VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

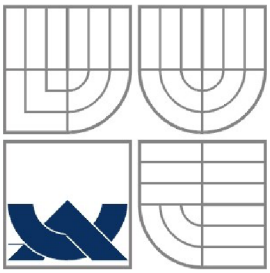
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

OVLÁDÁNÍ ROBOTICKÉHO RAMENA MITSHUBISHI MELFA 6 SL

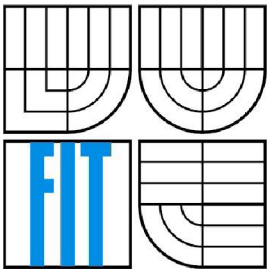
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Petr Schindler
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

OVLÁDÁNÍ ROBOTICKÉHO RAMENA MITSHUBISHI MELFA 6 SL

CONTROL OF ROBOTIC ARM MITSUBISHI MELFA 6 SL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Petr Schindler
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing., CSc., Zbořil František V.
SUPERVISOR

BRNO 2009

Abstrakt

Tato bakalářská práce popisuje ovládánírobotického ramene Mitsubishi Melfa 6 SL. Práce je pouze částí většího projektu, jehož cílem je vytvořit online laboratoř, která by měla sloužit pro podporu výuky umělé inteligence. Tento dokument se zaměřuje na popis vlastností ramene a jeho ovládání pomocí počítače. Popsáno je také vybavení laboratoře. Cílem projektu bylo vytvoření knihovny pro řízení ramene vlastním programem.

Abstract

This thesis is focused on the control of the robotic arm Mitsubishi Melfa 6 SL. It is a part of a bigger project, which goal is to build an online laboratory supporting the teaching of artificial intelligence. This document describes methods which can be used to control the arm, especially using personal computer. It also briefly describes the laboratory itself. The goal of this project was to create a set of routines, which can be used to control the robot using one's own software.

Klíčová slova

Robotické rameno, Mitsubishi Melfa 6 SL, Melfa-Basic IV, Python, robot, Cosirop

Keywords

Robotic arm, Mitsubishi Melfa 6 SL, Melfa-Basic IV, Python, robot, Cosirop

Citace

Petr Schindler: *Ovládání robotického ramena Mitshubishi Melfa 6 SL*, bakalářská práce, Brno, FIT VUT v Brně, 2009

Ovládání robotického ramena Mitshubishi Melfa 6 SL

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením doc. Ing., Františka V. Zbořila, CSc. Další informace mi poskytl Josef Skládanka.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Petr Schindler
20.5.2009

Poděkování

Rád bych poděkoval doc., Ing. Františku V. Zbořilovi za možnost pracovat na tomto projektu, za jeho rady a odborné vedení. Dále chci poděkovat Josefu Skládankovi za jeho nemalou pomoc. Chtěl bych také poděkovat všem, kteří mě při mé práci podporovali.

Obsah

1 Úvod

Tato práce se zabývá robotickým ramenem Mitsubishi Melfa 6 SL, jeho ovládáním a nakonec hlavně zpracováním knihovny, která bude mít na starosti řízení ramene pomocí počítače. Práce je součástí většího projektu online laboratoře, který bude zaměřen na podporu výuky umělé inteligence na fakultě informačních technologií VUT v Brně. Robotické rameno by mělo v koordinaci se soustavou kamer vyhledávat kostky a podle algoritmů umělé inteligence s nimi dále pracovat.

Na tomto velkém projektu pracuji ještě s kolegou Josefem Skládankou, jehož bakalářská práce je zaměřena na ovládání kamer a rozpoznávání kostek na pracovní ploše.

Na začátku této práce řeknu, co je to robot obecně. V další části pak popíšu laboratoř, v níž se robotické rameno nachází. Popíšu podrobněji robota Melfa a kamery Axis.

Další částí práce je popis ovládání ramene. V této kapitole rozeberu způsoby, jakými se rameno dá ovládat. Zaměřím se hlavně na program Cosirop a programovací jazyk Melfa-Basic IV.

Poslední část se týká práce na knihovně, která bude mít ve velkém projektu na starosti ovládání ramene. Popíšu návrh knihovny a její použití. Na konci kapitoly popíšu, jak probíhalo testování.

V závěru navrhnu možný směr pokračování projektu, či lépe zapojení do velkého projektu.

2 Pojmy

Tato kapitola se jen krátce zmíní o pojmu robot. Jelikož se má práce týká ovládání robota, myslím, že by bylo dobré mít alespoň nějakou představu o tom, co to robot je.

2.1 Robot

Robot je mechanická nebo virtuální umělá hybná síla. Obvykle se jedná o systém, který na základě svého vzhledu nebo pohybu vytváří dojem, že má svůj vlastní účel nebo působnost. Slovo robot se může vztahovat jak k fyzickým robotům, tak i k virtuálním softwarovým agentům, které však bychom měli řadit spíše k typu odlišných robotů. Zatímco se stále diskutuje o tom, které stroje spadají do kategorie robotů, je zřejmé, že typický robot bude muset mít (i když ne nezbytně) následující vlastnosti:

- nemá „přirozený“ původ, tzn. je uměle vytvořený
- vnímá své životní prostředí a manipuluje s předměty v něm nebo je s nimi v interakci

- má určitou schopnost provádět výběr podle daného prostředí, často s použitím automatického řízení nebo předem naprogramované sekvence
- je programovatelný
- pohybuje se v jedné nebo více rotačních nebo translačních rovinách
- provádí pohotové koordinované pohyby
- jeví se, jako by měl záměr nebo působnost [4]

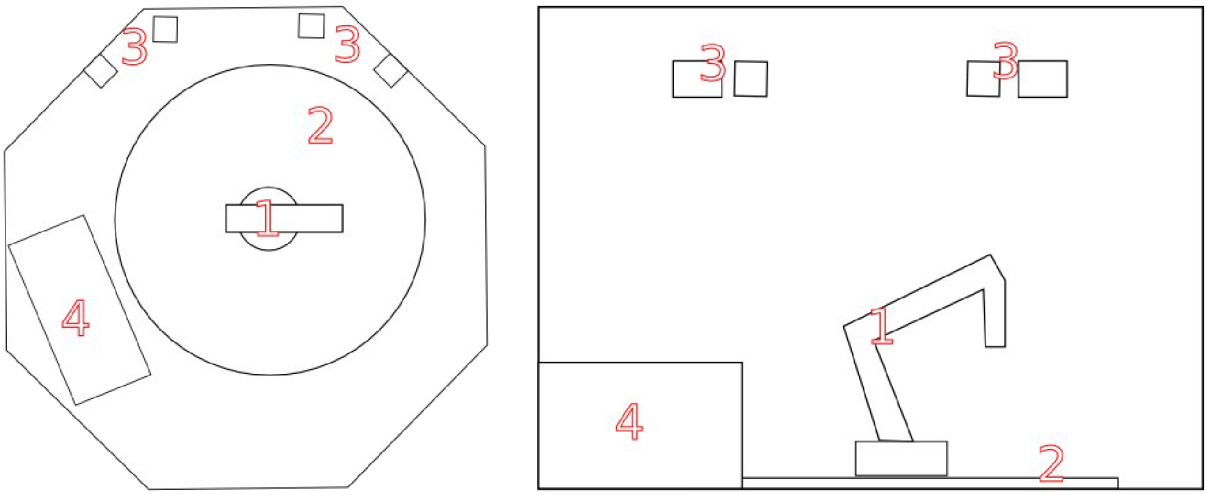
3 Laboratoř

Laboratoř robotického ramena se nachází v jiho-západní věži budovy Božetěchova 1/2. Uprostřed místnosti se nachází kruhová pracovní plocha, kterou tvoří dřevěná deska. Ve středu desky je pak připevněno samotné robotické rameno Melfa 6SL. Deska má rozměry odpovídající maximálnímu dosahu ramene a nějaké rezervě. V laboratoři se dále vyskytují 4 kamery Axis 214. Kamery jsou umístěny na kovové konstrukci připevněné ke stropu. Tyto kamery se po konstrukci mohou posouvat. Dále pak se v laboratoři nachází řídící jednotka robotického ramene a počítač.

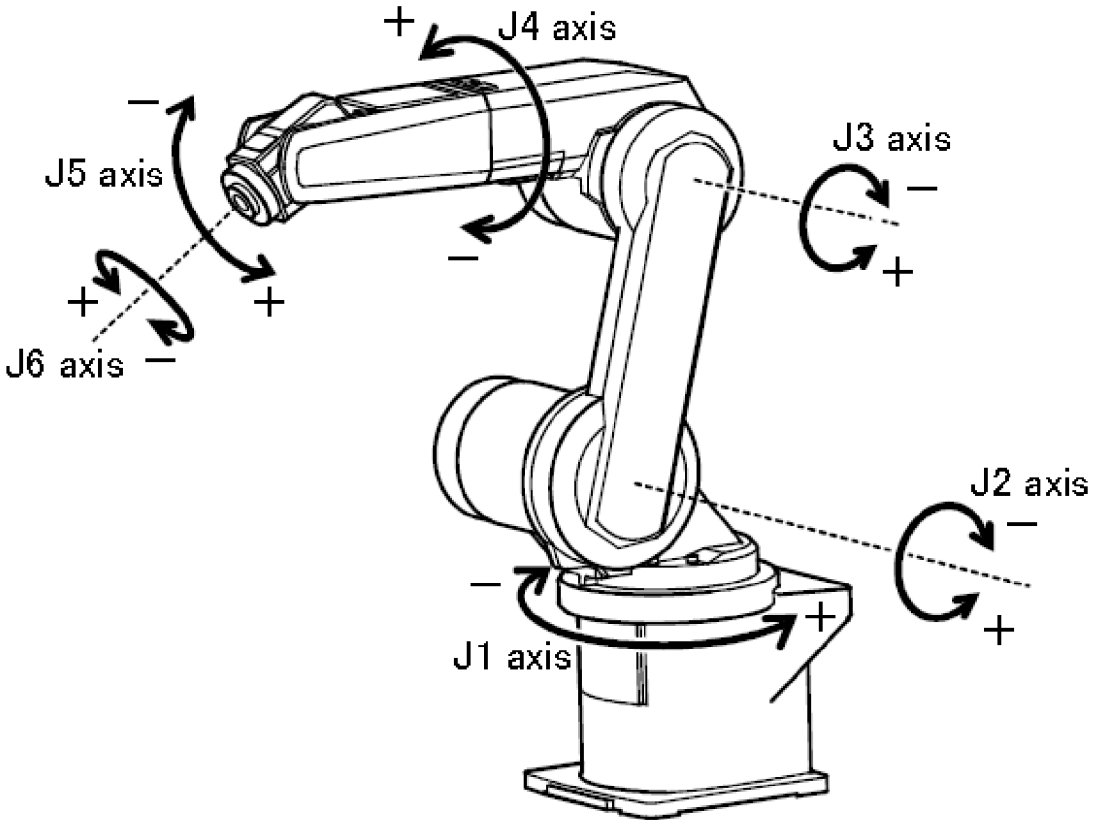
Místnost je dále vybavena síťovými přípojkami. Je zde tedy možné připojit se k síť VUT, případně k internetu.

