Mendelova univerzita v Brně
Provozně ekonomická fakulta

Manipulace s objekty pomocí robotu Mitsubishi RV-2AJ založená na analýze obrazu

Diplomová práce

Vedoucí práce:

Ing. Vít Ondroušek, Ph.D.

Bc. Filip Koutský

Brno 2015

Čestné prohlášení

Prohlašuji, že jsem práci: Manipulace s objekty pomocí robotu Mitsubishi *RV-2AJ* založené na analýze obrazu vypracoval/a samostatně a veškeré použité prameny a informace uvádím v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne:

.....

podpis

Abstract

KOUTSKÝ, F. *Object manipulation based on image analysis using a Mitsubishi RV-2AJ robot*. Diploma thesis. Brno: Mendel University in Brno, 2015

This Thesis deals with proposal, implementation and testing of a program for detection of objects to be manipulated and a program for control of robotic arm *Melfa RV-2AJ* manipulating with these objects. The Theoretical part describes selected algorithms for object detection using a Basler camera and introduces options for controlling the robotic arm *Melfa RV-2AJ*. The Practical part focuses on development and implementation of a program in the *LabVIEW Development Environment* for object detection using *LabVIEW* module "*NI Vision Acquisition Software*" and "*NI Vision Builder for Automated Inspection*". Furthermore it describes the development of a program for control of robotic arm *Melfa RV-2AJ* with use of the *LabVIEW* module for Mitsubishi robot control called "Imaging Lab". The program has been tested by a set of testing tasks and the test results subsequently used for suggestions for functionality improvements.

Keywords

robotic arm, *LabVIEW*, detection objects, camera, manipulation

Abstrakt

KOUTSKÝ, F. *Manipulace s objekty pomocí robotu Mitsibishi RV-2AJ založená na analýze obrazu*. Diplomová práce. Brno: Mendelova Univerzita v Brně, 2015

Diplomová práce se zabývá návrhem, implementací a testováním programu pro detekci objektů určených k manipulaci a programu pro řízení robotického ramene *Melfa RV-2AJ*, které s objekty manipuluje. V teoretické části práce jsou popsány vybrané algoritmy pro detekci objektů, nasnímaných za pomoci kamery Basler a možnosti řízení robotického ramene *Melfa RV-2AJ*. Praktická část je zaměřena na návrh a implementaci programu ve vývojovém prostředí *LabVIEW* pro detekci objektů, za pomoci modulů „*NI Vision Acquisition Software*" a „*NI Vision Builder for Automated Inspection*" a řízení robotického ramene *Melfa RV-2AJ* za pomoci modulu pro řízení robotů mitshubishi „*ImagingLab*". Tyto programy jsou následně otestovány vytvořenými testovacími úlohami. Výsledky jsou v závěru zhodnoceny a v závislosti na nich jsou navrhnuta možná vylepšení.

Klíčová slova

robotické rameno, *LabVIEW*, detekce objektů, kamera, manipulace

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Průmyslová robotika je v dnešní době již běžnou součástí různých výrobních procesů, kde nahrazuje lidskou práci. Jejich zařazení do výrobních procesů bývá většinou z důvodu levnějších nákladů, přesnosti a spolehlivosti. Jsou ale i další důvody, jako je možnost pracovat v nebezpečných podmínkách, ve kterých by lidé ani pracovat nemohli, nebo prostředí náročné na práci, kde je například vysoká prašnost, hlučnost apod.

S postupem času a neustále se vyvíjející technikou jsou i nároky na nynější průmyslové roboty stále větší. Z toho důvodu, jsou na trhu k dostání v mnoha konfiguracích. Nejdůležitějšími parametry jsou například, počet stupňů volnosti, přesnost, opakovatelnost, rychlost, nosnost, typ pohonu, nabídka koncových efektorů atd. Roboty jsou ve velkém množství naprogramovány staticky, kde vykonávají předem určenou posloupnost pohybů. Takové roboty nemají další senzorickou soustavu, která by nějak kontrolovala jejich pohyby nebo je jakkoli upravovala. U takto naprogramovaných robotů pak robot při jakékoli anomálii, buď nedokončí činnost, kterou má vykonat, nebo v horším případě dojde ke kolizi. Další možností je použití různých senzorických soustav, které se starají o kontrolu pracovního prostoru. V případě výskytu anomálie jsou schopny tuto anomálii rozpoznat a adekvátně zareagovat.

Velmi často bývá jako senzor použita kamera. Obraz získaný z kamery se zpracovává a vyhodnocuje za pomoci různých algoritmů. Pomocí nich jsme schopni z obrazu získat informace například o umístění objektů, jestli je objekt kompletní (správnost osazení desky plošných spojů), nebo zdali je výrobek požadovaných rozměrů. To vše spadá pod odvětví výpočetní techniky nazvané Počítačové vidění. V případě použití počítačového vidění v průmyslu je často používán pojem Strojové vidění. (1, 2)

1.1 Motivace

Na ústavu informatiky na Mendelově univerzitě je skupina studentů a učitelů pracujících na vývoji autonomních robotů, programování manipulačních systémů a robotů, která se nazývá AiStorm. V robotické laboratoři se nachází osvětlená klec pro manipulátor a dopravníkový pás osazená kamerami. V kleci se prozatím nachází robotické rameno KatHD300s. Místo tohoto robotického ramene má být do klece umístěno průmyslové robotické rameno Mitsubishi *Melfa RV-2AJ*. To má za pomoci analýzy obrazu detekovat objekty pohybující se po dopravníkovém pásu a následně objekt přemístit na předem dané souřadnice. Robot Mitsubishi *RV-2AJ* bylo doposud možné ovládat pouze za pomoci pevného programování pomocí jazyka *Melfa Basic* nebo příkazů *Movemaster*. To znamená, že doposud nebylo možné robot ovládat v reálném čase, tedy i když byly zjištěny souřadnice objektu, nebylo možné je robotu předat. Za tímto účelem vznikla tato diplomová práce, která se zabývá i analýzou obrazu, jak je popsáno níže v cíli práce.

1.2 Cíl Práce

Cílem diplomové práce je návrh, implementace a otestování programu pro detekci objektů v obraze a programu pro řízení robotického ramene Mitsubishi *RV-2AJ*, které bude s objekty manipulovat. Pro řešení jsou k dispozici moduly pro detekci obrazu ve vývojovém prostředí *LabVIEW Vision Acquisition software* a *Vision Builder for Automated Inspection* a modul pro řízení robotického ramene Mitsubishi *RV-2AJ ImagingLab*.

2 Vybrané algoritmy analýzy obrazu

Počítačové vidění, analýza obrazu, strojové vidění a další podobné názvy označují jednu vědní disciplínu. Jedná se o zpracování jakýchkoliv obrazových dat s využitím počítačové techniky. Účelem je zjištění určité informace z obrazu nebo o něm. Výzkum v oblasti počítačového vidění probíhá paralelně. Na jedné straně jsou matematické techniky pro zpětné získání trojrozměrného modelu a na druhé zkoumání a hledání různých vlastností v 2D obraze jako tvar, barva, počet objektů apod. Trojrozměrný model je například možné získat za pomoci velkého množství fotografií jednoho a toho samého objektu z různých úhlů. U rozpoznávání 2D obrazu je možné zase například zjistit počet osob a přiřadit jim jména podle určitých vlastností jako barva vlasů, očí, oblečení atd.

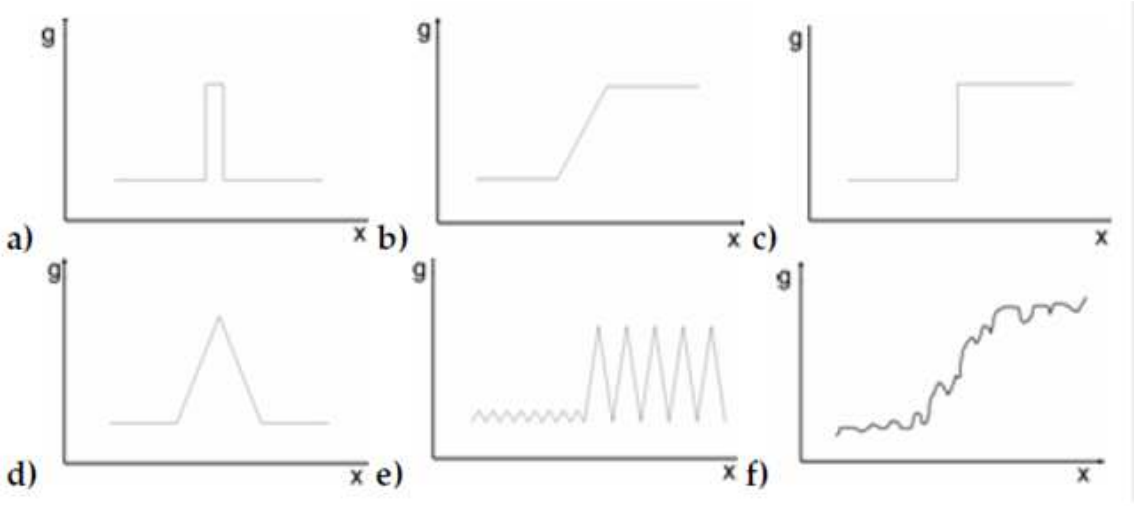
U počítačového vidění jsou kladeny vysoké požadavky na výkon hardwaru, který byl v minulosti schopen pracovat jen s malým množstvím dat. Naštěstí v dnešní době je už výkonného hardwaru dostatek a počítačové vidění se používá v mnoha odvětvích. Takové aplikace jsou například:

- **Optické rozpoznání textu (OCR – Optical Character Recognition):** převedení obrazu s textem na editovatelný text, rozpoznání ručně psaného písma…
- **Inspekce strojních součástí:** kontrola svárů v automobilovém průmyslu, kontrola tolerancí při výrobě součástek, kontrola úplnosti balení…
- **Zkoumání lidského těla:** rentgeny, detekce změny tkáně, získání dat o velikosti orgánů, toku krve…
- **Automobilová bezpečnost:** kontrola vyjetí z pruhu, detekce překážky před automobilem, automatické parkování…
- **Vytváření 3D modelů:** automatické modelování objektů z leteckých fotografií, laserové skenování objektů…

Tato práce se zabývá detekcí objektů, kde nebude zapotřebí konstruování jejich 3D modelů. Dále jsou tedy popisovány pouze algoritmy spojené s touto problematikou. Převážně se jedná o algoritmy pro vyhledávání předlohy v obraze, detekci hran atd. (3, 4)

2.1 Vyhledání hrany v obraze

Původně zamýšlenou technikou pro detekci objektů v obraze je detekce hrany v obraze. Hrana nemusí nutně být hranou nějakého objektu, ale může se jednat například o hranici mezi světlem a stínem. V obraze se hrana dá popsat jako skokový rozdíl jasové složky v obraze. Hrany se dají rozdělit dle jasového profilu do více typů, viz obrázek Obr. 1. Tyto profily jsou idealizované a reálně je jasový profil hrany v obraze zašuměný, viz profil *f* na obrázku Obr. 1.



Obr. 1 Jasové profily hrany v obraze: *a)liniová hrana; b)náběžná hrana; c)skoková hrana; d)střechová hrana; e)rozptylová hrana; f)zašuměná hrana*

Detekovat takovou hranu je možné za pomoci detekčních algoritmů. Ty se dají rozdělit na tři kategorie. První kategorie jsou detekční algoritmy založené na hledání maxim prvních derivací (*Robins, Prewitt, Kirsch*). Další kategorie algoritmů je založena na hledání průchodů druhých derivací nulou *(Marr-Hildreth*). Poslední kategorií jsou algoritmy založené na lokální aproximaci obrazové funkce parametrickým modelem (*Haralick*). (5)

2.1.1 Metody založené na hledání maxim prvních derivací

U těchto metod je základní myšlenkou to, že v místech, kde se nachází hrana, je i největší změna intenzity jasové složky. Zde také vycházejí maxima první derivace. V opačném případě, kde se intenzita nemění vůbec, první derivace vychází nulová.

