí *CamSetProgramMode* kamera přepnuta do programového módu (offline). V programovém módu se pomocí instrukce *CamLoadJob* načte konfigurační soubor obsahující informace o nastavení parametrů obrazu, kalibraci a vybraných nástrojích strojového vidění. Po jeho načtení se instrukcí *CamSetRunMode* kamera přepne do módu online, ve kterém je instrukcí *CamReqImage* pořízen

snímek. Instrukce *CamGetResult* získá výsledky z pořízeného snímku a uloží je do proměnné typu *cameratarget*. Součástí výsledků jsou i souřadnice předmětu, který byl ve snímku detekován použitým lokalizačním nástrojem. Na dalším řádku kódu se hodnoty těchto souřadnic uloží do proměnné typu *wobjdata* a přepíší její stávající hodnoty. Tím dojde k posunutí počátku souřadnicového systému objektu (*object frame*) podle souřadnic nalezeného předmětu (počátek souřadnicového systému je přesunut na pozici detekovaného předmětu). Se souřadnicovým systémem objektu se posune i pozice proměnné typu *robtarget*, která je definována právě v jeho počátku. Díky tomu se pomocí pohybové instrukci *MoveL* nástroj přesune na nově získanou pozici nalezeného předmětu. Kód následně stačí rozšířit o instrukce pro uchopení a přemístění předmětu.

Strojové vidění by v rámci vytvořené montážní operace mohlo být využito i pro kontrolu správného založení součástek. Jelikož, jak již bylo v práci několikrát zmíněno, ještě před spuštěním cyklu programu obsluha musí založit všechny potřebné součástky na připravené mechanické přípravky. Založení součástek potvrzuje stisknutím tlačítka, kterým zároveň zahajuje vykonávání činností montážní operace. V tento moment robot počítá s tím, že jsou všechny součástky korektně založeny. Opačná situace, kdy obsluha například zapomene založit některou ze součástek, není v programu důkladně ošetřena. Z toho důvodu by první činností robota mohla být právě kontrola založených dílů, s využitím strojového vidění. Po stisknutí tlačítka by robot pomocí kamery zkontroloval, zda jsou opravdu všechny součástky na svých místech. Pokud by tomu tak nebylo, obsluha by na tuto situaci byla upozorněna – například chybovým hlášením, zobrazeným na panelu FlexPendant. V tento okamžik by obsluha měla možnost doplnit chybějící díl a program by mezitím čekal na další stisknutí tlačítka, po kterém by znovu následovala kontrola založení. V případě, kdy by kamera vyhodnotila správné založení součástek, by robot zahájil vykonávání navazujících činností. Podobným způsobem by mohla být detekce správného založení použita i v kterémkoliv dalším kroku vytvořené aplikace.

Podrobný rozbor kroků konfigurace kamery, včetně popisu práce se šablonami a s instrukcemi strojového vidění, je součástí volně dostupných dokumentací na webových stránkách společnosti ABB.

6 ZÁVĚR

V rámci diplomové práce byla v požadovaném rozsahu realizována zadaná úloha s robotem realizující montážní operaci. K tomuto účelu byl využit dvouramenný kooperující robot ABB YuMi, nacházející se v jedné z laboratoří fakulty Elektrotechniky a informatiky na univerzitě v Pardubicích. Pro montážní operaci byly vybrány součástky stavebnice ROTO Abc, které svými vlastnostmi splňují nutná bezpečnostní kritéria. Na základě jejich rozměrů byly navrženy a pomocí 3D tiskárny vytištěny mechanické přípravky, které jsou nezbytnou součástí celé aplikace. Ty slouží k určení přesných pozic použitých součástek pro odebrání robotem a jejich zajištění při šroubovací operaci. Pro tvorbu programu byl využit software ABB RobotStudio. V rámci testování aplikace bylo vyzkoušeno několik různých přístupů pro spolehlivé nasazení matice na závit šroubu a utahování závitu. Pro najetí nástroje s maticí na závit šroubu se jako nejvhodnější ukázalo použití pevně definované pozice, ve které je matice na závit šroubu usazena. V každém cyklu programu jsou totiž použity vždy stejně dlouhé šrouby a díky vyrobeným mechanickým přípravkům jsou pokaždé založeny stejným způsobem. Pro utahování závitu byla zvolena varianta, ve které bylo využito možnosti měření velikosti kroutícího momentu osy vykonávající šroubovací pohyb. Díky tomu je šroubový spoj vždy pevně dotažen. Všechny vyzkoušené postupy byly podrobněji popsány v oddílu 5.2, který je zaměřen na problematiku šroubového spojení s pomocí robota ABB YuMi. V textu práce je také s využitím vývojových diagramů popsána struktura vytvořeného programu a komentovány kódy jednotlivých procedur.