V laboratoři se pak nachází tlačítko generální stop, které je na stěně blízko dveří. To při stisknutí vypne proud v celé místnosti. Podobné tlačítko se ještě nachází na Teaching device (viz. kapitola 4 Ovládání RR) a na řídící jednotce. Tyto dvě tlačítka pouze zastaví činnost ramene. S pomocí těchto tlačítek je možné se vyhnout nehodám, při kterých by mohlo dojít k újmě na zdraví či majetku.

Obrázek 1 znázorňuje rozmístění jednotlivých předmětů v laboratoři. Obrázek není v měřítku.



Obrázek 1: Laboratoř robotického ramena: 1 – robotické rameno, 2 – pracovní plocha, 3 – kamery, 4 – řídící jednotka a pc (obrázek převzat z [2])



Obrázek 2: Rameno Melfa 6SL (obrázek převzat z [3])

3.1 Robotické rameno Melfa 6 SL

Jedná se o robotické rameno firmy Mitsubishi se šesti klouby (obrázek 2). Každý z kloubů je nezávislý na ostatních. To znamená, že otáčíme-li jedním, ostatní zůstávají ve své poloze. K ramenu je připojeno chapadlo Schunk PG 70. Robotické rameno Melfa 6 SL je zaměřeno na automatickou, programem řízenou práci v průmyslu. Její určení se odvíjí od připojeného nástroje. V našem případě je připojena mechanické chapadlo, s jehož pomocí je možné chytat různé předměty. Díky vyměnitelným nástavcům, lze chapadlo přizpůsobit předmětům, která má držet.

Rameno Melfa 6 SL má maximální nosnost 6 kg. Přesnost servomotorů zajišťuje zopakovatelnost (*repeatability*) pohybu s přesností na 0,02 mm. Maximální rychlost je 8500 mm/s. Tyto parametry jsou pro účely projektu více než dostačující.

Robotické rameno ovládá řídící jednotka. Řídící jednotka obsahuje několik rozhraní pro komunikaci s uživatelem.

- *teaching device* je ovladač obsahující tlačítka pro ovládání pohybů ramene, z nichž pro nás nejdůležitější je červené tlačítko "stop", při jehož stisknutí se rameno okamžitě zastaví.
- COM port či Ethernet. Pomocí těchto dvou rozhraní je možné spojit řídící jednotku s počítačem.

3.1.1 Pozice

Při práci s ramenem je potřeba určovat aktuální a cílové pozice ruky a často i její úplnou dráhu. Zjištěné pozice se pak využívají v programech jako parametry funkcí (např. MOV). Pozice ruky se může určovat třemi způsoby. Prvním jsou souřadnice konce ramene (resp. ruky) s počátkem soustavy na pracovní podložce (ploše) uprostřed robota. Měřítko os je v milimetrech. Pro snadnější orientaci v tomto prostoru je na pracovní ploše několik značek udávající hodnoty na osách X a Y. Osa Z udává výšku konce ramene. Osy X a Y jsou pak naznačeny na zemi. Postavíme-li se ve směru osy X, osa Y roste doleva. U tohoto typu měření pozice je potřeba ještě určit naklonění samotného nástroje (ruky). Naklonění udávají další 3 parametry (označované jako A, B, C), které značí rotaci okolo os (X, Y a Z).

Druhým způsobem je určování pozice podle natočení jednotlivých kloubů. Natočení kloubů se určuje ve stupních a počítá se od počáteční pozice (nastavuje se pomocí parametrů).

Třetí souřadnicový systém je cylindrický. Tři základní parametry určují vzdálenost od středu, úhel otočení (odpovídá pohybu prvního kloubu) a výšku. Vzdálenosti jsou v milimetrech, úhly ve stupních. Natočení nástroje je stejné jako u pravoúhlého souřadnicového systému.

Pozice robota obsahuje ještě dva parametry. Těmi jsou Stavové data (*Posture data*) a Multi-rotační data (*Multi-rotation data*). Stavové data udávají polohu robota pomocí tří příznaků. První z nich označuje, jestli je rameno nalevo (*left*) nebo napravo *(right).* Toto se určuje podle pozice pátého kloubu vůči ploše, která prochází pátým kloubem a je kolmá k zemi. Další příznak udává, jestli je pátý kloub nad (*Above*) či pod (*Below*) rovinou procházející druhým a třetím kloubem. Poslední příznak udává, je-li šestý kloub překlopen (*Flip*) či ne *(NonFlip).* Překlopen je pokud je konec ruky nad rovinou danou čtvrtým a pátým kloubem.

Multi-rotační data udávají počet přetočení jednotlivých kloubů.

3.1.2 Nastavení

V řídící jednotce je uloženo mnoho parametrů, kterými lze ovlivňovat běh ramene. Tyto parametry lze přepisovat z počítače. Parametrů je několik stovek, pro nás jsou důležité a zajímavé jen některé. Nejdříve zde uvedu rozdělení parametrů do kategorií. Dále uvedu parametry, které jsem používal.

Každý z parametrů má svůj název a svou hodnotu, jejíž typ se mění v závislosti na tom, kterou vlastnost ovlivňuje daný parametr. Parametry jsou rozděleny do několika kategorií:

- Parametry pohybu (Movement parameter): Tyto parametry určují prostor ve kterém se může rameno pohybovat a souřadnicový systém. Mezi důležité parametry této skupiny patří např. User area, Free plane limit nebo Safe point position.
- Parametry pro práci se signály (Signal parametr): Tyto parametry pracují se signály zasílanými při výskytu nějaké události. Ve své práci tyto parametry nepoužívám.
- Operační parametry (Operation parameter): Těmito parametry nastavujeme vlastnosti řídící jednotky či Teching device. Jediná hodnota, kterou jsem při práci změnil je u parametru Buzzer ON/OFF, který zapíná či vypíná výstražný zvuk při vzniklé chybě.
- Příkazové parametrý (Command parameter): S jejich pomocí ovlivňujeme spouštění programů a jazyk robota. Ani zde jsem neměnil žádné hodnoty.
- Komunikační parametry (Communication parameter): Ovlivňují komunikaci s počítačem. Opět jsem hodnoty neměnil.

Parametry, které jsem nastavoval jsou uvedeny níže. Nastavení jsem prováděl pomocí programu Cosirop (viz. kapitola 4.3). Názvy parametrů jsem ponechal v anglickém jazyce.

- User area – pomocí těchto parametrů (je to skupina 5 parametrů) lze definovat prostor (kvádr, pomocí dvou protilehlých rohů). Slouží pro nastavování výstupních signálů, pokud se rameno nachází uvnitř tohoto prostoru. Protilehlé rohy definujeme parametry AREA*P1 a AREA*P2 (znak hvězdička udává číslo od 1 do 8, můžeme tedy definovat až osm těchto prostorů). Body

jsou určeny souřadnicemi v souřadném systému ramene (počátek soustavy je uprostřed ramene). Parametrem AREA*AT určujeme akci, která bude provedena, když rameno vstoupí do definovaného prostoru. Pokud je nastaven na nulu, je tento prostor ignorován. Nás zajímá nastavení na dvojku, poněvadž tím zajistíme vznik chyby při pokusu o vniknutí ramene do prostoru. Tohoto lze tedy využít pro ochranu jak robota samotného, tak i pro ochranu předmětů nacházejících se v jeho okolí (například kamer).

- Free plane limit – těmito parametry definujeme plochu, která má podobný účel jako *User area.* Plocha je určena třemi body. K definici slouží parametr SFC*P (opět můžeme nastavit až osm těchto ploch). Tři body jsou definovány třemi souřadnicemi. Do parametru se zapisují body postupně (tedy X1, Y1, Z1, X2, Y2, …, Z3). Parametrem SFC*AT určíme, kde má robot pracovat. Hodnota jedna znamená, že rameno pracuje na straně, kde se nachází počátek soustavy. Hodnota mínus jedna znamená, že rameno pracuje na opačné straně roviny. Pokud se rameno pokusí proniknout skrz tuto rovinu, skončí to chybou. Parametry jsem nastavil tak, aby rameno nemohlo narazit do země ani do stěn, případně do stolu s počítačem.
- Buzzer ON/OFF – je-li parametr BZR nastaven na jedničku, pak při vzniku chyby dává tuto skutečnost řídící jednotka najevo hlasitým pískáním. Toto pískání je při testování (obzvlášť při testování chyb) nepříjemné. Nastavením BZR na nulu se pískání vypne.
- Impact detection – parametr COL. Pomocí tohoto parametru lze teoreticky nastavit ochrana proti nárazu. Je-li tento parametr správně nastaven, měl by při nárazu vypnout rameno. Bohužel je náraz detekován pomocí zrychlení. Ke správnému fungování je potřeba mít správně nastavené parametry chapadla (rozměry, váhu, těžiště a podobně). Protože s chapadlem nyní nikdo nepracuje, tento parametr nelze použít.

3.2 Kamery Axis 214

Axis 214 je profesionální IP kamera určená především pro zabezpečení a vzdálený dohled. Streamovaný obraz z kamery může být po síti v reálném čase přenášen ve formátu MPEG-4, nebo jako Motion JPEG. Tento stream bude poskytován uživatelům laboratoře pro vizuální kontrolu.