Těmto metodám se také říká gradientní metody. Provede-li se parciální derivace obrazu podle x a y, získá se vektor udávající směr a velikost gradientu. Gradient ∇g(i, j) je vektor kolmý na vektor udávající směr hrany. Velikost hrany a úhel svírající s osou x, je možné spočítat za pomoci vzorců (1) a (2).

$$|\nabla g(i,j)| = \sqrt{\left(\frac{\partial \psi g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \tag{1}$$

$$\psi = acrtg\left(\frac{\frac{\partial g}{\partial y}}{\frac{\partial g}{\partial x}}\right) \tag{2}$$

Dalším krokem je určení parciálních derivací podle x a y a následuje aproximace vhodným výpočtem diferenciálu. Nejčastěji se používá Centrální diferenciální rovnice (3)

$$\Delta f(x) = \frac{(f(x+1)-f(x-1))}{2h} + O(h^2) \tag{3}$$

Více než určování parciálních derivací se používá výpočet jednotlivých složek gradientu za pomoci hranového operátoru. Provede-li se konvoluce obrazu za pomoci hranového operátoru, který je konvoluční maskou, obdržíme hledanou

složku gradientu. Ukázka jednotlivých hranových operátorů je na rovnicích 4 až 6.

Prewittův operátor $\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix}\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix}\begin{bmatrix}-1 & 0 & 1\\ -1 & 0 & 1\\ -1 & 0 & 1\end{bmatrix}$ (4)

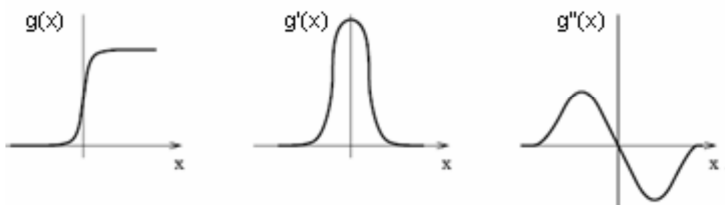
Robinsův operátor $\begin{bmatrix}-1 & 1 & 1\\ -1 & -2 & 1\\ -1 & 1 & 1\end{bmatrix}\begin{bmatrix}1 & 1 & 1\\ 1 & -2 & 1\\ -1 & -1 & -1\end{bmatrix}\begin{bmatrix}1 & 1 & 1\\ -1 & -2 & 1\\ -1 & -1 & 1\end{bmatrix}$ (5)

Kirschův operátor $\begin{bmatrix}-5 & 3 & 3\\ -5 & 0 & 3\\ -5 & 3 & 3\end{bmatrix}\begin{bmatrix}3 & 3 & 3\\ 3 & 0 & 3\\ -5 & -5 & -5\end{bmatrix}\begin{bmatrix}3 & 3 & 3\\ -5 & 0 & 3\\ -5 & -5 & 3\end{bmatrix}$ (6)

Nedostatkem při použití metody hledání maxim prvních derivací je vytváření příliš tlustých hran. Polohu hrany tedy není možné přesně určit. Tento nedostatek řeší Cannyho detektor hran, který se také dá zařadit do této kategorie. Canyho detektor hran nevyužívá pouze metodu hledání maxim prvních derivací, ale je rozšířen o Gaussův filtr na eliminaci šumu, nalezení lokálních maxim pro zpřesnění polohy hrany a o eliminaci nadbytečných hran. (5)

2.1.2 Metody založené na hledání průchodu druhé derivace nulou

Tato metoda se používá v případě, že není zapotřebí znát směr ani velikost příslušné hrany, ale pouze její polohu. Princip spočívá v hledání průchodu druhé derivace nulou, kde se nachází také maximum změny intenzity a maximum první derivace, jak je vidět na obrázku Obr. 2.



Obr. 2 Průběh obrazové funkce: a)původní funkce; b)první derivace; c)druhá derivace

Při výpočtu druhé derivace se setkáváme s problémem výpočtu diskrétní funkce. Ta nelze vypočítat pomocí dvojitého použití první derivace, viz rovnice (7), ale za pomoci rovnice (8).

$$\Delta g(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \tag{7}$$

$$\Delta^2 f(x) = \frac{(f(x+1)-f(x-1))}{h^2} + O(h^2) \tag{8}$$

Druhou možností je použití některého z hranových operátorů. Mezi nejčastěji používané se řadí Laplaceovy operátory, které se vyznačují kladením důrazu na střed, kde součet veškerých prvků je roven nule. Ukázku Laplaceova operátoru je možné vidět v rovnici (9)

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (9)$$

Laplaceovy operátory se z důvodu ještě větší citlivosti na šum než hranové operátory, používají v kombinaci s Gaussovým filtrem. Operátory jsou pak řazeny do kategorie LoG (Laplacian of Gaussian). Rovnice pro druhou derivaci Gaussova filtru je vyjádřena v rovnici (10).

$$G''(x,y) = c\left(\frac{x^2+y^2-\sigma^2}{\sigma^4}\right) e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (10)$$

Pro dobré výsledky je zapotřebí zvolit správně velikost jádra. Čím větší jádro, tím více je LoG odolnější proti šumu. Velikost jádra by dále měla odpovídat velikosti požadovaných detailů, které mají být zachyceny.

Další možností je použití filtru DoG (Difference of Gaussian), který pracuje na principu rozdílů Gaussových filtrů. (5, 6)

2.2 Vyhledání vzoru v obraze

Jedna z používaných technik v této diplomové práci je hledání předem daného vzoru v obraze, za účelem získání souřadnic hledaného objektu. Jedná se o jednu z velmi důležitých technik analýzy obrazu. Díky neustále se zvyšujícímu výkonu, je možné také tuto techniku používat při kontinuálním snímáním obrazu a vyhledávat objekty v obraze v reálném čase. Jako příklad lze uvést konkrétní uplatnění v této diplomové práci, kde se používá pro zjištění souřadnic značek s přesně zaměřenými souřadnicemi v pracovním prostoru robotu. Za pomoci těchto souřadnic je následně provedena transformace souřadného systému kamery do souřadného systému robotu.

Řešení této techniky, které je užito ve funkci *Find pattern* ve vývojovém prostředí *LabVIEW*, se dají rozdělit na dvě kategorie. Jednou z kategorií je vyhledávání vzoru na základě korelace a druhou kategorií je vyhledávání vzoru na základě porozumění obrazu. U korelace jde o hledání potencionální polohy předlohy, kdežto u porozumění obrazu je snaha o modelování objektů ve vzoru.(7)

2.2.1 Metoda normalizované korelace (NCC)

Klasické hledání vzoru zahrnuje normalizovanou korelaci, která je nejčastěji používanou metodou pro hledání vzoru v obraze. Základní koncepcí korelace je zjištění intensity v určitém bodě. Předpokládá se, že $f(x, y)$ udává intenzitu obrazu f o velkosti $M \times N$ v bodě (x, y), kde $x \in \{0, \ldots, M-1\}$ a $y \in \{0, \ldots, N-1\}$. Dále je zde vzor t o velikosti $K \times L$. Pro zjištění polohy vzoru t v obraze f je zapotřebí posoudit korelační koeficient γ v každém bodě (u, v) mezi vzorem t a obrazem f, kde vzor t je vždy posunut právě o u v ose x a v v ose y. Rovnice pro výpočet korelačního koeficientu je vyjádřena v rovnici (11).

$$\gamma = \frac{\sum_{x,y}\left(f(x,y)-\bar{f}_{u,v}\right)\left(t(x-u,y-v)-\bar{t}\right)}{\sqrt{\sum_{x,y}\left(f(x,y)-\bar{f}_{u,v}\right)^2 \sum_{x,y}\left(t(x-u,y-v)-\bar{t}\right)^2}} \quad (11)$$

V rovnici $\bar{f}_{u,v}$ určuje střední hodnotu *f(x, y)* v oblasti šablony *t* posunuté o (u, v), která se vypočte za pomoci rovnice (12).

$$\bar{f}_{u,v} = \frac{1}{N_x N_y}\sum_{x=u}^{u+N_x-1}\sum_{y=v}^{v+N_y-1} f(x,y) \quad (12)$$

Se stejnou notací $\bar{t}$ je myšlena hodnota předlohy *t*. Jmenovatelem v rovnici (1) je rozptyl nulového bodu průměru funkce obrazu $f(x,y) - \bar{f}_{u,v}$ a posunutého nulového bodu průměru funkce vzoru $t(x-u, y-v) - \bar{t}$. Díky této normalizaci je algoritmus nenáchylný na změny jasu a kontrastu obrazu.

Výsledné souřadnice (u, v) předlohy *t* jsou nalezeny tak, že se hledá nejvyšší korelační koeficient γ_{max}, odkud zjistíme i (u_{max}, v_{max}). Díky normalizaci je výpočet pozice předlohy v obraze přesnější než výpočet podobnými metodami, kterými jsou například jednoduchá kovariance a suma absolutních diferencí. Metoda normalizované korelace má i své nevýhody. První je relativně vysoký nárok na výkon. Další nevýhodou je nemožnost detekovat vzor, který je otočen nebo má jiné měřítko. Normalizovaná korelace je tak schopna detekovat většinou otočení pouze o 5 – 10°. (7,8, 9)

2.2.2 Metoda suma absolutních diferencí (SAD)

Metoda suma absolutních diferencí (SAD – Sum of Absolute Difference) je jednoduchá metoda pro měření podobnosti mezi obrazem a vzorem, který je výřezem z obrazu. Pracuje na principu absolutních diferencí každého pixelu ve zdrojovém obraze a vzoru. Z těchto diferencí je poté vytvořen souhrn, ze kterého je následně vytvořena jednoduchá metrika podobnosti.

Mějme výchozí obraz *S(x, y)* o rozměrech M×N a vzor *S(x, y)* o rozměrech K×L, kde je podmínkou, že K<M a L<N. Pro každý pixel (x, y) v obraze T se spočítá součet absolutních diferencí za pomoci rovnice (3).

$$SAD(x,y) = \sum_{i=0}^{(M-1)(N-1)}\sum_{j=0}^{(N-1)} |S(x+i, y+j) - T(i,j)| \quad (13)$$

Vzhledem k tomu, že se jedná o součet absolutních rozdílů, je důležité, aby pro dobré výsledky algoritmu byl vzor nejlépe vystřižen z výchozího obrazu. Stejně tak je problém při detekcích vzoru, který je v jiném měřítku. Metoda jinak dosahuje úspěšnosti přibližně 98%. (9, 10)

3 Získávání obrazu

V předchozí kapitole jsou popisovány různé algoritmy pro detekci objektů, hran, nebo vyhledávání předlohy v obraze. Aby bylo možné tyto algoritmy použít, je zapotřebí nějaký ten obraz získat. V případě této diplomové práce se pro snímání obrazu používají kamery. V dnešní době je na výběr nespočetné množství kamer různého druhu a zaměření, ale pro účely počítačového vidění, nebo v tomto případě spíše strojového vidění, je vhodné použít sofistikovanější kamery určené pro tyto aplikace.

Zde se setkáváme se dvěma typy kamer. Jedním je kamera poskytující pouze obraz, který se následně zpracovává na počítači. Druhým typem kamery je takzvaná Smart kamera, která má v sobě malý počítač na kterém již může běžet nějaký program zpracovávající obrazové data. U obou typů kamer je většinou na výběr velké množství příslušenství a různých konfigurací kamer.