Na vytvořené aplikaci měla být otestována i možnost využití strojového vidění pro lokalizaci součástek. Tento rozšiřující krok nakonec nebyl vzhledem k nastalým okolnostem realizován v plném rozsahu. V oddílu 5.6 byl alespoň popsán základní přístup k práci se strojovým viděním v software RobotStudio. Součástí toho je i rozbor možností případného využití strojového vidění v rámci vytvořené aplikace.

POUŽITÁ LITERATURA

AUTOMATIZACE PRŮMYSLOVÉHO BALENÍ: Simulace v robotizaci. 2011. *Svět balení* [online]. [cit. 20. 2. 2020]. Dostupné z: https://www.svetbaleni.cz/2011/03/01/sb-2-2011-hlavn-tma-automatizace-prmyslovho-balen-simulace-v-robotizaci/.

Ceník RobotStudio 6. 2018. *ABB* [online]. [cit. 20. 2. 2020]. Dostupné z: https://library.e.abb.com/public/979bab8b4a6042769205dbe8238db65b/Cenik%20RobotStudio%202018.pdf.

Collaborative robotics. 2016. *KUKA* [online]. [cit. 10. 1. 2020]. Dostupné z: https://www.kuka.com/cscz/tisk/news/2016/03/collaborative-robotics/.

Collaborative Robot. ©2011-2020. *FANUC* [online]. [cit. 10. 1. 2019]. Dostupné z: https://www.fanuc.co.jp/en/product/robot/f_r_collabo.html.

DUCHOSLAV, P. Co je to kolaborativní robot? 5 věcí, které byste o něm měli vědět. 2017. *Factory Automation* [online]. [cit. 10. 1. 2020]. Dostupné z: https://factoryautomation.cz/co-je-to-kolaborativni-robot-5-veci-ktere-byste-o-nem-meli-vedet/.

Engineer automated production systems using robotics and automation simulation. ©2020. *SIEMENS* [online]. [cit. 20. 2. 2020]. Dostupné z: https://www.plm.automation.siemens.com/global/en/products/manufacturing-planning/robotics-automation-simulation.html.

History. ©2018. *Kawasaki Robotics* [online]. [cit. 10. 1. 2020]. Dostupné z: https://robotics.kawasaki.com/en1/anniversary/history/.

IRB 14000 YuMi. ©2020. *ABB* [online]. [cit. 10. 1. 2020]. Dostupné z: https://new.abb.com/products/robotics/cs/prumyslove-roboty/yumi.

IRB 14050 Single-arm YuMi Collaborative Robot. ©2020. *ABB* [online]. [cit. 10. 1. 2020]. Dostupné z: https://new.abb.com/products/robotics/industrial-robots/irb-14050-single-arm-yumi.

IRC5. ©2020. *ABB* [online]. [cit. 15. 04. 2020]. Dostupné z: https://new.abb.com/products/robotics/cs/ridici-systemy/irc5.

KMR iiwa. ©2019. *KUKA* [online]. [cit. 10. 1. 2019]. Dostupné z: https://www.kuka.com/cs-cz/produkty,-slu%C5%BEby/mobilita/mobiln%C3%AD-roboty/kmr%C2%A0iiwa.