Pro zpracování obrazu je však vhodnější použít data, která neprošla ztrátovou kompresí. Zde je využita další vlastnost IP kamery – možnost získání samostatného snímku ve formátu BMP. Nekomprimovaná bitmapa je sice náročnější na online přenos, avšak krátká prodleva při získání snímku je vyvážena kvalitou dat pro další zpracování. Prodleva je taktéž výrazně minimalizována propojením kamer a řídícího počítače 1Gbit LAN sítí.[2]

3.2.1 Technická specifikace kamery AXIS 214

Senzor	1/4" HAD CCD
Ohnisková vzdálenost	4.1 - 73.8 mm
Světelnost	F1.3 - 3.0
Závěrka	1/10000 - 1 s
Zoom	18x optický, 12x digitální
Komprese videa	MPEG-4 Part 2, Motion JPEG
Maximální rozlišení	704x576 pixelů
API	VAPIX®

Kamery poskytují funkce pro noční vidění, automatické nastavení clony a ostření, detekci pohybu v obraze, automatické vytváření a archivaci záznamů atd. Pro účely rozpoznávání obrazu je však důležitá především kvalita objektivu, která je na vysoké úrovni, a dále maximální rozlišení obrazu, které je bohužel poměrně malé, a proto je nutné prostor laboratoře při zpracování segmentovat na menší části, které se zpracovávají samostatně. [2]

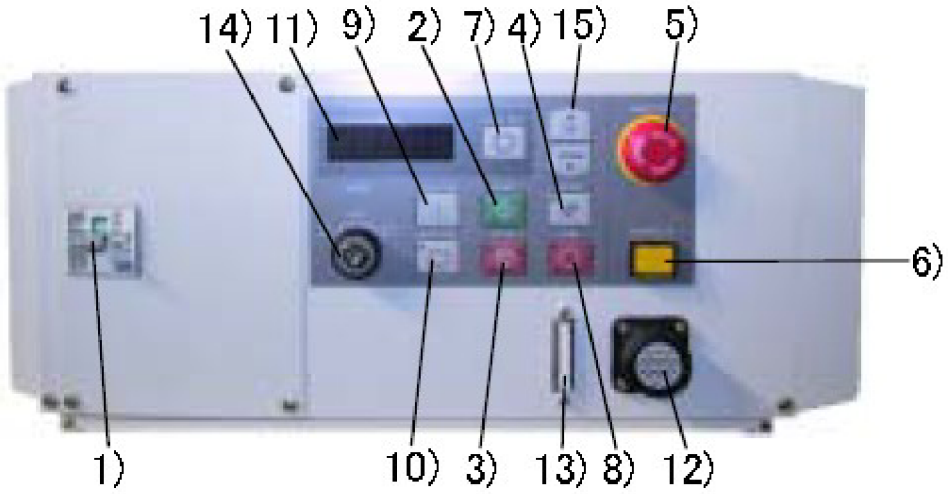
3.3 Chapadlo Schunk PG 70

Ovládání chapadla není součástí mé práce. Poněvadž je připojené zařízení od jiné firmy, než je rameno, nelze používat funkce jazyka Melfa-Basic IV *HOPEN* a *HCLOSE* pro jeho obsluhu. Chapadlo bylo před ukončením mé bakalářské práce ještě nebylo zprovozněno, tudíž jsem jej nemohl využívat.

4 Ovládání RR

Tato kapitola pojednává o způsobech, kterými je rameno možné ovládat. V první podkapitole se dovíme základní informace o řídící jednotce robota, najdeme v ní též popis jednotlivých ovládacích prvků a nastavení pro různé mody práce. Pro počáteční seznámení s ovládáním ramene mi sloužil tzv. teaching device. V druhé podkapitole jej popíši. Dále v této podkapitole bude popsána základní práce s tímto zařízením. Pro práci s robotem za pomoci počítače je dodáván program Cosirop, který jsem využíval k naučení se programovacího jazyka pro obsluhu robotického ramene – Melfa-Basic IV. Program Cosirop též sehrál podstatnou roli v získávání informací o komunikačním protokolu mezi počítačem a řídící jednotkou. V předposlední podkapitole bude popsán základ jazyka Melfa-Basic IV.

Nakonec této kapitoly se zaměřím na samotnou komunikaci s řídící jednotkou (přes COM port počítače).



Obrázek 3: Řídící jednotka (obrázek převzat z [1])

4.1 Řídící jednotka

Řídící jednotka posílá příkazy přímo rameni. Obsahuje interpret jazyka Melfa-Basic IV, který převádí příkazy jazyka do pokynů pro robota. Řídící jednotka obsahuje paměť na programy a paměť na uložené pozice robota (natočení jednotlivých kloubů).

Programy lze spouštět buď za pomocí tlačítek přímo na řídící jednotce nebo pomocí Teaching device či pc. K řídící jednotce lze připojit některá přídavná zařízení např. karta pro komunikaci přes COM port nebo karta pro ethernetové připojení.

Řídící jednotka je zobrazena na obrázku 3.

Popis obrázku je následující. 1 - vypínač. Tímto vypínačem se udává do provozu nebo vypíná robotické rameno s řídící jednotkou. 2 - toto tlačítko spouští program, který byl vybrán a je zobrazen na obrazovce. 3 - toto tlačítko okamžitě zastaví rameno, přičemž nevypne servomotorky (v kloubech ramene). 4 - restartuje rameno, hodí se obzvlášť při chybách, zároveň se restartuje i probíhající program. 5 - okamžitě zastaví činnost ramene a vypne servomotorky. 6 - toto tlačítko slouží k připojení či odpojení Teaching device za provozu. 7 - mění informaci, která je zobrazena na obrazovce. Na obrazovce se postupně objevuje - rychlost (procentuální oproti maximu), číslo programu a číslo řádku v programu. 8 - na konci příštího opakování prováděného programu zastaví činnost ramene. 9 - zapíná servo. 10 - vypíná servo. 11 - display, zobrazuje číslo chyby, která nastala, případně, číslo prováděného programu, rychlost, atd. 12 - konektor pro připojení Teaching device. 13

- připojení pro počítač přes COM port. 14 - mód ve kterém řídící jednotka pracuje. 15 - tlačítko sloužící pro listování v programech a nastavování rychlosti.

Přepínač číslo 14 je potřeba mít při práci správně nastavené. Jsou tři možnosti. Auto (Op.), ovládat rameno je možné jen s pomocí řídící jednotky. Nelze použít ani Teaching device ani počítač. Teach - v tomto módu je řízení možné pouze pomocí Teaching device. Auto (Ext.) - robot se dá ovládat pouze z externího zařízení (tzn. počítače).

Řídící jednotka je schopna spustit program, který má v paměti. Tento způsob se ovšem v naší práci nijak nepoužívá, tudíž se o něm zmiňovat nebudu.

4.2 Teaching device

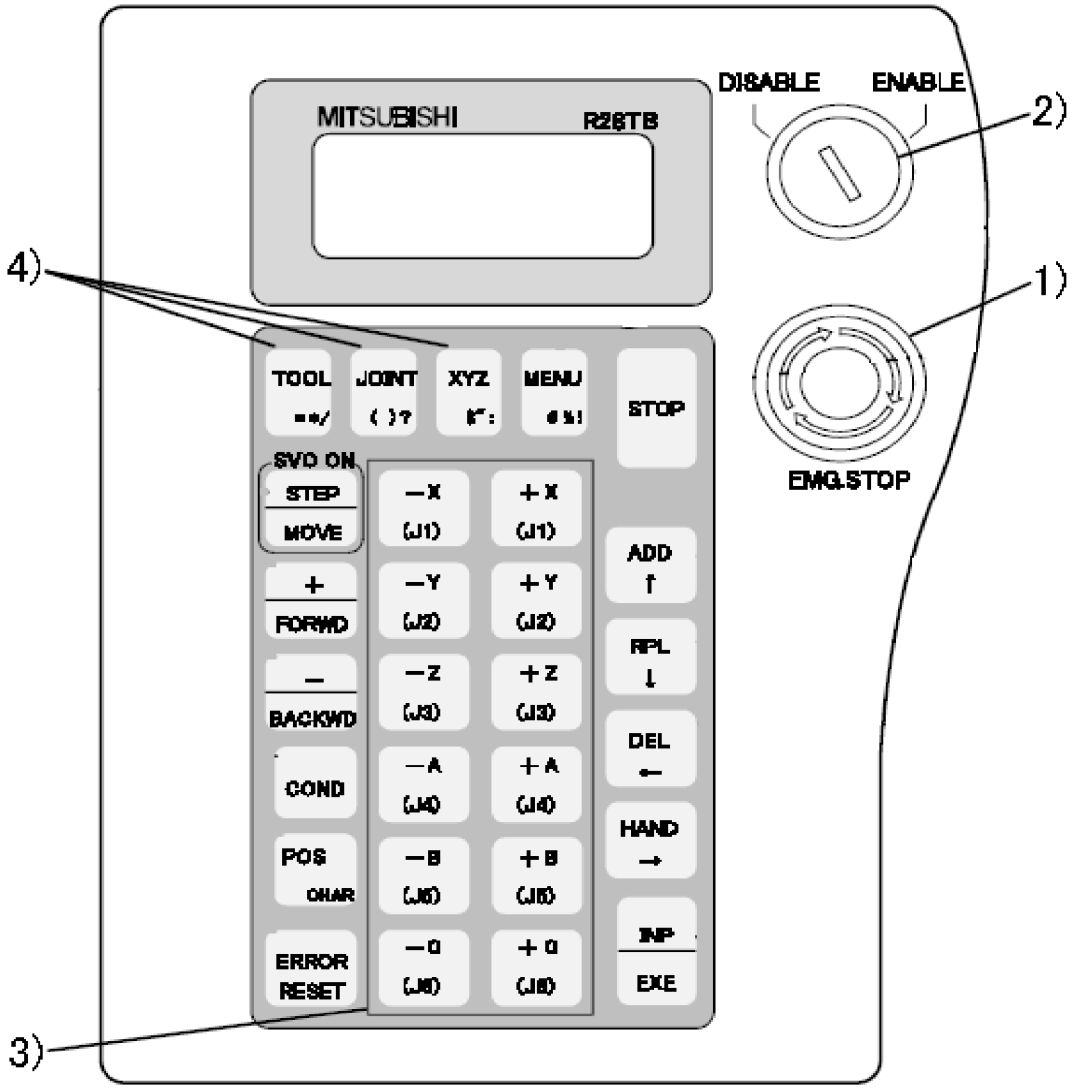
Teaching device (dále jen TD) je ovladač přímo připojený k řídící jednotce. Při používání TD je potřeba mít přepínač řídící jednotky v pozici teach a přepínač na samotném TD musí být nastaven do pozice enable (číslo 2 na obrázku 4).

S pomocí TD je možné psát programy v jazyce Melfa-Basic IV, dále je možné pohybovat s ramenem a to ve třech módech. Buď se robot pohybuje v souřadných osách (ať už polárních nebo ortogonálních) nebo pomocí otáčení jednotlivých kloubů a nebo pohybem v souřadné soustavě nástroje. Mezi těmito módy se přepíná pomocí tlačítek 4 na obrázku 4.