Nejdůležitějším parametrem při výběru kamery je obrazový snímač. Nejde jen o typ, ale i o velikost či počet snímačů v kameře. Co se týče typů, u průmyslových kamer pro klasické snímání obrazu se používají především dva. Jedním je snímač CMOS, který má dobrou citlivost na světlo a dokáže poskytovat vysoké rozlišení při nízké spotřebě. Vyskytuje se u něj však jeden problém a tím je zkreslování (křivení) objektů při rychlejším pohybu. Naopak výhodou je jeho jednoduchá a levná výroba. Druhým je snímač typu CCD, který, pokud se nejedná o nějaký nejlevnější, dosahuje lepších výsledků než snímač CMOS. CCD snímače poskytují kvalitní obraz s malým šumem, kde mají oproti CMOS snímačům velký náskok. Dalším velmi důležitým faktorem je rozlišení výsledného obrazu. V dnešní době je běžné již rozlišení Full HD (1920 × 1080), ale z důvodu velkých objemů dat se používá spíše méně. Pro potřeby průmyslových kamer je ale důležitá také rychlost snímání obrazu. Udává se ve snímcích za sekundu (fps – frames per second).(4)

3.1 Kamera Basler acA1600-uc

Kamera Basler acA1600-20uc je kamera osazená barevným obrazovým CCD čipem od výrobce Sony typ ICX274 o velkosti 7.16 × 5.44 mm. Ten je schopen zvládnout 20 snímků za sekundu o velikosti 1628 × 1236 pixelů. O propojení kamery s počítačem se stará rozhraní USB 3.0. Spoušť kamery je možné ovládat za pomoci kamerového API (Application Programming Interface). Kompletní specifikace je v tabulce Tab. 1. (11)

Obr. 3 Kamera Basler acA1600-uc (11)

Parametr	**Hodnota**
Rozlišení	1624 × 1234 pixel
Velikost pixelu	4.4 × 4.4 µm
Rychlost snímání	20 fps
Připojitelné rozhraní	USB 3.0
Výstupní formát	Mono 8, YUV 4:2:2 Packed, YUV 4:2:2 (YUYV) Packed, Bayer BG 8, Bayer BG 12, Bayer BG 12 Packed, RGB8 and BGR8
Bitová hloubka pixelu	12 bit
Synchronizace	Externí spoušť, volný běh
Řízení spouště	Skrz kamerové API, pomocí externího signálu
Uchycení (d × š × v) v mm	29,3 × 29 × 29
Pracovní teplota	0 – 50°C
Uchycení objektivu	C-Mount, CS-Mount
Digitální vstup	1
Digitální výstup	1
Napájení	Skrz rozhraní USB 3.0
Spotřeba	2,9 W
Váha	80 g
Výrobce senzoru	Sony
Typ senzoru	ICX274
Technologie senzoru	CCD
Velikost senzoru	7,16 ×5,44 mm

Tab. 1 Kompletní popis kamery Basler acA1600-uc (11)

Ke kameře je na výběr velké množství objektivů od tří výrobců Ricoh, Computar nebo Edmund Optics. Jedná se o objektivy s pevnou ohniskovou vzdáleností. Tato kamera je vybavena objektivem s ohniskovou vzdáleností f25mm a světelností F1,4. Parametry byly voleny pro použití kamery s tímto objektivem, v kleci s manipulátorem a pásovým dopravníkem. Ostření objektivu se provádí manuálně, stejně tak se provádí nastavení clony. Objektivy se montují na úchyty *C-Mount, CS-Mount*. (11)

Obr. 4 Objektiv Computar M2514-MP2 (11)

3.2 Karta pro USB 3.0 kamery PCIe-8242

Jedná se o kartu pro zachytávání obrazu z kamer vybavených komunikačním rozhraním USB 3.0 od National Instruments. Karta se používá pro propojení a inicializaci kamery z vývojového prostředí *LabVIEW*. Karta podporuje většinu kamer dostupných na trhu. Díky rozhraní USB3.0 s rychlostí dosahující 400MB/s je vhodná pro rychlý přenos dat do počítače při strojovém vidění. Karta je určena do slotu PCI expres a je vybavena dvěma rozhraními USB3.0. Ke kartě je rovněž přibalena knihovna *Vision Acquisition software* pro snímání obrazu z kamer, jeho analýzy apod. Více je software popsán v kapitole 5.1. (12)

Obr. 5 Rozšiřující karta pro kamery s rozhraním USB3.0 (12)

4 Průmyslový manipulátor Mitsubishi *RV-2AJ*

Dle normy ISO 8373 jsou průmyslové manipulátory definovány jako automaticky řízené, programovatelné, víceúčelové manipulátory programovatelné ve třech a více osách. Tyto manipulátory jsou většinou používány pro lakování, montáže, svařování, přemisťování objektů apod. Jedná se o velice přesné a rychlé stroje s vysokou spolehlivostí. Právě díky těmto třem vlastnostem se stává, že tyto manipulátory nahrazují lidskou práci, která v těchto vlastnostech nedosahuje kvalit manipulátorů.

Stejně jako u kamer, je zde mnoho parametrů, podle kterých lze manipulátor přizpůsobit požadovaným úkonům. Jedná se o:

- **počet os** – dvě osy jsou zapotřebí pro dosažení bodů na ploše, tři osy zajistí dosažení bodů v prostoru. Pro větší variabilitu (vybočení, přetočení atd.) je lepší větší počet os.
- **počet stupňů volnosti** – určuje základní směry posunu a otáčení, kterými se manipulátor může pohybovat
- **kinematika** – uspořádání tuhých členů robotu, určující jeho možné pohyby. Dělí se na kloubové, kartézské, paralelní a SCARA
- **nosnost** – maximální zátěž na konci ramene
- **rychlost** – určuje rychlost změny pozice robotu. Může být udávána jako úhlová rychlost kloubů, rychlost lineárního posuvu v každé ose, nebo rychlost koncového efektoru
- **zrychlení** – maximální možné zrychlení osy
- **přesnost** – udává, s jakou přesností je možné se dostat na určené souřadnice. Zde je možné přesnost zvýšit za pomoci externích senzorů kamer apod.
- **opakovatelnost** – určuje jak přesně je robot schopen se dostat opakovaně na jedny souřadnice (13)

4.1 Popis robotu RV-2AJ

Mitsubishi *RV-2AJ*, viz obrázek Obr. 6, je průmyslové robotické rameno s pěti stupni volnosti. Robot má štíhlou konstrukci, která je vhodná pro použití ve stísněných prostorách jako například integrace do výrobní linky, nebo instalace do uzavřené kabiny. Takovému použití je přizpůsobená i nosnost ramene, která je u modelu *RV-2AJ* 2 kilogramy. Je zde i možnost rameno připevnit na strop nebo na stěnu. Maximální dosah ramene při koncovém efektoru orientovaným směrem dolů je 410 mm. Přesnost s jakou je možné se dostat na udané souřadnice je ±0,02 mm. Maximální rychlost ramene je 2 100 mm/s. Kompletní technická specifikace je zobrazena v tabulce Tab. 2.(14)

		Jednotky	Specifikace
Typ			RV-2AJ
Stupeň volnosti			5
Poloha instalace			Na podlahu, zavěšený
Struktura			vertikální, víceosý typ
Pohonný systém			AC servomotory (J1 - J3:50W s brzdou, J5:15W s brzdou, J6:15W bez brzdy)
Systém detekce polohy			Číslicové snímače absolutní polohy
Délka ramene	Předsunuté rameno	mm	0
	První rameno		250
	Druhé rameno		160
	Třetí rameno		72
Operační rozsah	J1	stupně	300(-150 to +150)
	J2		180(-60 to +120)
	J3		230(-110 to +120)
	J5		180(-90 to +90)
	J6		400(-200 to +200)
Rychlost pohybu	J1	stupně/s	180
	J2		90
	J3		135
	J5		180
	J6		210
Maximální výsledná rychlost		mm/s	přibližně 2100
Zatížení	Maximální	kg	2
	Doporučené		1,5
Opakovatelnost		mm	± 0,02
Pracovní teplota		°C	0 to 40
Váha		kg	přibližně 17
Přípustné momentové zatížení	J5	N · m	2,16
	J6		1,10
Přípustná setrvačnost	J5	kg · m²	$3{,}24\times10^{-2}$
	J6		$8{,}43\times10^{-3}$
Maximální dosah při sklopeném koncovém efektoru		mm	410
Možnosti připojení koncového efektoru			Čtyři vstupní signály (u koncového efektoru), čtyři výstupní signály (u základny), výstup pro elektrickou ruku (u koncového efektoru)
Pneumatické potrubí pro koncový efektor			Φ4x4 (od základny ke koncovému efektoru)
Napájecí tlak		MPa	0,5 ± 10%
Izolace			IP30

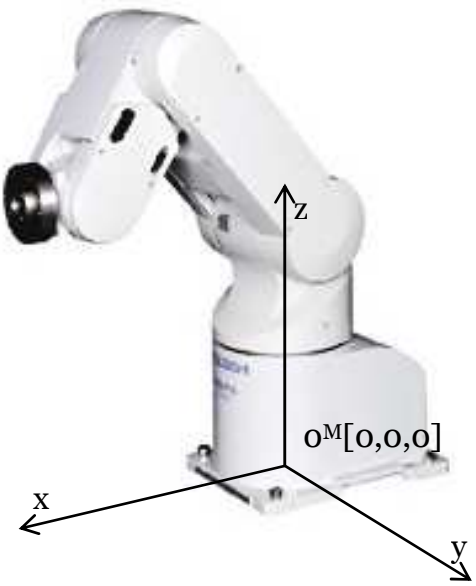
Tab. 2 Technické parametry robotu Melfa RV-2AJ (15)

Robotické rameno je poháněno střídavými servomotory s integrovanými číslicovými snímači absolutní polohy, které drží polohu i po vypnutí přívodu napájení. Odpadá tak kalibrování najížděním robotu do referenční polohy, při které by mohlo dojít ke kolizi.

Koncový efektor je řešen elektrickým chapadlem. Pro širší záběr použití je do robotického ramene zabudován i pneumatický rozvod pro koncový efektor s pneumatickým úchopem. Potrubní rozvod je veden uvnitř ramene, čímž je snižováno riziko možného poškození při kolizi. Vstupy jsou umístěny v základně společně s přívodem napájení a ovládacími kabely. Výstup, jak pro elektrický, tak pro pneumatický koncový efektor, je na předposledním rameni co nejblíže ke koncovému efektoru.

Pro řízení ramene je použita řídicí jednotka, která se běžně používá pro větší roboty. Ta disponuje velkou škálou volitelného příslušenství a rozšíření

jako například modul pro připojení robotu k ethernetové síti s protokolem TCP/IP, nebo modul, pomocí kterého lze rozšířit rameno o další stupeň volnosti. (14)



Obr. 6 Robotické rameno Mitsubishi Melfa RV-2AJ a jeho souřadný systém (16)

4.1.1 Řídicí jednotka CR1-571

Řídicí jednotka CR1-571 dodávaná k robotu Mitsubishi *Melfa RV-2AJ*, viz obrázek Obr. 7, je vybavena velmi rychlým 64 bitovým procesorem, který je schopen zpracovávat až 32 paralelních úloh. To znamená, že když jsou obsluhovány vstupy a výstupy, přijímají se data komunikačním kanálem a řeší matematické výpočty, tak mimo to je schopna zvládat dalších 28 úloh. (14)

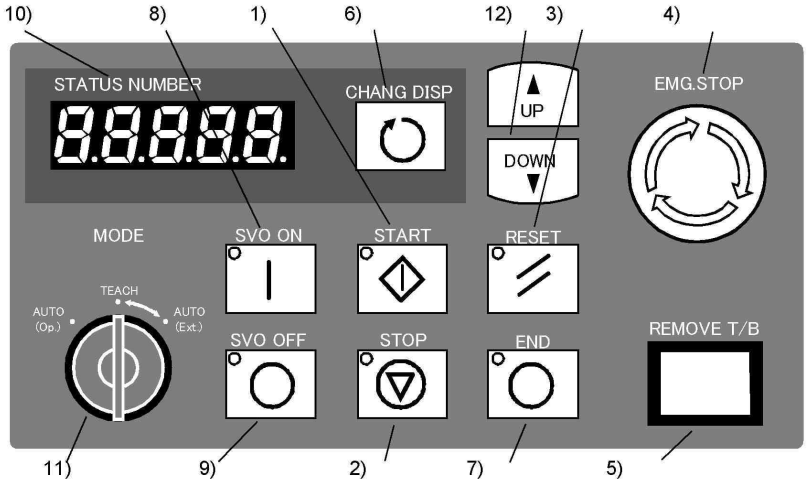
Obr. 7 Řídicí jednotka CR1-571 pro robot Mitsubishi RV-2AJ (17)

Komunikace je možná za pomoci sériového portu RS-232 a nebo šestnácti digitálních vstupů/výstupů. Tato komunikační rozhraní je možné dále rozšířit za pomoci rozšiřujících modulů do sběrnice až na 240 vstupů a 240 výstupů. Tyto moduly nejsou určeny pouze k rozšiřování počtu vstupů a výstupů, ale i k rozšiřování funkcí jednotky. Jedním takovým modulem je například modul ethernetového připojení za pomoci protokolu TCP/IP, který umožňuje začlenit robot do ethernetové sítě. Takto může velmi rychle komunikovat s řídicím programem běžícím na PC nebo na programovatelném automatu. Dalším zajímavým modulem je velmi výkonná síť CC-Link vyvinutá přímo firmou Mitsubishi Eletric. Hodí se převážně na opravdu velmi rychlou výměnu dat mezi robotem a programovatelným automatem. Je to způsobeno použitím krouceného páru vodičů pro přenos veškerých signálů, místo propojování jednotlivých vstupů a výstupů mezi roboty. Kompletní technická specifikace je vidět v tabulce Tab. 3. (14)