Kolaborativní roboti společnosti Universal Robots. ©2020. *Universal Robots* [online]. [cit. 10. 1. 2020]. Dostupné z: https://www.universal-robots.com/cs/produkty/.

KOZLÍK, P. 2019. Roboty místo zaměstnanců?. *Lidovky.cz* [online]. [cit. 27. 2. 2020]. Dostupné z: https://www.lidovky.cz/byznys/firmy-a-trhy/nektere-profese-by-clovek-ani-nemel-delat-rika-odbornik-na-robotizaci-vitezslav-lukas.A190204_162154_firmy-trhy_pkk.

LBR iiwa. ©2019. *KUKA* [online]. [cit. 10. 1. 2019]. Dostupné z: https://www.kuka.com/cs-cz/produkty,-slu%C5%BEby/robotick%C3%A9-syst%C3%A9my/pr%C5%AFmyslov%C3%A9-roboty/lbr%C2%A0iiwa.

ROBOTICS ONLINE MARKETING TEAM. 2018. *Collaborative Robots Market Update 2018* [online]. [cit. 10. 1. 2020]. Dostupné z: https://www.robotics.org/blog-article.cfm/Collaborative-Robots-Market-Update-2018/84.

RobotStudio. ©2020. *ABB* [online]. [cit. 20. 2. 2020]. Dostupné z: https://new.abb.com/products/robotics/cs/robotstudio.

Řada robotů TX2. 2019. *Stäubli* [online]. [cit. 10. 1. 2020]. Dostupné z: https://www.staubli.com/cs-cz/file/21193.show.

TAUFER, I.; KOTYK, J.; JAVŮREK, M. 2014. *Jak psát a obhajovat závěrečnou práci, bakalářskou, diplomovou, rigorózní, habilitační.* 2. opravené a doplněné vydání. Pardubice: Univerzita Pardubice. 48 s. ISBN 978-80-7395-746-9.

Teaching a robot using hand guidance. ©2019. *FANUC* [online]. [cit. 10. 1. 2020]. Dostupné z: https://www.fanuc.eu/uk/en/robots/accessories/hand-guidance.

Technical data IRB 14000 YuMi. ©2020. *ABB* [online]. [cit. 10. 1. 2020]. Dostupné z: https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi/irb-14000-yumi-data.

UNIVERSAL ROBOT UR3e. ©2020. *Universal Robots* [online]. [cit. 10. 1. 2020]. Dostupné z: https://www.universal-robots.com/products/ur3-robot/.

VOJÁČEK, A. 2017. Robot vs. Cobot. *Automatizace.hw.cz* [online]. [cit. 10. 1. 2020]. Dostupné z: https://automatizace.hw.cz/robot-vs-cobot.html.

Využití robotiky. 2019. *Český statistický úřad* [online]. [cit. 10.1.2020]. Dostupné z: https://www.czso.cz/documents/10180/61601888/06200518k07.pdf/1c788638-8535-4c4e-ad75-ee2bd43f51ce?version=1.1.

WIRELESS TEACH PENDANTS FOR ROBOTS. 2019. *Roboticsbook* [online]. [cit. 26. 2. 2020]. Dostupné z: https://roboticsbook.com/wireless-teach-pendants-for-robots/.

YuMi. ©2015. *ABB* [online]. [cit. 10. 1. 2020]. Dostupné z: https://library.e.abb.com/public/5662b754739545899518eea9ea7e3781/yumi_datasheet_cz.pdf.

ŽÁČEK, M. 2014. Historie robotů? Sahá až do řecké mytologie!. *Factory Automation* [online]. [cit. 10. 1. 2020]. Dostupné z: https://factoryautomation.cz/historie-robotu-saha-az-do-recke-mytologie/.

PŘÍLOHY

A – CD

Příloha A

Příloha k diplomové práci

Montážní operace s šroubovým spojením s využitím robota ABB YuMi

Petr Čepelák

CD

Obsah