Použití TD pro pohyb je následující. Nejdříve je potřeba stisknout spodní pojistku do střední polohy (toto je zde kvůli bezpečnosti pracovníka) dále pak je potřeba stisknout klávesu SVO ON, čímž se zapnou serva. Dále již jen pomocí tlačítek (na obrázku 3 jako číslo 3) pohybujeme robotem podle nastaveného módu pohybu.

Pomocí TD lze vytvářet programy pro rameno, tato možnost je ovšem zbytečná, máme-li připojený počítač s potřebným softwarovým vybavením.

Na TD se nachází tlačítko Emergency stop (číslo 1). Při různých experimentech toto tlačítko dobře poslouží jako záchrana před poškozením majetku či zdraví.

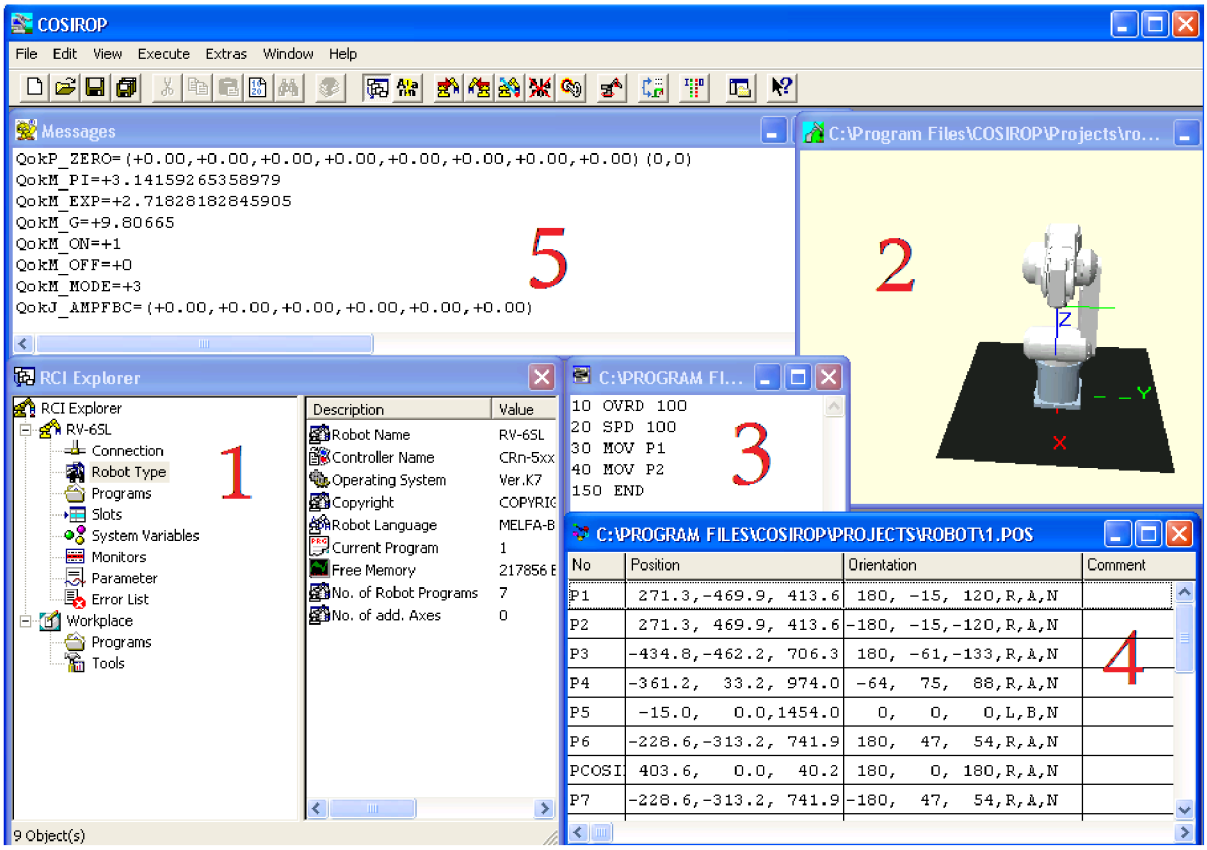


Obrázek 4: Teaching device (obrázek převzat z [1])

4.3 Program Cosirop

Cosirop je program od firmy EF-Robotertechnik GmbH, vyrobený pro firmu Mitsubishi Electric. Slouží k práci a správě robotů více typů. Pomocí tohoto nástroje jsem se seznamoval s ovládáním ramene, parametry jeho nastavení a hlavně jsem se díky tomuto programu dozvěděl, jak lze komunikovat s ramenem pomocí COM portu.

Po spuštění je potřeba vytvořit nový projekt. Po zadání názvu je nutné vybrat typ robota, který je připojen k PC. Na výběr jsou robotická ramena Mitsubishi typu RV, RH a RP různých velikostí a různého počtu kloubů. Výběrem typu ramene se vytvoří projekt, s jehož pomocí můžeme vytvářet programy, nastavovat parametry a zjišťovat stav robota.



Obrázek 5: Hlavní okno programu Cosirop

Na obrázku 5 můžeme vidět typický obsah okna po otevření projektu. Podokno s číslem 1 je hlavní. Jsou v něm obsaženy veškeré informace o ramenu, které jsou umístěny do stromu souborů. Hlavní složka se jménem *RCI Explorer* obsahuje složku pro každého robota, který je v projektu zahrnut (v našem případě je to složka *RV-6SL).* Dále je zde složka nazvaná Workplace. Ta obsahuje složku s programy a složku obsahující různé nástroje pro práci s robotem. Nakonec jsou v této složce umístěny soubory, které informace o projektu, jako kdo projekt vytvořil, popis projektu atd.

Ve složce *RV*-6SL se nachází složky týkající se našeho ramene. Tyto informace jsou uloženy přímo v řídící jednotce a při otevření složky jsou staženy. Podsložka *Connection* obsahuje informace o připojení pc k řídící jednotce. Zobrazuje typ spojení (v našem případě je to spojení přes kabel s koncovkou RS232). Dále zobrazuje stav spojení (spojeno nebo nespojeno). Když stiskneme nad složkou *Connection* pravé tlačítko myši, objeví se kontextové menu, ve kterém nás nejvíce zajímjí nabídka *Properties* a nabídka *Connect.*

Před zahájením práce je nejdříve nutné se připojit k řídící jednotce. Výběrem *Properties* zjistíme, že se lze pomocí programu Cosirop připojit jak přes COM port tak i přes ethernetové spojení. Při komunikaci přes ethernetovou síť je nutné zadat IP adresu robota a port, na kterém bude

naslouchat. Případně je možné nastavit v řídící jednotce pracovní jméno ramene a to poté používat při spojení s ním. Z důvodu jednoduššího dekódování komunikace jsem se rozhodl pro komunikace přes COM port. Nastavení spojení je popsáno v tabulce 1. Je-li nastavení spojení hotové, můžeme se výběrem *Connect* z kontextové nabídky složky *Connection* spojit s řídící jednotkou. Poté se otevře okno s bezpečnostním upozorněním pro práci s ramenem a dále ještě několik informací z řídící jednotky. Stisknutím tlačítka *OK* se okno zavře a my můžeme pracovat.

Port	COM6
Baud Rate	19.2k
Data Bits	8
Parity	Even
Stop Bits	2
Flow Control	DTR, RTS/CTS

Tabulka 1: Nastavení spojení přes COM port

Další složka je *Robot Type*. Ta obsahuje informace o ramenu. Nalézá se zde informace o názvu robota, typu řídící jednotky, ke které je připojen, programovací jazyk, který používá atd. Některé z uvedených informací jsou již v této publikaci zmiňovány a zbylé nejsou pro moji práci potřebné.

Ve složce *Programs*, se nalézají všechny programy, uložené v řídící jednotce. Tyto programy lze přímo spouštět. Po stisku pravého tlačítka myši nad souborem s programem, který chceme aby se provedl, vybereme nabídku *Start (CYC)* pro spuštění programu pouze jednou, nebo nabídku *Start (REP)* pro spuštění programu v nekonečné smyčce. Program zastavíme výběrem nabídky *Stop*. Pokud bychom chtěli soubor s programem uložit na počítači, můžeme to provést stiskem tlačítka *Upload* z kontextové nabídky.

Složky *Slot*, *System Variables* a *Monitors*. Jsem při své práci s ramenem nepotřeboval, tudíž je tu popisovat nebudu a přejdu na další, velice důležitou složku *Parameter*. V této složce jsou uloženy všechny parametry řídící jednotky a ramene. Po dvojím rychlém stisknutí levého tlačítka myši nad parametrem se objeví okno, které zobrazuje aktuální hodnotu a popis a dále pak místo pro zadání nové hodnoty. Chceme-li některý parametr změnit, musíme po přepsání jeho hodnoty restartovat řídící jednotku. Některé důležité parametry, potřebné pro mou práci jsou popsány v kapitole 3.1.2. Nastavení.

Poslední podsložkou *RV-6SL* je *Error List*. Obsahuje výčet chybových stavů, které se v minulosti vyskytly. U každého záznamu o vzniklé chybě je uvedeno datum a čas výskytu, číslo, které je přiřazené označuje vzniklou chybu a nakonec popis chyby. V tabulce 2 jsou vypsány nejčastější chyby, které se během práce objevovaly, zpráva je přeložena do češtiny (řídící jednotka ji ovšem posílá v angličtině).