		Jednotka	Specifikace	Poznámky
Typ			CR1-571	
Maximální počet ovládaných os			6	
Procesor			64 bit RISC (Reduced Instruction Set Computing), and DSP (Digital Signal Processor)	
Paměťová kapacita	Počet naprogramovaných pozic a kroků programu	Pozic Kroků	2,500 5,000	
	Počet programů		88	
Programovací jazyk			MELFA-BASIC IV nebo *MOVEMASTER* COMMAND	
Metoda učení			Metoda učení pozic, metoda MDI	
Externí vstupy a výstupy	Vstupy a výstupy		16/16	Max. 240/240 (při rozšíření)
	Jednoúčelové vstupy a výstupy		„STOP“ tlačítko	
	Otevření/zavření ruky, vstupy a výstupy		4 vstupy, 0 výstupů	Navýšení výstupů je možné za pomoci rozšíření
	Nouzový vypínač			Samostatná linka pro nouzový vypínač
	Vstup pro dveřní kontakt			Samostatná linka pro dveřní kontakt
Rozhraní	RS-232C	port		Pro rozšíření jako například ovládání pomocí PC
	RS-422	port		Vyhrazen pro ovladač „Teach Box Controller“
	Slot vyhrazený pro uchopovací ruku	slot		Vyhrazen pro pneumatický uchopovací mechanismus
	Rozšiřující slot	slot	0	3 slotový rozšiřující modul jako možnost
	Vstupně-výstupní linka pro robot	kanál	1	Použit pro běžní vstupy/výstupy (Max. 240/240)
Zdroj napájení	Rozsah vstupního napětí	V	1 fázový, AC 90 - 132 1 fázový, AC 180 - 253	
	Příkon	KVA	0.7	Neobsahuje špičkový proud
Vnější rozměry		mm	212(Š)x290(H)x151(V)	
Váha		kg	Přibližně 8	
Konstrukce			Samonosný uzavřená, otevřená na montáž do boxu	IP20
Rozsah provozní teploty		°C	0 to 40	
Relativní vlhkost		%	45 to 85	
Zemnění		Ω	100 nebo méně	Zemnící třída D
Barva			Světle šedá	

Tab. 3 Technické parametry řídicí jednotky CR1-571 (15)

Jednotka má tři základní režimy řízení. Prvním je plně automatický režim, kdy je v jednotce nahrán příslušný program a jednotka obstarává vše potřebné. Dalším režimem je taktéž plně automatický režim, který je ale ovládán externě přes programovatelný automat či počítač. Program se tedy neprovádí v jednotce, ale právě v programovatelném automatu, nebo počítači a řídicí jednotce posílá pouze instrukce, které má vykonat. Posledním režimem je řízení za pomoci ovládacího panelu *Teach box R28TB*, což je manuální ovládání za pomoci ovladače. Popis ovládacích prvků řídicí jednotky Mitsubishi CR1-571 najdete na obrázku Obr. 8. (15)



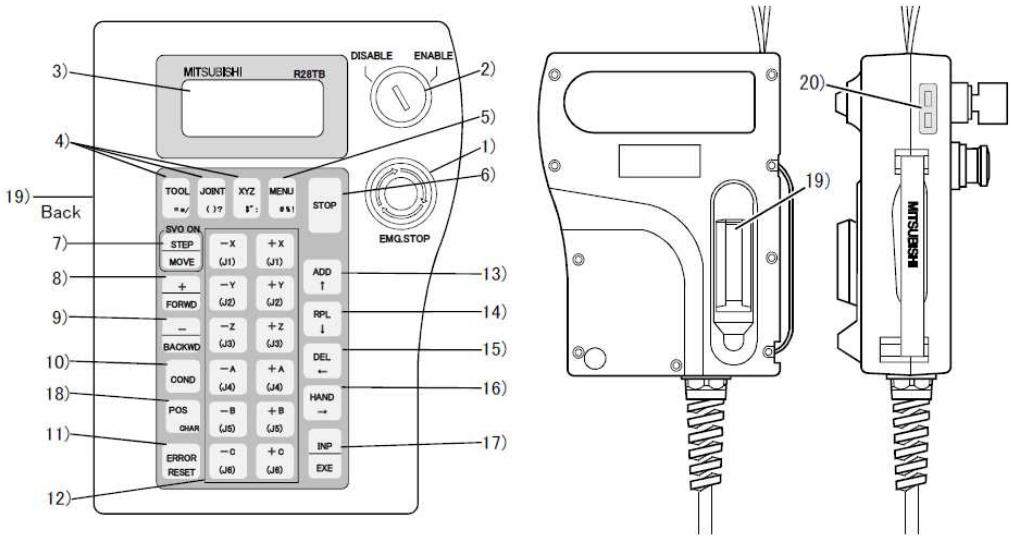
Obr. 8 Přední panel řídicí jednotky Mitsubishi Cr1-571 (15)

1. Tlačítko pro spuštění programu. Program probíhá kontinuálně. Při běhu programu svítí zelená kontrolka, jinak nesvítí vůbec.
2. Tlačítko pro zastavení programu. Po stisku se robot okamžitě zastaví, ale servomotory zůstávají zapnuty. Při zastaveném programu svítí červená kontrolka, jinak nesvítí.
3. Tlačítko na resetování chyb, vzniklých při provozu. Při běhu programu resetuje program. Při výskytu chyby kontrolka svítí, jinak je zhasnutá
4. Nouzové zastavení robotu. Zastavení servomotorů.
5. Tlačítko sloužící pro připojení/odpojení ovladače *Teach box R28TB*, bez nutnosti vypnutí řídicí jednotky. Odpojení/připojení je nutné provést do 5 vteřin, jinak se spustí alarm.
6. Tlačítko přepíná zobrazení určitých dat na displeji v pořadí číslo programu, číslo řádku, přepis. V případě chyby vypisuje na displej kód chyby
7. Tlačítko pro zastavení běžícího programu na posledním řádku nebo na příkazu end. Pokud je program zastaven, svítí červená kontrolka
8. Tlačítko pro zapnutí servomotorů. Při zapnutých servomotorech svítí zelená kontrolka, jinak nesvítí.
9. Tlačítko pro vypnutí servomotorů. Při vypnutých servomotorech svítí červená kontrolka, jinak nesvítí.
10. Stavový kód
11. Přepínač pro změnu způsobu ovládání:
    - AUTO (Op.): pouze pro ovládání řídicí jednotkou.
    - TEACH: pouze pro ovládání za pomoci ovladače *Teach box R28TB*.
    - AUTO (Ext): pouze pro ovládání z externího zařízení, jako počítač apod.
12. Tlačítka pro listování v detailech o programu, řádcích a chybách zobrazených na displeji (15)

4.1.2 Ovládací panel *Teach box R28TB*

Jak je zmíněno výše, jednou z variant jak robotické rameno *Mitsubushi Melfa RV-2AJ* řídit, je řízení pomocí ovladače *Teach box R28TB*. Za jeho pomoci lze

provádět veškerá nastavení jako například kalibrace, změny souřadného systému, nastavení posunu souřadného systému koncového efektoru pro daný typ a podobně. (18)



Obr. 9 Ovládací panel *Teach Box Controller R28TB* (18)

1. Tlačítko pro nouzové zastavení robotu.
2. Přepínač pro aktivování nebo deaktivování ovladače *Teach box R28TB*
3. LCD displej
4. [TOOL] Klávesa pro nastavení pohybu v souřadném systému koncového efektoru
5. [JOINT] Klávesa pro nastavení pohybu po jednotlivých osách robotu
6. [XYZ] Klávesa pro nastavení pohybu v souřadném systému robotu.
7. Klávesa pro vyvolání menu
8. Klávesa pro zapnutí servomotorů. Pro pohyb je nutné jej držet. Možné nastavit i krokový pohyb
9. Při současném držení s klávesou 17) je možné nastavit krokování. Při společném držení s klávesou 7) je možné zvýšit rychlost robotu. Jinak se používá pro pohyb mezi řádky programu.
10. Klávesa pro zobrazení programových instrukcí
11. Při současném držení s klávesou 17) je resetován právě běžící program. Jinak se používá pro resetování vyskytnutého chybového hlášení.
12. Klávesy pro pohyb s robotem při současném držení klávesy 7). Pohyb záleží na předem nastaveném způsobu pohybu.
13. Pohyb kurzorem nahoru. Při společném držení klávesy 7) lze přidávat nebo korigovat poziční data v editačním okně
14. Pohyb kurzorem dolů. Při společném držení s klávesou 7) se přejde na další obrazovku v editačním okně pozic
15. Klávesa pro mazání pozičních dat. Pohyb kurzorem doleva

16. Při společném s držením s klávesou [+C (J6)] nebo [-C (J6)] je možné ovládat chapadlo 1
    Při společném s držením s klávesou [+B (J5)] nebo [-B (J5)] je možné ovládat chapadlo 2
    Při společném s držením s klávesou [+A (J4)] nebo [-A (J4)] je možné ovládat chapadlo 3
    Při společném s držením s klávesou [+Z (J3)] nebo [-Z (J3)] je možné ovládat chapadlo 4
    Pohyb kurzorem doprava
17. Klávesa, která vloží program a provádí krokování/návrat
18. Tato klávesa slouží k zobrazení okna pro editaci displeje. Při držení mění význam kláves na znaky.
19. Spínač je aktivní pouze pokud je přepínač v poloze ENABLE. Spínač má dvě polohy. Při lehkém zmáčknutí je možné pracovat s ovladačem *Teach box R28TB*. Pokud se zmáčkne více až do druhé polohy, deaktivují se servomotory.
20. Kolébkové tlačítko na nastavení jasu displeje

Pro zprovoznění ovladače je zapotřebí přepnout řídicí jednotku do režimu *Teach* a zároveň aktivovat ovladač *Teach box R28TB* do režimu *Enable*. Ovladač je vybaven velkým spínačem tzv. *Deadmen Switch*, který se musí po celou dobu ovládání držen, jinak ovladač nereaguje na ostatní tlačítka. Popis celého ovladače *Teach box R28TB* naleznete na obrázku Obr. 9.

Za pomoci ovladače *Teach box R28TB* lze s robotem pohybovat několika způsoby. Nejzákladnější možností pohybu je rotace jednotlivými vazbami robotu. Další možností je pohyb v souřadném systému O^M viz obrázek Obr. 6, kde se robot pohybuje v jednotlivých osách. Na stejném obrázku lze vidět i umístění počátku souřadného systému, který je situován v ose první vazby a základny robotu. Souřadný systém je v případě potřeby, například při umístění robotu na stěnu, strop, nebo v případě už zavedeného jiného souřadného systému, možné předefinovat příslušným parametrem *MEXBS*. Ten se dá měnit jak za pomoci ovladače, tak v nějakém vytvořeném programu. (18)

4.1.3 *Melfa Basic* a příkazy *Movemaster*

Melfa Basic a příkazy *Movemaster* jsou programovací jazyky určené pro ovládání robotických manipulátorů od firmy Mitsibishi Electric. Tyto programovací jazyky jsou implementovány do různých programů pro tvorbu řídicích programů, jako jsou *RT Tool Box 2, MELFA WORKS, COSIMIR a COSIROP*.

Programovací jazyk *Melfa Basic* je tvořen jednoduchými příkazy, většinou s parametry. Příkazy se píší na samostatné řádky, po kterých se pak program postupně provádí. Ukázku řídicího programu je možné vidět pod odstavcem. *Melfa Basic* slouží pro kompletní ovládání a nastavení řídicí jednotky robotu. (19)

```
10  MVS P_SAFE          Přesun na pozici předem definovanou pozici
                        P_SAFE
20  IF M_IN(8) = 0      Čekání na nastavení 8 bitu, v případě 0 se
    THEN 20 ELSE 30     opakuje řádek 20, jinak se pokračuje na
                        řádku 30
30  HOPEN 1             Otevření chapadla 1
40  MVS PI,-50          Přesun 50 mm nad předem definovanou pozici
                        P1
50  MVS PI              Přesun na pozici P1
60  HCLOSE1             Zavřít chapadlo 1
70  DLY 0.2             Pauza 0,2 s z důvodu řádného uzavření
                        chapadla
80  MVS PI,-50          Přesun 50 mm nad pozici P1
90  MVSP2.-50           Přesun 50 mm nad předem definovanou pozici
                        P2
100 MVSP2               Přesun na pozici P2
110 HOPEN 1             Otevření chapadla 1 a odložení objektu
120 DLY 0.2             Pauza 0,2 s z důvodu řádného otevření
                        chapadla
130 MVSP2.-50           Přesun 50 mm nad pozici P2
140 IF M_IN(8) = 1      V případě, že je k dispozici další objekt
    THEN 40 ELSE 150    (8. bit) na přesun, skoč na řádek 40
150 MVS P_SAFE          V případě, že není další objekt na přesun,
                        přesun na pozici P_SAFE
160 END                 Konec programu
```

Příkazy *Movemaster* jsou obdobné jako příkazy *Melfa Basic*. Jsou vytvořeny pro řady robotů Mitsubishi *Movemaster* a E/EN. Příkazy programovacího jazyku *Movemaster* nejsou pro všechny typy a řady robotů Mitsubishi a je nutné zkontrolovat jejich kompatibilitu. Kompletní tabulky s příkazy a jejich vysvětlením pro programovací jazyky *Melfa Basic* a *Movemaster* je možné vidět v tabulkách v příloze A a B. (19)

5 Vývojové prostředí *LabVIEW*

LabVIEW (*Laboratory Virtual Instrument Engeneering Workbench*) je vývojové prostředí pro vizuální programování od vývojářů z *National Instruments*. *LabVIEW* používá místo použití klasického textového programovacího jazyka, grafický programovací jazyk G. Program se sestavuje z funkčních bloků do blokového diagramu. Funkční bloky mají vstupy a výstupy, které se propojují s ostatními funkčními bloky. Každý takový program se skládá ze dvou částí. Jednou je takzvaný *Front panel*, kde je možné zobrazit například nějaké proměnné, ovládací prvky, nebo indikátory stavu. Druhou částí je blokové schéma, kde je obsažena veškerá logika daného programu. Z vytvořených programů je možné vytvořit i podprogramy, které je pak možno použít v jiném programu.