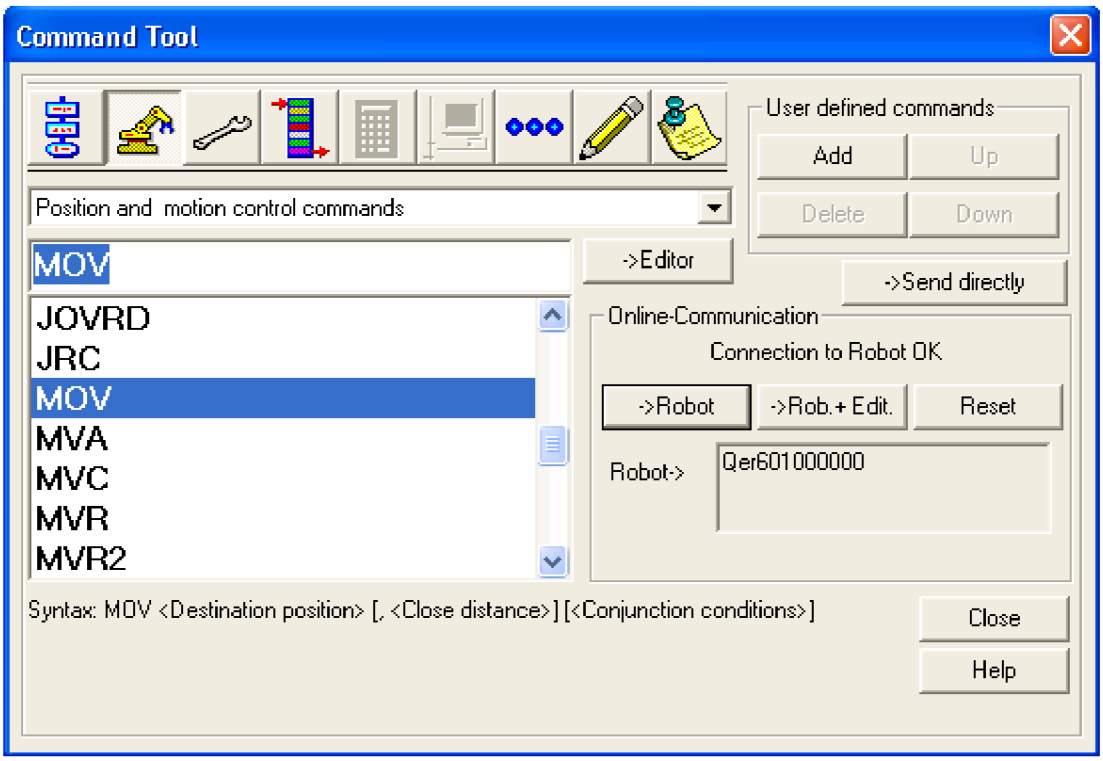
Číslo chyby	Zpráva	Příčina
60	EMG signál je aktivní	Je stisknuto tlačítko EMG Stop na řídící jednotce
70	EMG signál je aktivní	Je stisknuto tlačítko EMG Stop na teaching device
211n	Free plane číslo n překročena	Pokus o překročení plochy n-té plochy free plane limit
2800	Špatná pozice	Požadovaná pozice nelze dosáhnout

Tabulka 2: Nejčastější chyby (převzato z [6])

Ve složce Workplace jsou pouze dvě podsložky - *Programs* a *Tools*. *Programs* obsahuje soubory typu .MB4 a .POS. Jsou to soubory se zdrojovými kódy napsané v jazyce Melfa-Basic IV (MB4) a s pozicemi ramene (POS). Soubory obou typů jsou ukládány v textové podobě na lokální disk. Jak vypadá zdrojový kód v jazyce Melfa-Basic IV popíšu v kapitole 4.4. Soubor s pozicemi obsahuje definice pozic zapsané podle syntaxe jazyka Melfa-Basic IV.

Chceme-li nahrát soubor se zdrojovým kódem do řídící jednotky, stiskneme pravé tlačítko myši nad požadovaným souborem a z nabídky, která se objeví vybereme *Download*. Totéž můžeme provést i s pozicemi. Pokud používáme ve zdrojovém kódu nějaké pozice, musí být tyto pozice nadefinovány v souboru se stejným jménem (ovšem jinou koncovkou), jako má soubor s kódem.

V poslední složce, *Tools,* jsou nástroje pro práci s robotem. Jediné dva nástroje, které jsem používal byly *Command Tool* a *Jog Operation*. *Command Tool* je nástroj, který má za úkol usnadňovat psaní programů. Umí zobrazit syntaxi příkazů. Umí příkazy vkládat do zrovna otevřeného editoru kódu či přímo je přímo posílat do řídící jednotky k provedení. *Command Tool* je zobrazen na obrázku 6.



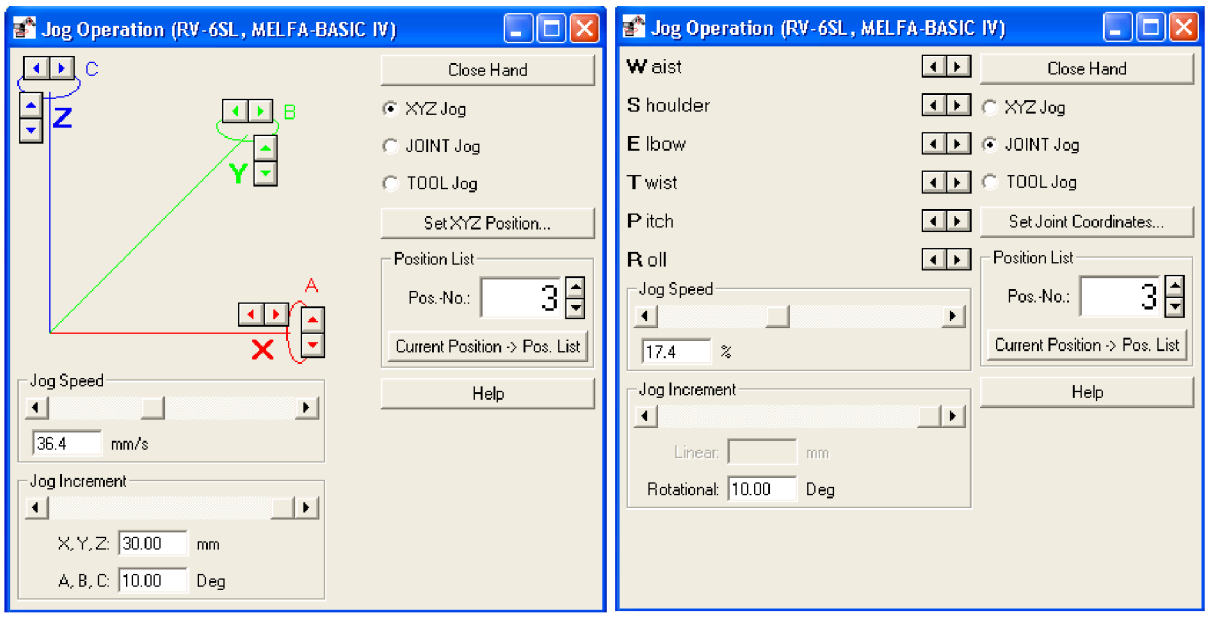
Obrázek 6: Okno nástroje Command Tool

Druhý zmiňovaný nástroj jsem používal při své práci velice často, proto jej zde popíšu trochu více. *Jog Operation* slouží k nastavování pozice ramene. Pozice lze nastavit přímo zadáním souřadnic (resp. natočením jednotlivých kloubů) nebo postupným posunováním o zvolenou vzdálenost (resp. o zvolený úhel). Na obrázku 7 jsou vidět obě varianty pohybu, tedy posouvání v souřadnicové soustavě (okno nalevo) a posouvání jednotlivých kloubů (okno napravo).

V pravém horním rohu pod tlačítkem *Close Hand* (které nemá nyní žádný význam) si můžeme vybrat jeden ze tří způsobů pohybu ramenem. Možnosti *XYZ Jog* a *TOOL Jog* se liší pouze souřadnou soustavou, kterou používají (viz. kapitola 3.1.1). Poté je nutné nastavit rychlost pohybu. To lze provést nastavením posuvníku *Jog Speed*, či přímo zadáním hodnoty do textového pole o kus níže. Znak procenta v nastavení rychlosti u *JOINT Jog* udává, jak velkou část z maximální rychlosti servomotoru daného kloubu se má rameno pohybovat. U *XYZ Jog* a *TOOL Jog* se nastavuje rychlost v mm/s. Dále je nutné nastavit vzdálenost (resp. úhel), o který se má robot posunout. Lze to provést stejně jako u nastavení rychlosti nastavením posuvníku *Jog Increment*, nebo přímo zadáním hodnoty v jednotkách uvedených za textovým polem.

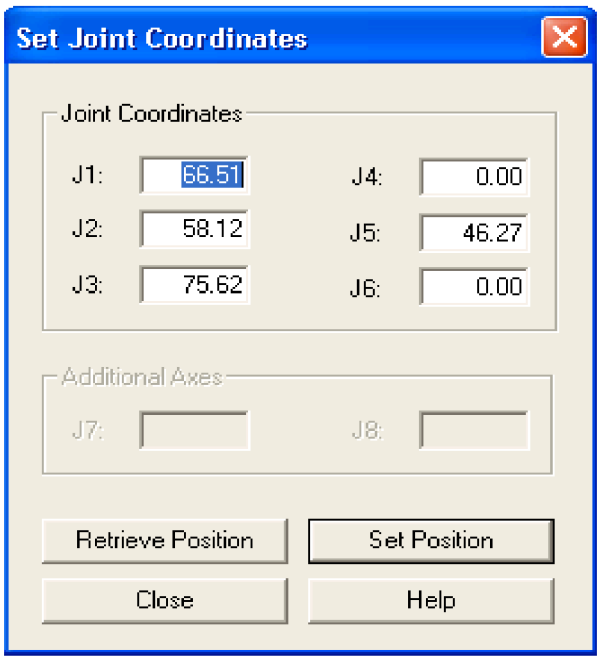
Je-li vše nastaveno, stačí pak pouze stisknout šipku u požadovaného typu pohybu. Při použití *XYZ Jog* (resp. *TOOL Jog)* se rameno pohybuje po osách X, Y a Z a nebo kolem nich rotuje (což udávají souřadnice A, B a C). V případě *JOINT Jog* můžeme natáčet jednotlivé klouby (*Waist* označuje kloub číslo jedna a *Roll* kloub číslo 6). Tímto způsobem nastavenou pozici robota poté

můžeme přímo uložit do definičního souboru pozic. Stačí nastavit *Pos.-No.,* což bude číslo pozice (pozice se tedy bude jmenovat např. P3) a poté stisknout tlačítko *Current Position → Pos. List*.



Obrázek 7: Okno nástroje Jog Operation

Po stisknutí tlačítka *Set Joint Coordinates* se objeví okno, které je na obrázku 8. Tento nástroj umožňuje přímo zadat nastavení všech kloubů ramene (případně souřadnic) a stisknutím tlačítka *Set Position* pak poslat příkaz k provedení. Tlačítko *Retrieve Position* slouží k načtení aktuálního nastavení ramene z řídící jednotky. Tento nástroj ovšem může být velice nebezpečný. Při pokusu o nastavení robota na určitou pozici (zadanou pomocí souřadnic) se může stát, že se rameno pokusí při přemisťování projít samou sebou, což by vedlo k jeho poškození. Tento způsob pohybu je tedy nevhodný při vzdálené obsluze. V případě, že je to nutné, je lepší nastavit robota postupně pomocí šipek, jak je uvedeno výše, nebo s použitím natáčení kloubů.



Obrázek 8: Okno pro nastavení pozice robota

Nyní se vrátím k popisu obsahu okna na obrázku 5. Číslem 2 je zde označeno okno s vyobrazením ramene. Slouží pouze pro ilustrační účely. Jediné využití, které jsem pro tento nástroj našel je kontrola, jestli se rameno při cestě mezi dvěma body nebude snažit projít skrze sebe. Lze totiž nastavit souřadnice pozice a model pak ukáže jak bude toto nastavení vypadat. Je tedy možné vizuálně zkontrolovat, kudy by vedla přímá cesta.

Číslo 3 na obrázku označuje textový editor pro psaní programů. Jazykem Melfa-Basic IV se zabývá kapitola 4.4. Je to obyčejný editor, kódy v něm lze prohlížet, upravovat a ukládat. Cosirop podporuje psaní kódu několika nástroji. Je jím například výše zmiňovaný nástroj *Commnad Tool*, který umožňuje vyhledávat příkazy a přímo je vkládat do editoru. Další usnadnění je v možnosti přečíslování řádků. Najdeme jej v menu na horní liště, nabídka *Edit* a v ní zvolíme *Renumber*, případně stiskneme klávesy *ctrl* a *R* zároveň. Při stisku pravého tlačítka myši nad editorem se objeví nabídka s příkazy jazyka, což usnadňuje hledání potřebných příkazů.

Okno, které zobrazuje zaznamenané pozice ramene je na obrázku 5 označeno číslicí 4. Dvojklikem na pozici se tato pozice nastaví na ramenu zobrazeném v okně číslo 2. Pokud chceme uloženou pozici změnit, musíme na ni kliknout pravým tlačítkem myši a zvolit *Properties*. Nyní můžeme vepsat potřebné souřadnice.

Poslední okno, označené číslicí 5, obsahuje zprávy, které zaslala řídící jednotka. Děje se tak jako odpověď na některé naše akce. Většinou se jedná pouze o potvrzení příkazu *Qok*.

4.4 Jazyk Melfa-Basic IV

V této kapitole popíšu základy jazyka, který je používán k tvorbě programů pro některé roboty Melfa. Nejdříve řeknu několik obecných slov o tomto jazyku. Napíšu zde, jak by měl vypadat soubor s programem a s uloženými pozicemi. Popíšu syntaxi základních konstrukcí. Dále pak zde ukážu funkce, které jsem při své práci používal a které sou povětšinou i součástí knihovny, kterou jsem měl za úkol vytvořit. Mnohé možnosti a konstrukce jazyka jsem nepotřeboval při své práci. Programy, které lze vytvořit pomocí tohoto jazyka můžou být mnohem složitější. Má například podporu pro multitasking. Při návrhu mého projektu i při návrhu „velkého projektu“, jehož je má práce částí, nepočítám s tím, že by se tento jazyk využil plně. Předkládám zde proto jen tu část, která bude použita.

4.4.1 Charakteristika jazyka

Programovací jazyk Melfa-Basic IV je používán pro robotická ramena firmy Mitsubishi. Tento jazyk je interpretovaný. Řídící jednotka čte postupně jednotlivé příkazy a okamžitě je provádí. Díky tomuto přístupu lze posílat řídící jednotce pouze jednotlivé příkazy, které jsou okamžitě provedeny.

Z pohledu míry abstrakce se jedná o vyšší programovací jazyk. Některé jeho příkazy (například `MOV`) způsobí provedení několik akcí (u `MOV` je to nastavení všech kloubů ramene na požadovanou hodnotu, kterou vypočítá řídící jednotka).

Melfa-Basic IV je jazyk procedurální. To znamená, že program je tvořen posloupností příkazů, které jsou prováděny postupně. Dále je tento jazyk strukturální. Syntaxe jazyka tedy obsahuje řídící struktury jako jsou příkazy větvení (`IF THEN ELSE`), příkazy cyklu (`FOR` a `WHILE`) a příkazy pro volání podprocedur (`GOSUB` a `CALLP`). Jazyk ovšem obsahuje i příkaz skoku (`GOTO`).

4.4.2 Syntaxe jazyka

V tomto jazyce je nutné číslovat řádky. Nejvyšší možné číslo, které se může objevit jako pořadí řádku je 32767, nejnižší 1. Tímto je daný maximální rozsah kódu. Pro naše účely je to více než dostačující. V případě potřeby delšího programu by bylo nutné jej rozdělit na více částí a ty pak postupně spouštět. Dalším omezením je délka jedné řádky. Ta může obsahovat nejvýše 127 znaků (bez znaku konce řádku). Na každém řádku se může nacházet pouze jeden příkaz (*Command statement*).

Za číslem řádky následuje příkazové slovo, některé příkazy popíši v další kapitole. Případné parametry jsou odděleny od příkazového slova mezerou. Za parametry může ještě následovat tzv. přidaný příkaz (*Appended statement*). Tyto příkazy ve své práci nepoužívám. Jsou to příkazy, které se spouští zároveň s akcí, kterou chceme provést. Jsou uvedeny vyhrazenými slovy *WTH* a *WTHIF*.

Používají se například pro nastavení signálu při pohybu robota. Tímto mechanizmem lze docílit, aby se v daném prostoru nepohybovaly dva roboty zároveň.

Program můžeme psát všemi znaky anglické abecedy (jak velkými tak i malými) a číslicemi. Dále pak jsou povoleny znaky interpunkce, znaky pro porovnání, aritmetické výpočty, závorky jednoduché, složené i hranaté atd. Celý výčet znaků je uveden v tabulce 3.

Třída	Povolené znaky
Alfabetické znaky	ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
Číslice	1234567890
Symboly	" ' & ( ) * + - . , / : ; = < > ? @ ` [ ] \ ^ { } ~ | ! # $ % _
Mezery	mezera

Tabulka 3: Povolené znaky jazyka Melfa-Basic IV

Pravidla pro názvy jsou následující. Jméno programu se může skládat pouze z velkých písmen a číslic a může mít nanejvýš dvanáct znaků. Pokud ovšem je název delší než čtyry znaky nelze takový program spustit přímo z řídící jednotky. Název proměnné musí začínat písmenem. Zároveň se může skládat pouze z velkých znaků a číslic a ještě ze znaku podtržítka. Jestli použijeme malé znaky, budou automaticky překonvertovány na velké.

Programy by nešly psát bez proměnných. V jazyce Melfa-Basic IV existuje několik typů proměnných. Proměnné, jejichž názvy nezačínají písmeny P, C a J, se nazývají číselné (*Numeric value variables*). Pokud chceme určit typ proměnné, připojíme na konec jejího názvu znak % (celá čísla), ! (reálná čísla, tento typ je implicitní) a nebo # (reálná čísla s dvojnásobnou přesností). Jednou zaregistrovaný typ proměnné může být změněn pouze z celého čísla na reálné číslo s jednoduchou přesností.

Další typ proměnných jsou znaky a řetězce (*Character string variable*). Tyto proměnné začínají písmenem C a měly by končit znakem $. Proměnnou typu znaková proměnná můžeme též nadefinovat pomocí příkazu `DEF CHAR`. V takovémto případě může název začínat i jiným písmenem než C.

Velice důležité jsou poziční proměnné (*Position variables*). Jsou to spolu s proměnnými nastavení kloubů jediné proměnné, které používám ve svém projektu (tzn. při implementaci knihovny). Názvy těchto proměnných začínají písmenem P (stejně jako u znaků lze definovat poziční proměnnou pomocí `DEF POS`, pak název nemusí začínat na P). Poziční proměnné mají osm složek, ty udávají souřadnice konce chapadla a rotaci kolem těchto os. Poslední dvě složky jsou příznaky polohy (viz. kapitola 3.1.1). Souřadnice jsou udávány v milimetrech, natočení v osách je udáváno v radiánech (pokud bychom potřebovali úhel ve stupních, použili bychom funkci `DEG, která je popsána v [1]`). Poslední dvě složky jsou kódovány jako čísla. Stavové data je kódováno jako

osmi bitové číslo, přičemž nejnižší bit je nastaven, pokud není robot překlopený (*NonFlip*), druhý bit je nastaven v případě, že je robot v horní části (*Above*) a třetí bit je nastaven, pokud je robot na pravo (*Right*), v opačných případech jsou bity vynulovány. Takto dostaneme číslo v rozmezí od nuly do sedmi (reálně jsou tedy použity pouze tři bity). Multi-rotační data jsou zakódovány jako 8 hexadecimálních čísel. Nejnižší číslo představuje počet přetočení pro první kloub, nejvyšší pro osmý kloub (poslední dva tedy nejsou třeba).

Proměnné nastavení kloubů (*Joint variables)* jsou podobné předchozím. Místo souřadnic se ovšem udává pouze natočení každého kloubu ve stupních. Tyto proměnné začínají písmenem J (výjimka je opět při definování pomocí `DEF JNT`).

Proměnným lze přiřazovat hodnota pomocí znaku rovno (stejně jako je tomu v jazyce C). Následuje několik příkladů s komenářem:

`M1 = 100`	(proměnná typu reálné číslo)
`M2# = 0.123`	(proměnná typu reálné číslo s dvojnásobnou přesností)
`C1$ = "ABC"`	(proměnná typu řetězec)
`P1 = PORG`	(poziční proměnná)
`M1 = DEG(P1.A)`	(do proměnné M1 přiřadí souřadnici A ve stupních)
`JSTARAT = (0, 0, 90, 0, 90, 0, 0, 0)`	(proměnná nastavení kloubů)
`DEF JNT K10`	(použití příkazu `DEF` pro definování proměnné nastavení kloubů)

Příklady jsem převzal z [1]. Komentáře by do kódu nepatřili, jsou zde pouze na vysvětlení významu. S proměnnými typu pozice a nastavení kloubů lze provádět aritmetické operace. Lze je sčítat a odčítat. Toho můžeme s výhodou využít při relativních pohybech. Díky proměnné `P_CURR` resp. `J_CURR`, ve kterých jsou uloženy aktuální pozice, stačí mít pouze pozici, která udává změnu a tu přičíst k současné pozici.