Vývojové prostředí *LabVIEW* obsahuje mnoho modulů pro různé aplikace. *LabVIEW* je možné nadále rozšířit o další přídavné moduly, které přidají další možnosti vývojového prostředí. Takovými moduly jsou například, modul pro snímání obrazu z kamer *Vision Acquisition software,* modul pro vytváření pokusných programů pro snímání obrazu a jeho analýzy *Vision Builder for Automated Inspection,* nebo modul pro řízení robotického ramene Mitsubishi Melfa RV-2AJ *ImagingLab for Mitsubishi robots*. Tyto tři knihovny jsou popsány detailněji v dalších třech kapitolách, protože jsou používány pro tuto diplomovou práci.

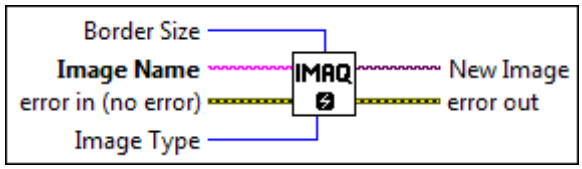
Další způsob, jak navýšit možnosti vývojového prostředí, jsou rozšiřující karty do počítačů. Jedná se většinou o karty pro připojení různých měřících zařízení, senzorů apod., se kterými vývojové prostředí *LabVIEW* lépe komunikuje. Samozřejmě je možností použít i standardní senzory a měřicí přístroje a komunikaci si vytvořit vlastní. (20)

5.1 Vision Acquisition software

Vision Acquisition software je modul pro jednoduché snímání, ukládání a zobrazování obrazu. Jedná se o nástavbu nad dvěma knihovnami, která je určená pro jednodušší nastavení snímání a ostatní práce s obrazem. V této kapitole jsou spíše popsány právě funkce z knihoven, kterou softwarový modul používá. Jak již bylo řečeno dříve, softwarový modul používá dvě knihovny. Jednou je knihovna pro snímání obrazu z analogových, digitálních, linkových a chytrých kamer, která se nazývá *NI-IMAQ*. Ta je k dispozici ve vývojovém prostředí v základu. Druhou je knihovna pro získávání obrazu z kamer vybavených komunikačním rozhraním USB 3.0, GigE, IP a IEEE 1394. Ta se nazývá *NI-IMAQdx* a je nutné mít zakoupenou licenci, nebo hardwarový modul, který má klíč ke knihovně v sobě. K plné funkčnosti obou knihoven, tedy i modulu *Vision Acquisition software*, je nutné mít nainstalován modul *Vision Development*. V následujících kapitolách jsou postupně popsány používané funkce této knihovny. Každá funkce obsahuje vstup a výstup pro chyby. Tyto chybové vstupy/výstupy nejsou popisovány, protože se stále opakují. (21)

Funkce IMAQ create

Jedná se o funkci pro vytvoření obrazového okna v paměti. Je zapotřebí zadat jméno pro obraz, což je jediný povinný parametr. Ostatní parametry, jako je šířka rámečku, typ obrazu, povinné nejsou. Výstupem je prázdný obraz alokovaný v paměti. Obrazů lze vytvářet libovolné množství. Je nutné, aby bylo jméno obrazu pokaždé unikátní.



Obr. 10 Schéma funkce IMAQ create

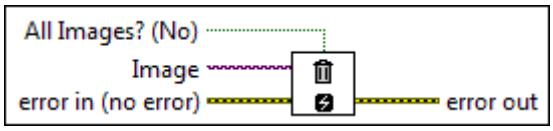
- *Border Size* – jedná se o nastavení velikosti rámečku okolo obrazu
- *Image Name* – jméno obrazu. Pokud se používá více obrazů, je nutné, aby byl unikátní. (povinný parametr)
- *Image Type* – zde je na výběr z osmi typů obrazu viz tabulka Tab. 4
- *New Image* – výstup obrazu alokovaného v paměti

Stupně šedi (U8)	8 bitů na pixel (výchozí hodnota)
Stupně šedi (U8)	16 bitů na pixel
Stupně šedi (U8)	32 bitů na pixel
Komplexní (CSG)	2 × 32 bitů na pixel
RGB (U32)	32 bitů na pixel (červená, zelená modrá, alfa)
HSL (U32)	32 bitů na pixel (odstín, saturace, jas, alfa)
RGB (U64)	64 bitů na pixel (červená, zelená modrá, alfa)
Stupně šedi (U8)	16 bitů na pixel

Tab. 4 Popis typů obrazu

Funkce IMAQ dispose

Funkce pro zrušení alokace obrazu v paměti. Tato funkce nemá žádné povinné parametry. Vstupem je obraz a volba pro smazání všech, nebo jen konkrétního obrazu. Výstup funkce je pouze chybový výstup, který není nutný popisovat.

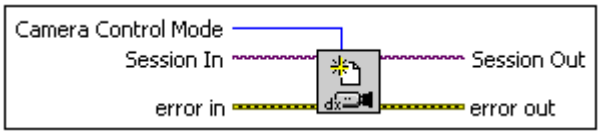


Obr. 11 Schéma funkce IMAQ dispose

- *All Images?* – volba zda smazat všechny, nebo jen konkrétní obraz (výchozí hodnota *NO*)
- *Image* – vstup obrazových dat, které se mají zrušit

Funkce IMAQdx Open Camera

Jedná se o funkci otevření relace kamery. Vstupem je výběr kamery a mód řízení kamery. Výstupem je pak relace kamery. (22)

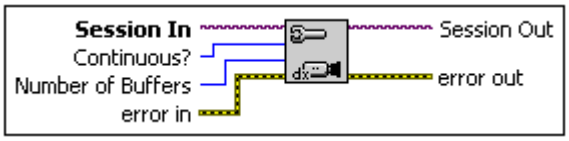


Obr. 12 Schéma funkce IMAQdx Open Camera

- *Session In* – je vstup pro výběr kamery u které se má vytvořit relace
- *Camera Control Mode* – nastavení režimu, ve kterém bude kamera pracovat. Na výběr je možnost *Controller* (možnost nastavovat kameru a přijímat data), což je i výchozí hodnota a *Listener* (pasivní snímání dat bez možnosti nastavování)
- *Session Out* – otevřená relace kamery

Funkce IMAQdx Configure Acquisition

Jedná se o funkci, která nastavuje snímání obrazu v otevřené relaci kamery. Povinným parametrem je vstup relace vytvořené za pomoci funkce *IMAQdx Open Camera*. Pak je možné nastavit, zda bude snímání kontinuální, nebo se pouze udělá jeden snímek. Další možností je nastavit počet snímků, které se nasnímají (v případě vypnutého kontinuálního snímání), nebo velikost vyrovnávací paměti (v případě kontinuálního snímání). Výstupem je zkonfigurovaná relace kamery definované dříve.

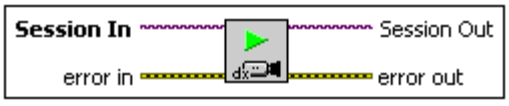


Obr. 13 Schéma funkce IMAQdx Configure Acquisition

- *Session In* – otevřená relace kamery (povinný parametr)
- *Continuous?* – nastavení typu snímání. Je zde na výběr typ *continuous,* který nastaví průběžné snímání obrazu, nebo *one shot*, který nasnímá určitý počet snímků dle dalšího parametru *Number of Buffers*
- *Number of Buffers* – při kontinuálním snímáním nastavuje počet přednačítaných snímků. V režimu jednotlivých snímků pak počet snímků, které se mají pořídit.
- *Session Out* – výstup zkonfigurované relace kamery

Funkce IMAQdx Start Acquisition

Tato funkce spouští relaci kamery s předchozím nastavením za pomoci funkce *IMAQdx Configure Acquisition*. Je zde pouze povinný vstup, kde se napojuje zkonfigurovaná relace kamery a výstupem je spuštěná relace kamery.

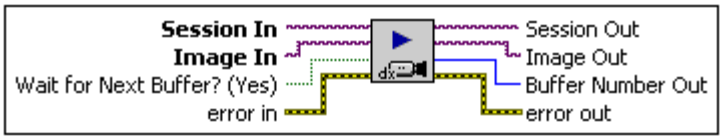


Obr. 14 Schéma funkce IMAQdx Start Acuisition

- *Session In* – Vstup zkonfigurované relace kamery (povinný parametr)
- *Session Out* – Spuštěná relace zkonfigurované relace kamery (22)

Funkce IMAQdx Grab

Jedná se o funkci zprostředkovávající již snímání obrazu z dané relace kamery. Má dva povinné vstupy, kde jeden je otevřená relace kamery, která může být i zkonfigurována (poté musí být spuštěna) a druhým je obraz alokovaný v paměti. Dalším parametrem je čekání na další snímek z vyrovnávací paměti. Výstupem je obraz z kamery, pokračování relace kamery a číslo snímku ve vyrovnávací paměti.

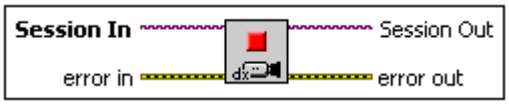


Obr. 15 Schéma funkce IMAQdx Grab

- *Session In* – vstup relace kamery (povinný parametr)
- *Image In* – vstup alokovaného obrazu v paměti (povinný parametr)
- *Wait for Next Buffer?* – nastavení zda má funkce čekat na další snímek ve vyrovnávací paměti. Možné volby jsou *Yes* (Ano, výchozí hodnota), nebo *No* (Ne)
- *Session Out* – výstup relace kamery pro další použití
- *Image Out* – výstup obrazu z kamery
- *Buffer Number Out* – číslo snímku ve vyrovnávací paměti

Funkce IMAQdx Stop Acquisition

Funkce starající se o ukončení snímání v relaci kamery. Vstupem je povinný parametr předchozí relace kamery, která byla spuštěna. Výstupem je relace kamery, která je zastavená.



Obr. 16 Schéma funkce IMAQdx Stop

- *Session In* – vstup předchozí spuštěné relace (povinný parametr)
- *Session Out* – výstup relace kamery, která je zastavená

Funkce IMAQdx Unconfigure Acquisition

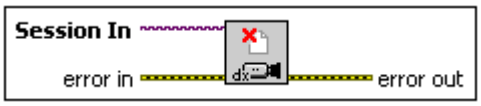
Jedná se o funkci, která ruší nastavení předchozí relace kamery. Vstupem je povinný parametr relace kamery a výstupem je nezkonfigurovaná relace.

Obr. 17 Schéma funkce IMAQdx Unconfigure Acquisition

- *Session In* - vstup relace kamery, kde se má zrušit konfigurace (povinný parametr)
- *Session Out* – výstup relace kamery se zrušenou konfigurací (22)

Funkce IMAQdx Close Camera

Funkce na ukončení relace kamery. Má jenom jediný povinný vstup a to relaci kamery, která se má zrušit.

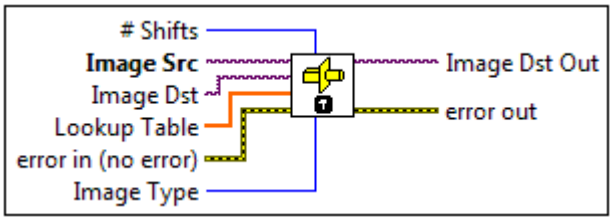


Obr. 18 Schéma funkce IMAQdx Close Camera

- *Session In* – vstup relace kamery, která se má zrušit (povinný parametr)

Funkce IMAQ Cast Image

Tato funkce slouží pro konverzi obrazu na jiný typ. Povinným parametrem je vstupní obraz, který se má konvertovat. Dalším parametrem je alokovaný obraz v paměti pro výstupní obraz. Pokud je konvertován obraz z 8bit obrazu do 16bit, nebo 16bit do 8bit, nebo z 8bit/16bit obrazu do 32bit s desetinnými místy, je možné použít konverzní tabulku. Velmi potřebným parametrem je právě výběr typu, do kterého se bude konvertovat. Posledním vstupním parametrem je určení počtu výměn provedených při konverzi. Výstupem je potom obraz zkonvertovaný na určitý typ.