4.4.3 Funkce

Nyní popíšu příkazy, které jsem během své práce používal a které využívám v knihovně. Při pozorování komunikace programu Cosirop s řídící jednotkou jsem zjistil, že jsou používány příkazy, které nejsou v manuálu, popíšu je zde tak jak se projevují:

- Pohyb jako interpolace kloubů (*Joint interpolation movement*) – jedná se o příkaz `MOV`. Příkaz má jeden parametr a tím je konečná pozice ramene. Rameno se do této pozice dostává postupným otáčením jednotlivých kloubů do požadované hodnoty.
- Pohyb po přímce (*Linear interpolation movement*) – příkaz `MVS`. Tento příkaz pohne robotem na parametrem zadanou pozici. Na rozdíl od příkazu `MOV` se bude rameno mezi počátečním a cílovým bodem pohybovat po přímce. Tento druh pohybu je relativně nebezpečný. Když

bychom chtěli aby se přemístil na bod souměrný podle počátku soustavy (který je uprostřed robota), pokusí se projít přímo skrz sebe, což by vedlo k poškození ramene.

- Nastavení rychlosti pohybu (*speed control*) – k tomuto účelu slouží hned několik příkazů. Pomocí příkazu `OVRD` můžeme ovlivnit rychlost všech pohybů. Tento příkaz má jako argument číslo, které udává jaké procento z maximální možné rychlosti se může použít. Příkaz `JOVRD` se týká pouze pohybu interpolací kloubů (`MOV`). Opět jeho jediný parametr udává procento z maximální rychlosti, které se má použít. Příkaz `SPD` se používá při lineární a kruhové interpolaci pohybu (příkaz `MVS`, kruhovou interpolaci nepoužívám). Parametr udává maximální povolenou rychlost v milimetrech za sekundu.
- Servo – příkaz `SERVO ON/OFF` slouží k sepnutí a vypnutí serv. Tento příkaz je vhodný pro okamžité zastavení pohybu ramene.
- Restartování alarmu – příkaz `RSTALRM` je první z manuálem nepopsaných. Při zaslání tohoto příkazu řídící jednotce se vypne alarm a ukončí se chybový stav.
- Chybový stav – funkce `ERROR` slouží ke zjištění, zda-li nenastala chyba. Odpovědí je buď *Qok*, když je vše v pořádku, nebo Qer a číslo chyby, nastala-li v běhu ramene chyba.
- Chybová zpráva – pošleme-li řídící jednotce příkaz `ERRCONT<číslo chyby>`, dostaneme jako odpověď zprávu, která chybu popisuje.
- Zjištění pozice – příkazy `PPOSF` a `JPOSF` slouží ke zjištění prostorového nastavení robota. `PPOSF` vrací souřadnice, kde se právě robot nachází a `JPOSF` vrací natočení jednotlivých kloubů.

4.5 Počítač

Cílem mého projektu je ovládat robotické rameno přes počítač. Ten je k řídící jednotce připojen přes COM port. Nejprve bylo potřeba zjistit, jak počítač komunikuje s řídící jednotkou. Využil jsem toho, že k robotickému rameni je dodáván program Cosirop, který jsem již dříve popsal.

Pro to, abychom zjistili komunikační protokol mezi pc a řídící jednotkou jsem odposlouchával komunikaci přes COM port. Toto sledování mi umožnil program PortMon, který je volně dostupný na internetu. PortMon umí na zvoleném COM portu zaznamenávat proudící informace. Sledoval jsem tedy datové proudy mezi počítačem a řídící jednotkou při práci s Cosiropem. Po analýze průchozích zpráv jsem zjistil, že protokol je textový a že příkazy zasílané ramenu jsou téměř totožné s jazykem Melfa-Basic IV.

Komunikace s robotem začíná ustavením spojení přes COM port. Dále proběhne inicializace pracovního prostoru (myslím tím v paměti kontroléru). Poté řídící jednotka pošle informace o sobě

a o připojeném ramenu. Na konec inicializační části se Cosirop ptá na polohu ramene a jeho nynější konfiguraci.

Příkazy zasílané přes COM port, které chceme aby se provedly vypadají následně:

```
1;1;EXEC<příkaz v jazyce Melfa-Basic>
```

např.:

```
1;1;EXECJ
1;1;EXECMOV P1
```

Při studiu záznamů komunikace a při pokusech vyvolat potřebné zprávy jsem zjistil, že některé příkazy, které používá program Cosirop nejsou v manuálu popsány. Bylo tedy třeba pro každou situaci zjistit, jak na ni Cosirop reaguje a zjistit, které příkazy při tom používá. Příkladem je například ovládání chybových stavů. To používá příkaz `ERROR`. Ten ovšem v manuálu [1] není popsán. Sledováním a postupným zkoušením jsem získával informace potřebné pro tvorbu knihovny.

5 Knihovna pro ovládání RR

V této kapitole se dostávám ke zpracování knihovny. Popíšu návrh knihovny, co by měla obsahovat, aby byla provozuschopná. Dále zde napíšu o implementaci knihovny, ta je provedena v jazyce Python, jehož stručný popis bude v této kapitole také obsažen. Nakonec kapitoly popíšu dva programy, které knihovnu využívají pro práci s robotem. První z nich je příkazová řádka, ze které lze robota ovládat pomocí jednoduchých příkazů. Druhý popsaný program je aplikace, která využívá ramene jako nástroj pro kreslení obrázků. Oba dva programy byly využity pro testování knihovny.

5.1 Návrh knihovny

Knihovna pro ovládání robotického ramene musí mít prostředky pro řízení pohybů ramene, odhalení chyb, připojení a komunikaci s robotem, zjišťování pozice ruky a pro rychlé zastavení v případě nebezpečí (tzv. softwarový emergency stop).

Protože má práce bude součástí většího projektu, bude potřeba vytvořit knihovnu, která by splňovala výše uvedené podmínky a byla by lehce importovatelná do většího projektu, kde by měla na starosti právě ovládání robotického ramene.

Jako dobré řešení se mi jeví objektový návrh. Knihovna by pak obsahovala objekt, který by byl abstrakcí ramene. Jako atributy jsou potřeba informace o poloze robota. Protože robot bude pracovat s chapadlem nastaveným kolmo k zemi, nebude potřeba pamatovat si natočení okolo os (to jest

parametry A, B a C). Bude tedy potřeba uchovávat informaci o pozici v souřadnicové soustavě. Dále si můžeme ukládat nastavení jednotlivých kloubů. Ty umožní snadnější verifikaci dalšího kroku (ať už bezpečnost akce, či pouze zjištění, jestli daná akce lze provést) a díky informaci o aktuální poloze můžeme provádět relativní pohyby. Relativním pohybem myslím, posun v souřadných osách o určitou vzdálenost. Protože známe aktuální pozici, stačí pak pouze přičíst hodnoty, o které se chceme posunout k nynější pozici a pak poslat robotu příkaz k přesunutí na konečné místo. Další atribut by mohla být informace o chybovém stavu. Pokud se jedná o chybu, kterou nelze napravit vzdáleně, je dobré mít o tom informaci.

Základní úlohou třídy robot by mělo být připojení se k ramenu. Nejdříve je nutné připojit se ke COM portu, přes který se bude komunikovat s řídící jednotkou. Je potřeba zvolit číslo COM portu. Dále ustavit spojení podle parametrů uvedených v tabulce 2 na straně 12. Po spojení je potřeba provést inicializační část. Protože se mi nepodařilo zjistit funkci všech příkazů, které používá program Cosirop po spojení, budu v knihovně používat posloupnost podobnou. Po spojení by bylo dobré nastavit rychlost pohybu. Rychlost pohybu bude třeba omezovat. Kdyby rychlost byla příliš velká, bylo by nebezpečí, že toho budou pozdější uživatelé zneužívat (v konečné fázi projektu budou s robotem pracovat studenti).

Protože budeme komunikovat po sériovém rozhraní, je třeba mít metodu, která bude zasílat a přijímat zprávy. Problémem při komunikaci s řídící jednotkou je asynchronnost. Řešení je nastavit čas, po který se bude čekat na případnou odpověď. Tato metoda též bude kontrolovat příchozí zprávy, abychom měli možnost, jak zjistit, jestli nenastala chyba.

Další metody třídy by měly odpovídat akcím, které rameno může provádět. Knihovna tedy musí obsahovat metody, které budou robotem hýbat. Z celé škály možností, jak pohybovat robotem jsou pro účely projektu podstatné jen některé. Za použitelné považuji příkaz pro interpolaci jednotlivých kloubů (tedy příkaz `MOV`) a příkaz pro interpolaci souřadnic (to znamená příkaz `MVS`). Pro pohyby v rámci souřadnicových os, což bude pravděpodobně výhodnější při součinnosti s kamerami a přibližování se k cíli, by mohla být používána interpolace souřadnic. Pro bezpečnější přemístění (nikoliv ovšem úplně bezpečné) bude používána interpolace kloubů. U tohoto pohybu nehrozí, že by se rameno rozhodlo projít skrze sebe. Obě varianty by měly mít možnost relativního a absolutního pohybu. To znamená, že by mohl být zadán příkaz k přesunu na konkrétní pozici, nebo povel k posunutí o určenou vzdálenost. V absolutním pohybu ovšem vidím nebezpečí poškození robota. Pokud by byl uživatel nepozorný mohl by lehce poškodit robota zadáním pozice, při níž by přímá trasa vedla přes rameno samo (týká se příkazu `MVS`). Jako doplňkovou možnost jsem navrhl přesun pouze po jedné ose, či otočení jedním kloubem.

Dále jsou potřeba metody pro správu alarmu. Je potřeba zjistit, zda-li se vyskytla nějaká chyba. Pokud se tak stalo, bylo by dobré mít možnost zjistit o jakou chybu se jedná a co je potřeba provést

k jejímu odstranění. Důležité je chybu detekovat. Pomocí programu PortMon jsem zjišťoval, jaký příkaz používá program Cosirop. Podobně jsem postupoval i v případě restartování alarmu a zjištění popisu vzniklé chyby.

Abychom mohli udržovat třídní proměnné (pozice a natočení kloubů) je potřeba mít metody, které by toto zajišťovaly. Opět jsem musel experimentální cestou zjistit, který příkaz umí řídící jednotku požádat o tyto informace.

Velice potřebnou funkcí, kterou by měla poskytovat knihovna, je náhražka tlačítka *Emergency stop*. Toto softwarové „červené tlačítko“ by mělo sloužit k okamžitému zastavení ramene. Jako nejrychlejší možnost se mi jeví použít příkaz k vypnutí servomotorů. Tím se zamezí ramenu v jakémkoli pohybu. Tento příkaz se provede okamžitě, jak je přijat řídící jednotou a prodlení mezi odesláním příkazu uživatelem a zastavením akce nebude velké.

Poslední metodou, kterou jsem zde ještě neuvedl je konstruktor. Ten bude obsahovat ustavení spojení na zadaném portu (nepovinně) a dále bude mít za úkol inicializaci třídních atributů.

5.2 Implementace

Jak jsem již na začátku kapitoly uvedl, implementace knihovny bude provedena v jazyce Python. Tento jazyk je dostatečně pro tento projekt dostatečně silný a po domluvě s kolegou Josefem Skládankou jsme se rozhodli právě pro něj. Pro knihovnu, která je cílem mé práce je důležité, že Python je jazyk objektový. Tato podkapitola popíše stručně jazyk Python a dále implementaci samotné knihovny.

5.2.1 Programovací jazyk Python

Python je interpretovaný objektově orientovaný programovací jazyk, který v roce 1990 navrhl Guido van Rossum. Python je vyvíjen jako open source projekt, který zdarma nabízí instalační balíky pro většinu běžných platforem (Unix, Windows, Mac OS); ve většině distribucí systému Linux je Python součástí základní instalace.

Python je dynamický interpretovaný jazyk. Někdy bývá zařazován mezi takzvané skriptovací jazyky. Jeho možnosti jsou ale větší. Python byl navržen tak, aby umožňoval tvorbu rozsáhlých, plnohodnotných aplikací (včetně grafického uživatelského rozhraní - viz například wxPython, který využívá wxWidgets).

Python je hybridní jazyk (nebo také víceparadigmatický), to znamená, že umožňuje při psaní programů používat nejen objektově orientované paradigma, ale i procedurální a v omezené míře i funkcionální, podle toho komu co vyhovuje nebo se pro danou úlohu hodí nejlépe. Python má díky

tomu vynikající vyjadřovací schopnosti. Kód programu je ve srovnání s jinými jazyky krátký a dobře čitelný.

K význačným vlastnostem jazyka Python patří jeho jednoduchost z hlediska učení. Bývá dokonce považován za jeden z nejvhodnějších programovacích jazyků pro začátečníky. Tato skutečnost je dána tím, že jedním z jeho silných inspiračních zdrojů byl programovací jazyk ABC, který byl jako jazyk pro výuku a pro použití začátečníky přímo vytvořen. Python ale současně boural zažitou představu, že jazyk vhodný pro výuku není vhodný pro praxi a naopak. Podstatnou měrou k tomu přispívá čistota a jednoduchost syntaxe, na kterou se při vývoji jazyka hodně dbá.

Význačnou vlastností jazyka Python je produktivnost z hlediska rychlosti psaní programů. Týká se to jak nejjednodušších programů, tak aplikací velmi rozsáhlých. U jednoduchých programů se tato vlastnost projevuje především stručností zápisu. U velkých aplikací je produktivnost podpořena rysy, které se používají při programování ve velkém, jako jsou například přirozená podpora jmenných prostorů, používání výjimek, standardně dodávané prostředky pro psaní testů (unit testing) a dalšími. S vysokou produktivností souvisí dostupnost a snadná použitelnost široké škály knihovních modulů, umožňujících snadné řešení úloh z řady oblastí.

Python se snadno vkládá do jiných aplikací (embedding), kde pak slouží jako jejich skriptovací jazyk. Tím lze aplikacím psaným v kompilovaných programovacích jazycích dodávat chybějící pružnost. Jiné aplikace nebo aplikační knihovny mohou naopak implementovat rozhraní, které umožní jejich použití v roli pythonovského modulu. Jinými slovy, pythonovský program je může využívat jako modul dostupný přímo z jazyka Python (tj. extending).

Programování v Pythonu klade velký důraz na produktivitu práce programátora. [5]

5.2.2 Použití knihovny

Výsledný produkt popisuje programová dokumentace melfa.html, která se nachází v příloze. Knihovna potřebuje pro správný běh knihovnu *serial*, která není standardní součástí jazyka Python. Knihovna *serial* je dostupná na počítači v laboratoři. Dále je potřeba, k počítači byla připojena řídící jednotka přes kabel s koncovkou RS232 (tedy ke COM portu). Knihovnu do programu vložíme tímto příkazem:

```
from melfa import Melfa
```

knihovna melfa.py se musí nacházet ve stejném adresáři, jako program, který ji chce využívat.

Instanci Melfa vytvoříme příkazem

```
robot = Melfa(port)
```

Tímto příkazem vytvoříme instanci, která bude komunikovat s řídící jednotkou na COM portu, který udává parametr konstruktoru `port`. Pokud port nezadáme, vyhledá program porty, ke kterým je možné se připojit a my pak musíme vybrat ten, ke kterému je řídící jednotka připojena.

5.2.3 Implementace

Nyní se zmíním o řešení asynchronnosti komunikace. Při posílání zpráv pomocí metody *sendAndReceive* čekáme většinou nějakou odpověď od řídící jednotky. Čas, do kterého by měla jednotka odpovědět není znám. Je tedy nutné nastavit nějakou čekací dobu. V knihovně je to globální proměnná `TIMEOUT`. Po odeslání zprávy postupně vybíráme zprávy z bufferu a ukládáme do pole `data`. Zkontrolujeme zda nám jednotka neposlala oznámení o chybě (pokud ano, příjem končí). Mezi jednotlivými výběry čekáme po určitou dobu, až celková doba převýší hodnotu `TIMEOUT`, skončíme. Zkontrolujeme, jestli bylo něco přijato, pokud ne, tak vrátíme řetězec „TIMEOUT“, který dá uživateli vědět, že do vypršení času pro příjem nedošla žádná zpráva. Jinak vrátíme přijatý řetězec, který pak vypíšeme na výstup.

Další problém při komunikaci s řídící jednotkou je, že nedává vědět (ani to sama neví), kdy je ukončen pohyb ramene. Tento problém je zásadní ve chvíli, kdy chceme posílat několik příkazů pro pohyb za sebou. V takovém případě se provede první pohyb, ale když během něj zašleme příkaz k druhému pohybu, skončí to chybovou hláškou: „Robot je v pohybu.“ Toto lze řešit dvěma způsoby. Prvním je kontrola odpovědi, pokud nastala chyba, příkaz opakovat později. Druhé možné řešení je použití metody *waitUntilPositioned*. Tato metoda čeká, dokud nejsou souřadnice předané této metodě jako parametry shodné s aktuální pozicí.

5.3 Testování

Pro účely testování jsem vytvořil program, který umožňuje řízení ramene pomocí příkazů, zadávaných do příkazové řádky. Tento program se nachází v příloze i s dokumentací (program se jmenuje cmd.py a dokumentace cmd.html). Tento program je velice jednoduchý, ale prokázal funkčnost knihovny melfa.py. Aplikace se snaží o rozpoznání zadaného příkazu, pokud odpovídá známému příkazu, volá se pak příslušná metoda z knihovny. Pokud zadaný příkaz není rozpoznán je považován za normální příkaz jazyka Melfa-Basic IV a je poslán řídící jednotce ke zpracování. Je zde i vyřešen rychlé zastavení robota v případě nebezpečí. Pokud uživatel potvrdí volbu, aniž by něco zadal, rameno se okamžitě zastaví.

Příkazy, které program cmd.py zná a jejich popis jsou napsány v tabulce 4. V programu jsou z bezpečnostních důvodů zakomentovány pohyby, které hýbou robotem na určenou pozici.

Další program jenž byl demonstrován již před rokem. Bylo kreslení s pomocí ramena. K chapadlu byla připevněna fixa. Program načetl černobílý obrázek, který následně podle daného algoritmu nakreslil. Program využíval starší knihovnu. Jen pro zajímavost jsem ho umístil do přílohy pod názvem *jehlickovka.py*. Výsledkem byl nakreslený obrázek hlavy Mickeyho. Tento pokus, který se zdá být spíše hrátkou prokázal, že knihovna může sloužit k účelům, ke kterým vznikla. Tím

myslím, k sestavení programu pro ovládání ramena pomocí určitého algoritmu. Toto bude potřeba při další práci na „velkém projektu.“

Příkaz	**Počet argumentů**	**Popis**
halt, exit	0	Ukončí program
alarm	0	Restartuje alarm
joint	2	Otočí kloubem specifikovaným prvním parametrem o úhel daný druhým parametrem
mvsrel	3	Posune se o počet mm daný argumenty – x, y a z
speed	1	Nastaví rychlost na hodnotu danou parametrem
pposf, jposf	0	Zjistí nynější pozici (pposf) nebo natočení kloubů (jposf)
send	x	Pošle sekvenci příkazů, ty musí být odděleny #

Tabulka 4: Příkazy programu cmd.py

6 Závěr

Během práce na tomto projektu jsem splnil všechny body zadání bakalářské práce. Naučil sem se ovládat robotické rameno Melfa 6 SL od firmy Mitsubishi. Seznámil jsem se s jeho možnostmi, které jsou dostačující pro práci, ke které byl školou zakoupen. Knihovna, kterou jsem navrhl podle dosavadních zkušeností s prací s ramenem je dostačující pro pokračování v projektu online laboratoře.

Další pokračování projektu vidím v lepším zabezpečení proti poškození vybavení laboratoře. Momentálně se mi zdá jako nejlepší varianta omezit pracovní prostor, ve kterém bude rameno pracovat. Další možností a cílem projektu by mohlo být vytvoření simulačního nástroje, na kterém by se daly verifikovat akce před jejich provedením.

Literatura

[1] Mitsubishi Industrial Robot: Instruction manual, Melfa BFP-A5992-K, 2005

[2] Skládanka Josef: *Robotické rameno – navigace pomocí kamery Axis 214*. Brno, 2009, bakalářská práce, FIT VUT v Brně.

[3] Mitsubishi Industrial Robot: Instruction manual, Melfa BFP-A8323-K, 2005

[4] Peňázková, M.: Když se řekne... Robot [online]. [cit. 2009-05-18] URL <http://www.automatizace.cz/article.php?a=2194>

[5] Wikipedia: Python [online]. [cit. 2009-05-18] URL <http://cs.wikipedia.org/wiki/Python>

[6] Mitsubishi Industrial Robot: Instruction manual, Melfa BFP-A5993-K, 2005

Seznam příloh

Příloha 1. CD