Obr. 19 Schéma funkce IMAQ Cast Image

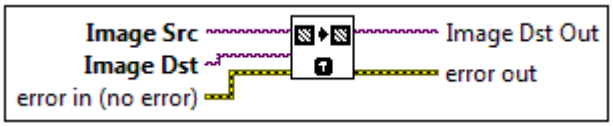
- *Shifts* – definování počtu výměn při konverzi
- *Image Src* – vstupní obraz určený pro konverzi do jiného typu (povinný parametr)
- *Lookup Table* – konverzní tabulka pro převody mezi typy 8bit na 16bit, 16bit na 8bit a 8/16bit na 32bit s desetinným místem
- *Image Type* – výběr obrazového typu, na který se má obraz zkonvertovat. Typy je možné vidět v tabulce Tab. 5
- *Image Out Dst* – výstupní zkonvertovaný obraz (22)

Stupně šedi (U8)	8 bitů na pixel (výchozí hodnota)
Stupně šedi (U8)	16 bitů na pixel
Stupně šedi (U8)	32 bitů na pixel
Komplexní (CSG)	2 × 32 bitů na pixel
RGB (U32)	32 bitů na pixel (červená, zelená modrá, alfa)
HSL (U32)	32 bitů na pixel (odstín, saturace, jas, alfa)
RGB (U64)	64 bitů na pixel (červená, zelená modrá, alfa)
Stupně šedi (U8)	16 bitů na pixel

Tab. 5 Tabulka typů obrazu

Funkce IMAQ Copy

Funkce pro zkopírování vlastností jednoho obrazu do druhého obrazu se zachováním původního. Funkce má dva povinné vstupy, kde jeden je vstupní obraz pro zkopírování a druhým je alokovaný obraz v paměti. Výstupem je zkopírovaný vstupní obraz do alokovaného obrazu v paměti.

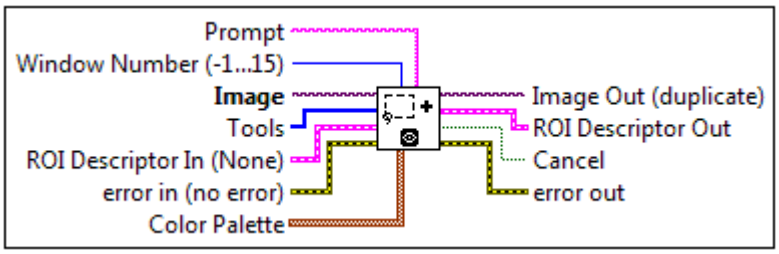


Obr. 20 Schéma funkce IMAQ Copy

- *Image Src* – původní obraz, který se má zkopírovat (povinný parametr)
- *Image Dst* – nový obraz alokovaný v paměti (povinný parametr)
- *Image Dst Out* – Výstupní zkopírovaný obraz

Funkce IMAQ ConstructROI

Funkce pro vytvoření oblasti zájmu. Vstupním povinným parametrem je obraz. Dále je možné nastavit, jaký nástroj bude použit pro výběr oblasti. Je možné definovat výchozí oblast, kterou je možné dále upravovat. Hlavním výstupem je definovaná oblast, která byla vybrána. Více je vidět na schématu funkce a jeho popisu.

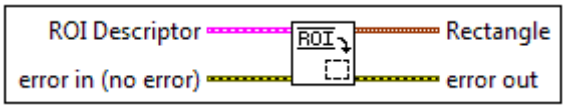


Obr. 21 Schéma funkce IMAQ CostructROI

- *Prompt* – parametr na pojmenování okna
- *Window Number (-1...15)* – Definování čísla okna ve kterém se má obraz zobrazit
- *Image* – vstupní obraz, ve kterém se definuje oblast výběru (povinný parametr) (22)
- *Tools* – Definování nástroje pro výběr oblasti zájmu. Veškeré možnosti jsou popsány v tabulce

- *ROI Descriptor In* – parametr definuje výchozí oblast zájmu, která se dá dále upravovat
- *Color Palete* – Zde je možné určit, v jakých barvách se obraz zobrazí
- *Image Out* – výstupní obraz, který je stejný jako vstupní
- *ROI Descriptor Out* – výstupní definovaná oblast zájmu.
- *Cancel* – výstup logické proměnné zda bylo zmáčknuto tlačítko *cancel*

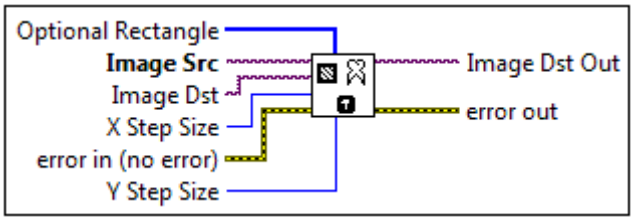
Funkce IMAQ Convert ROI to Rectangle
Funkce pro převedení oblasti zájmu do obdélníku. Vstupem je tedy oblast zájmu a výstupem jsou souřadnice levého horního a spodního pravého rohu.



Obr. 22 Schéma funkce IMAQ Convert ROI to Image

- *ROI Descriptor* – vstupní parametr definované oblasti zájmu
- *Rectangle* – souřadnice horního levého a pravého dolního rohu obdélníku

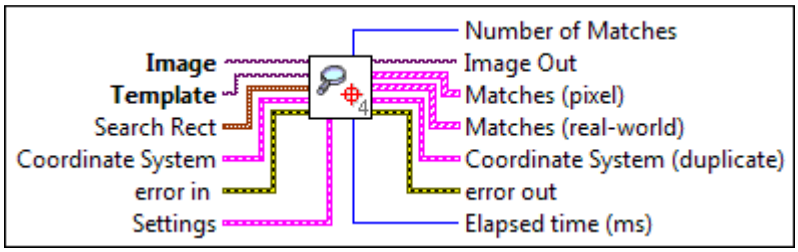
Funkce IMAQ Extract
Tato funkce slouží k vyříznutí určité části obrazu do obrazu nového. Hlavním povinným vstupem je obraz, ze kterého se vyřezává. Poté je možné definovat obdélníkovou část v obraze, která se bude vyřezávat. Výstupem je vyříznutá část obrazu, vložená do alokovaného obrazu v paměti, který byl definován.



Obr. 23 Schéma funkce IMAQ Extract

- *Optional Rectangle* – oblast definovaná pro výřez
- *Image Src* – vstupní obraz, ze kterého se vystřihuje (povinný parametr)
- *Image Dst* – alokované místo v paměti pro vystřižený obraz
- *X Step Size* – definování krokování v ose X
- *Y Step Size* – definování krokování v ose Y
- *Image Dst Out* – výstupní vystřižený obraz

Funkce IMAQdx Find Pattern
Find Pattern je funkce pro nalezení předem daného vzoru v obraze. Hlavními parametry jsou vstupní obraz, ve kterém se má vzor vyhledat a vzor, který se má v obraze vyhledat. Další zajímavou možností je přímo nastavit souřadný systém, do kterého se výstupní souřadnice přetransformují. (22)



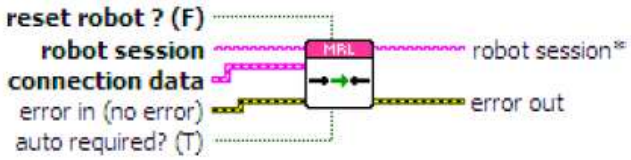
Obr. 24 Funkce IMAQdx Find Pattern

- *Image* – vstupní obraz, ve kterém se bude vyhledávat vzor (povinný parametr)
- *Template* – vzor pro vyhledání v obraze (povinný parametr)
- *Search Rect* – definování oblasti pro vyhledávání
- *Coordinate System* – definování souřadného systému pro transformaci
- *Settings* – zde je možnost nastavit parametry pro vyhledávání. Jedním zásadním parametrem je technika vyhledávání. Jednou je varianta *Shift Invariant*, kde nesmí být obraz otočen o více jak 5°. Druhou možností je volba *Rotation Invariant*, která je pro obrazy, které jsou otočeny více. Dalšími možnostmi jsou volby jako požadovaný počet shod, jak přesně musí být obraz určen, maximální otočení atd.
- *Number of Matches* – udává počet nalezených vzorů v obraze
- *Image Out* – výstupní obraz s označenými nalezenými vzory
- *Matches (pixel)* – pole s hodnotami o nalezených vzorech v pixelech
- *Matches (real-world)* – pole s hodnotami nalezených vzorů v souřadnicích definovaného souřadného systému
- *Coordinate System* – duplikovaný předem definovaný souřadný systém
- *Elapsed time (ms)* – čas potřebný na nalezení vzorů v obraze (22)

5.2 *ImagingLab*

Knihovna *ImagingLab* je vyvinuta pro řízení robotů od firem, jako jsou Denso, Kuka, Mitsubishi atd. Tyto roboty je většinou možné ovládat pouze za pomoci dodávaných programů, jako již zmíněný RT Tool Box, COSIMIR atd. Za pomoci těchto programů je možné pouze pracovat s pevným programem, který jde ovlivňovat pouze signály na vstupech/výstupech řídicí jednotky. Knihovna *ImagingLab* právě umožňuje se k robotu připojovat a řídit z prostředí *LabVIEW*, ať už funkcemi přímo pro řízení, nebo funkcemi pro editaci, spouštění a ukončování programů. Níže jsou popsány funkce používané pro řízení robotu Mitsubishi *RV-2AJ* s jednotkou CR1-571 za pomoci tohoto modulu. (23)

Funkce *Open Session*
Funkce *Open Session* slouží pro otevření komunikace mezi řídicí jednotkou robotu a počítačem. Funkce má tři povinné vstupní parametry (resetování, název relace a data pro nastavení a navázání komunikace) a jeden volitelný (zda je vyžadován mód Auto na řídicí jednotce). Výstup je jeden a to relace mezi řídicí jednotkou a počítačem.

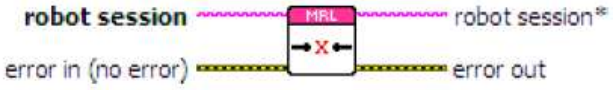


Obr. 25 Funkce Open Session

- *reset robot ?* – nastavení zda se má po navázání komunikace resetovat chybová hlášení (povinný)
- *robot session* – pojmenování relace robotu (povinný)
- *connection data* – data potřebná pro navázání komunikace mezi řídicí jednotkou a počítačem. Jedná se o zvolení příslušného sériového portu, přenosové rychlosti, IP adresy (pokud používáme ethernetovou komunikaci), port (pokud používáme ethernetovou komunikaci), typ robotu (5osý, 6osý) a typ řídicí jednotky (500, 700)
- *auto required?* – volba, zda je nutné, aby řídicí jednotka byla v módu Auto
- *robot session** - jméno relace robotu, která byla spuštěna

Funkce Close Session

Funkce pro ukončení relace robotu. Má pouze jeden parametr a to jméno relace, která se má ukončit.

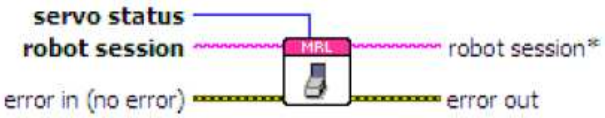


Obr. 26 Funkce Close Session

- *robot session* – jméno relace robotu, která se má ukončit (povinný)
- *robot session** - jméno relace robotu pro další možné použití

Funkce Servo ON/OFF

Funkce sloužící pro zapnutí nebo vypnutí servomotorů robotu. Má dva povinné parametry. Jeden je volba zdali zapnout nebo vypnout servomotory a druhá jméno relace robotu, u kterého se tak má stát. Nepovinným výstupem je pak dále jméno relace robotu.



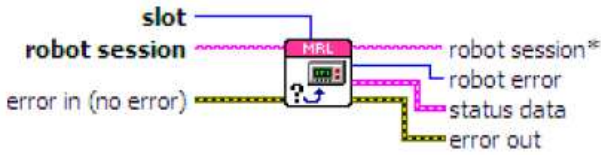
Obr. 27 Funkce Servo ON/OFF

- *servo status* – volba zapnutí nebo vypnutí servomotorů (povinný)
- *robot session* – jméno relace robotu, kde se má funkce provést (povinný)
- *robot session** - výstup jména relace pro další použití (24)

Funkce Get Status

Jedná se o funkci pro zjištění různých stavů robotu a řídicí jednotky. Funkce má dva povinné parametry, kde jedním je číslo slotu (definování slotu, ve kterém

běží určitý program) a jména relace robotu. Poté je možné získávat data o aktuálním stavu robotu.

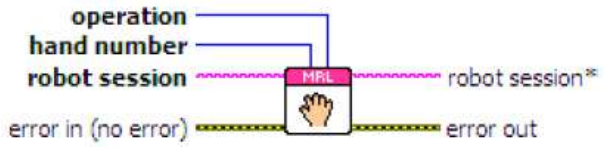


Obr. 28 Funkce Get Status

- *slot* – výběr slotu, ve kterém běží požadovaný program, u kterého má být stav zjišťován (povinný)
- *robot session* – jméno relace robotu (povinný)
- *robot session** - výstup jména relace robotu pro další použití
- *robot error* – výstup chybových kódů
- *status data* – proměnná obsahující data o stavech robotu jako například, stav servomotorů, zdali běží program nebo je skončen, v jakém je řídicí jednotka módu apod.

Funkce Open/Close Hand

Funkce pro otevření nebo zavření kleštin koncového efektoru robotu. Hlavními a povinnými vstupy jsou jméno relace robotu, číslo identifikující koncový efektor a operace, kterou má koncový efektor provést.

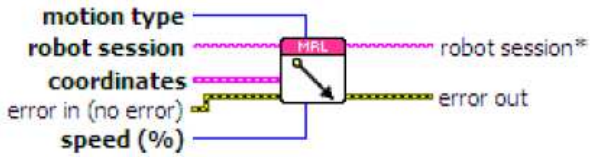


Obr. 29 Funkce Open/Close Hand

- *operation* – operace, kterou má koncový efektor provést (povinný)
- *hand number* – definování, se kterým koncovým efektorem bude operováno (povinný)
- *robot session* – jméno relace robotu (povinný)
- *robot session** - výstup jména relace robotu pro další použití

Funkce Move

Jedná se o funkci pro pohyb robotického ramena na definované kartézské souřadnice. Je možné nastavit, jakou trajektorií bude pohyb proveden a jakou rychlostí.



Obr. 30 Funkce Move

- *motion type* – nastavení trajektorie pohybu (povinný)

- *robot session* – jméno relace robotu (povinný)
- *coordinates* – struktura definující kartézské souřadnice, na které se má koncový efektor robotického ramena přemístit (povinný)
- *speed (%)* – nastavení rychlosti robotického ramene (povinný)
- *robot session** - výstup jména relace robotu pro další použití (24)

Funkce Wait Motion Stop

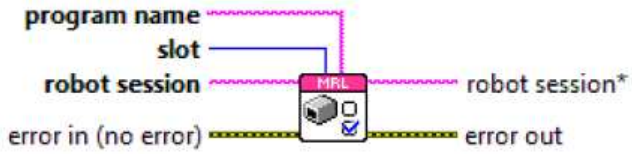
Funkce, která čeká, dokud se nevykoná definovaný pohyb.

Obr. 31 Funkce Wait Motion Stop

- *robot session* – jméno relace robotu (povinný)
- *robot session** - výstup jména relace robotu pro další použití

Funkce Load Program

Tato funkce slouží pro načtení daného programu do definovaného slotu.

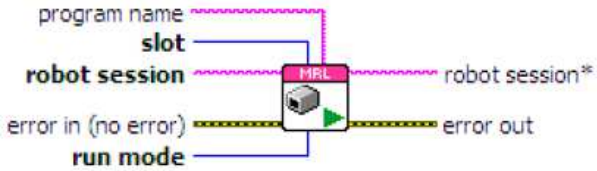


Obr. 32 Funkce Load Program

- *program name* – jméno programu, který se má načíst (povinný)
- *slot* – slot, do kterého se program načte (povinný)
- *robot session* – jméno relace robotu (povinný)
- *robot session** – výstup relace robotu pro další použití

Funkce Start Program

Funkce sloužící pro spuštění programu. Program musí být předem připraven a načten do slotu.

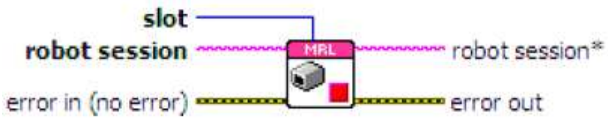


Obr. 33 Funkce Start Program

- *program name* – jméno programu, který se má spustit
- *slot* – slot, kde je nahraný příslušný program (povinný)
- *robot session* – jméno relace robotu (povinný)
- *run mode* – režim spuštění; zdali má být proveden na začátku reset, či ne (povinný)
- *robot session** - výstup jména relace robotu pro další použití (24)

Funkce Stop Program

Funkce pro ukončení programu načteného v řídící jednotce v určitém slotu.

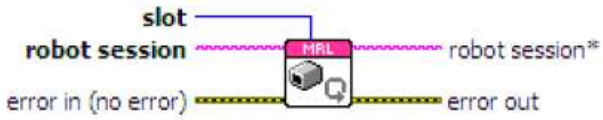


Obr. 34 Funkce Stop Program

- *slot* – slot, ve kterém je načten program, který chceme zastavit (povinný)
- *robot session* – jméno relace robotu (povinný)
- *robot session** - výstup jména relace robotu pro další použití

Funkce Reset Program

Funkce pro resetování programu v řídící jednotce robotu.

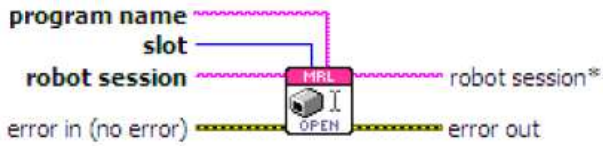


Obr. 35 Funkce Reset Program

- *slot* – slot, ve kterém je načten program, který chceme zastavit (povinný)
- *robot session* – jméno relace robotu (povinný)
- *robot session** - výstup jména relace robotu pro další použití

Funkce Open Program Edit

Funkce pro otevření editace příslušného programu.

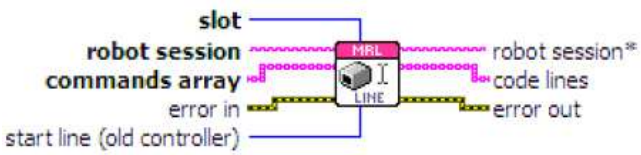


Obr. 36 Funkce Open Program Edit

- *program name* – jméno programu, který se má spustit
- *slot* – slot, kde je nahraný příslušný program (povinný)
- *robot session* – jméno relace robotu (povinný)
- *robot session** - výstup jména relace robotu pro další použití

Funkce Edit Program Line

Funkce pro editaci nebo vytvoření programu.

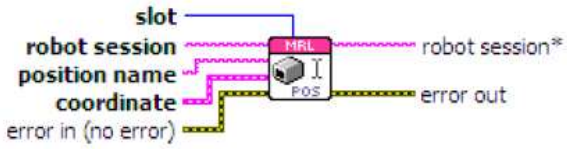


Obr. 37 Funkce Edit Program Line

- *slot* – slot, ve kterém se program bude provádět (povinný)
- *robot session* – jméno relace robotu (povinný)
- *commands array* – pole s jednotlivými řádky kódu (povinný)
- *start line* – číslo řádku, od kterého se má program vykonávat
- *robot session** - výstup jména relace robotu pro další použití
- *code lines* – výstup pole s jednotlivými řádky programu (24)

Funkce Edit Program Position

Funkce pro editaci pozice v programu. Funkce otevře program, danou pozici změní a program uloží.

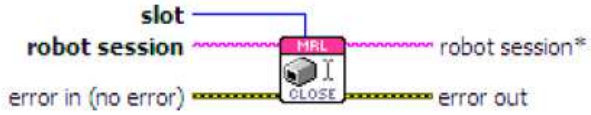


Obr. 38 Funkce Edit Program Position

- *slot* – slot, ve kterém se program bude provádět (povinný)
- *robot session* – jméno relace robotu (povinný)
- *position name* – jméno pozice, která se má změnit (povinný)
- *coordinate* – souřadnice, na které se má pozice změnit (povinný)
- *robot session** - výstup jména relace robotu

Funkce Close Edit Program

Funkce pro ukončení editace programu.

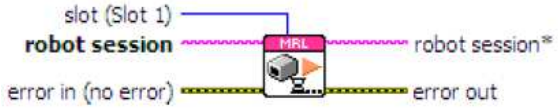


Obr. 39 Funkce Close Edit Program

- *slot* – slot, ve kterém se program bude provádět (povinný)
- *robot session* – jméno relace robotu (povinný)
- *robot session** - výstup jména relace robotu

Funkce Wait Program End

Funkce, která čeká na řádné dokončení programu.



Obr. 40 Funkce Wait Program End

- *slot (Slot 1)* – slot, ve kterém se program bude provádět (povinný)
- *robot session* – jméno relace robotu (povinný)
- *robot session** - výstup jména relace robotu (24)

Funkce Get Current Position
Funkce, která vrací pozici koncového efektoru dané relace robotu. Je možné vybrat, jaké souřadnice získáme (cartesian, cylindrical, joint, XYZ465)

Obr. 41 Funkce Get Current Position

- *robot session* – jméno relace robotu (povinný)
- *robot session** - výstup jména relace robotu
- *coordinate* – pole se souřadnicemi dané relace robotu (24)

5.3 *Vision Builder for Automated Inspection*

Jedná se o další rozšiřující modul do vývojového prostředí *LabVIEW,* za pomoci kterého je možné provádět pokročilejší metody analýzy obrazu jako je například detekce hran, vyhledání vzoru v obraze, měření tolerancí a podobně. To vše probíhá za pomoci jednoduchého průvodce na vytváření aplikace. Ten je schopen vytvořit i kompletní složku s podprogramy pro kompletní aplikaci. Modul je vhodný na odzkoušení některých jednoduchých aplikací v pokročilé analýze obrazu, ale pro vytvoření kompletního programu pro řešení komplexnější úlohy je nevhodný. Postrádá totiž možnost zasahovat do blokových diagramů, které vytváří. Do těch je možné zasáhnout až v případě vyexportování celého programu. Tím vznikne složka se spoustou podprogramů a jedním hlavním programem. V blokových diagramech je těžké se vyznat, protože modul generuje mnoho věcí navíc a diagramy jsou složité. Proto tento modul je využíván pouze na odzkoušení funkčnosti, popřípadě řešení nějakých nejasností s implementováním některých funkcí. (21)

6 Metodika

V praktické části budou nejdříve odzkoušeny možné způsoby snímání obrazu z kamery Basler acA1600-20uc z vývojového prostředí *LabVIEW* za pomoci knihoven *Vision Acquisition software, Vision Builder for Automated Inspection a Vision Development*. Po zajištění snímání obrazu z kamery budou zkoumány možné způsoby detekce značek nebo objektů v obraze za účelem zjištění jejich polohy. Poloha bude využívána na naplnění transformačních matic pro transformaci souřadného systému kamery do souřadného systému robotu. Zde bude nutné zajistit i transformaci obrazu při pootočení kamery.

Po zajištění transformace souřadného systému obrazu z kamery do souřadného systému robotu bude zapotřebí navrhnout způsob detekce objektů, které budou určeny k manipulaci. Bude vytvořen program, na kterém budou možnosti detekce testovány. Po nalezení vhodného způsobu bude vytvořen program složený z programu pro transformaci souřadného systému a programu pro detekci objektů. V něm bude testováno, zdali je schopen detekovat objekty a přepočítávat jeho souřadnice v obraze na souřadnice v souřadném systému robotu.

V další části bude zapotřebí prozkoumat možnosti řízení robotického ramene Mitsubishi *RV-2AJ*. Nejdříve bude zajištěna kalibrace robotu a zjištění zdali souhlasí data poskytovaná řídicí jednotkou s realitou. Poté bude vytvořen program na řízení robotického ramene za pomoci knihovny *ImagingLab*. Na tomto programu bude zkoumáno, zdali bude možné řídit robot za pomoci dat získaných předešlými programy. Následně se program zase spojí s předchozím programem pro detekci objektů a bude testována funkčnost.

Aby bylo možné detekovat více předmětů pro manipulaci, bude nutné vytvořit další část programu pro identifikování objektů. Bude vycházeno z předchozích programů a uzpůsobeno tak, aby bylo možné program zakomponovat do již vytvořených programů. Mělo by zde být možné zadat data o objektu jako například jeho výšku, nebo místo, kam bude přemisťován.

Protože bude kamera i robot po většinu doby na jednom místě a nebude s ním manipulováno, měl by program zvládat uložení hodnot pro transformaci a hodnot pro nastavení robotického ramene. Tyto hodnoty se pak budou moci načíst při dalším spouštění. Odstraní se tím nutnost pokaždé provést zaměření bodů pro naplnění transformačních matic při spuštění programu.

V neposlední řadě bude nutné program a celou soustavu otestovat. Za tímto účelem bude vytvořena testovací úloha s několika typy objektů. Ty bude za úkol roztřídit do předem určených boxů. Testování bude provedeno vícekrát i s opakovaným vytvořením transformačních matic při pootočené kameře, či změně zaměřujících bodů.

V závěru bude diskutováno nad výsledky testu a nad možnými způsoby vylepšení.

7 Návrh a implementace řídicího software

V následujících kapitolách praktické části je popisován, postup při řešení problému s detekcí objektů v obraze a jejich následnou manipulací za pomoci robotického ramene. Veškeré programy jsou vytvořeny ve vývojovém prostředí *LabVIEW* a jeho knihoven. Kapitola je rozdělena na tři stěžejní části, kde nejdříve je popsán způsob snímání obrazu z kamery Basler acA1600-20uc a jeho transformace, tedy přepočtení do souřadného systému robotu. Další podkapitola je zaměřena na detekci objektů v obraze a za pomoci programu z předchozí podkapitoly, zjištění jeho reálných souřadnic. Poslední podkapitolou je popis způsobu řízení robotického ramene Mitsubishi *RV-2AJ*.

7.1 Transformace souřadného systému kamery do souřadného systému robotu

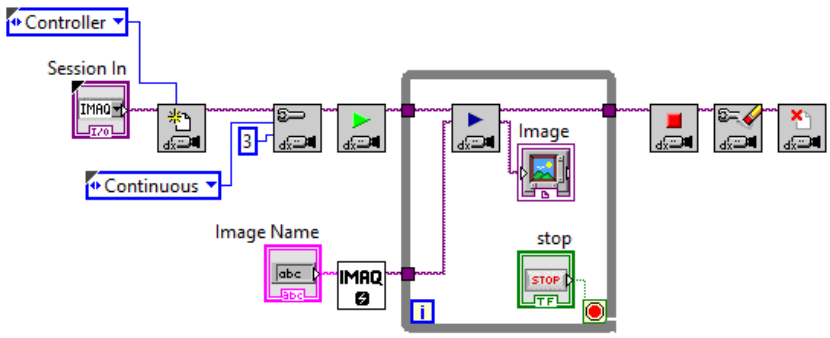
Dříve, než je možné začít s návrhem programu pro snímání obrazu z kamery, je zapotřebí namontovat kartu pro komunikaci s kamerami s rozhraním USB3.0 do počítače. Po nainstalování potřebných ovladačů a přiložené knihovny *Vision Acquisition software* do vývojového prostředí *LabVIEW* je odzkoušeno, zda je možné se ke kameře připojit. To je provedeno za pomoci softwaru *Measurement & Automation Explorer*, který slouží pro kompletní správu hardwaru od *National Instruments*.
V následující podkapitole je popsán způsob získávání obrazu z kamery Basler acA1600-20uc. V dalších podkapitolách je popsán způsob, jakým se detekují značky sloužící pro vytvoření transformačních matic. V poslední podkapitole je popsán způsob transformace souřadného systému obrazu z kamery do souřadného systému robotu.

7.1.1 Získávání obrazových dat z kamery

Pro získávání obrazových dat z kamery jsou ve vývojovém prostředí *LabVIEW* knihovny *IMAQ* a *IMAQdx*. Veškeré funkce ze zmíněných knihoven, které jsou v následujících programech použity, jsou popsány v kapitole 5.1.

Aby bylo možné snímat obraz, je zapotřebí inicializovat kameru, která se má použít. Pro tento účel slouží funkce `IMAQdx Open`, která umožňuje výběr kamery z výčtu kamer a následně otevře relaci zvolené kamery. Je zde možnost nastavení režimu kamery, ale ten je ponechán na výchozí hodnotě *Controller*, kdy je možné kameru nastavovat i přijímat data. Dále je pro možné nastavování snímání použita funkce `IMAQdx Configure Acquisition`. Relace je nastavena na kontinuální snímání a výchozí 3 vyrovnávací buffery. Aby bylo možné z kamery číst, musí se relace spustit a to pomocí funkce `IMAQdx Start Acquisition.` Pro zajištění kontinuálního snímání je použita funkce `IMAQdx Grab`. K této funkci je nutné přiřadit alokovaný obraz v paměti, což je provedeno za pomoci funkce `IMAQ Create`. Protože se jedná o kontinuální snímání, je nutné funkci `IMAQdx Grab` vložit do cyklu, aby se prováděla opakovaně, dokud není ukončena. V jiném případě se nasnímá pouze jeden

snímek a program se ukončí. V tuto chvíli je možné kontinuálně snímat data z kamery a zobrazovat je na výstupu pomocí funkce *Image*. Ta vytvoří na *Front panelu* obrazovku, na které se obraz zobrazí. Po ukončení cyklu, tím pádem i ukončení snímání je nutné relaci kamery zastavit a ukončit, aby čímž je kamera připravena pro další použití. To je provedeno pomocí funkcí *IMAQdx Stop Acquisition* a *IMAQdx Close.* Ukázka programu pro snímání je na obrázku Obr. 42



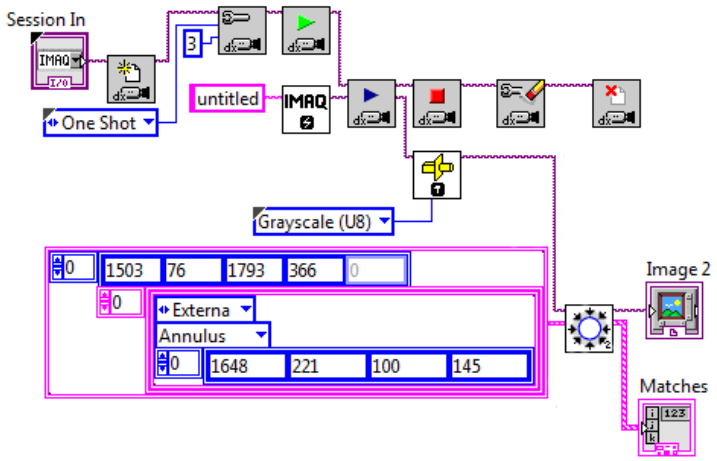
Obr. 42 Blokový diagram pro snímání obrazu z kamery

Při řešení snímání obrazu z kamery se nevyskytuje žádný problém a program je schopen plynule snímat obraz. Obraz je jen zapotřebí manuálně zaostřit a upravit světelnost pomocí objektivu Computar M2514-MP2. Je také otestováno snímání v režimu *OneShot*, který pořídí určitý počet snímků podle nastavení parametru *Number of Buffers* ve funkci *IMAQdx Configure Acquisition.*

7.1.2 Návrh a implementace způsobu detekce značky

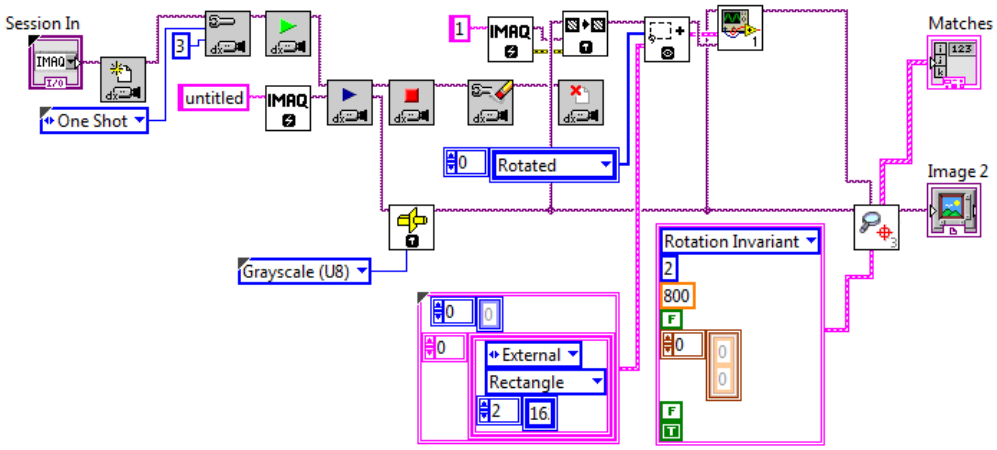
Pro provedení transformace souřadného systému je zapotřebí nějakým způsobem nalézt v obraze značky a zjistit jejich polohu. Po projití funkcí nabízených ve vývojovém prostředí *LabVIEW*, se nabízí několik funkcí. Z těchto funkcí jsou vybrány dvě, které jsou testovány. Jednou je detekce kružnice v obrazu a druhou je vyhledání vzoru v obraze. Obě funkce pracují s obrazem ve stupních šedi. K tomu je použita funkce *IMAQ Cast Image*, která obraz do stupňů šedi převede. Funkce poté musí být zapojena mezi funkcí *IMAQdx Grab* a jednou nebo druhou funkcí pro detekci značky.

Pro vyzkoušení funkčnosti je použit předchozí program na snímání obrazu. Do smyčky k funkci *IMAQdx Grab,* je přidána funkce pro detekci kruhové hrany *IMAQdx Find Circular Edge*. U této funkce je důležité správně definovat oblast zájmu pro detekci (ROI – Region of Interest). Oblast musí být kruhového typu (angular), jinak funkce končí s chybou, že je špatně definována oblast zájmu. Výstupem funkce je poté obraz a kruhová hrana detekovaná v definované oblasti zájmu. Připojením výstupu obrazu z funkce na funkci *Image* získáme vyznačenou nalezenou kruhovou hranu. Na dalším výstupu funkce pak informace o nalezené kruhové hraně, jako je například poloměr, souřadnice středu atd. Blokový diagram je možné vidět na obrázku Obr. 43.



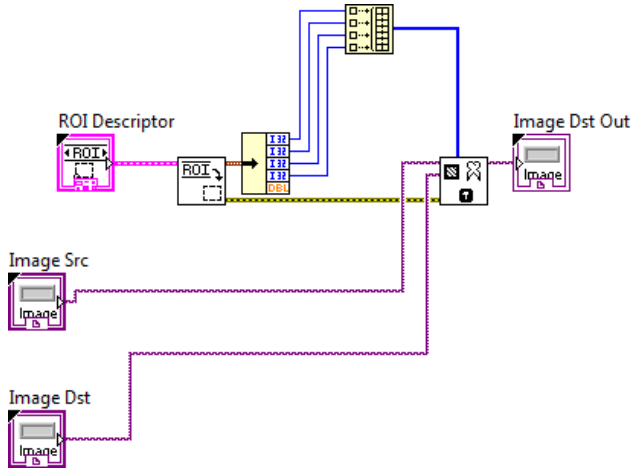
Obr. 43 Blokové schéma pro detekci značky pomocí funkce *Find Circular Edge*

Druhou variantou je použití funkce pro detekci vzoru v obraze *IMAQdx Find Pattern 3.* Stejně jako u předchozí funkce, je umístěna ve smyčce společně s funkcí pro snímání obrazu. Aby funkce na detekci fungovala, je zapotřebí zajistit vzor. Pro nejlepší výsledky je dobré použít vzor vystřižený přímo ze snímaného obrazu. Pro první testy je použito uložení obrazu z kamery na disk a vzor následně vystřižen mimo vývojové prostředí *LabVIEW*. Ten je poté načten do *LabVIEW* a následně použit ve funkci *IMAQdx Find Pattern 3.* Stejně jako ve funkci *IMAQdx Find Circular Edge,* po připojení výstupu obrazu na funkci *Image*, získáme vyznačené nalezené značky. Jak z předchozí věty vyplývá, funkce dokáže nalézt vzor v obraze vícekrát. Data o nalezených vzorech jsou poté ve výstupu *Matches*. Funkce má také několik důležitých nastavení. Jedním z nich je technika vyhledávání. V nabídce jsou dvě a to *Shift Invariant,* kde nesmí být obraz otočen o více jak 5° a *Rotation Invariant*, která je schopna detekovat i více otočené obrazy. V tomto případě je používána technika *Rotation Invariant*. Dalším důležitým nastavením je požadovaná přesnost shody. Tímto nastavení je možné vyloučit detekování podobného, ale přitom nesprávného vzoru. Posledním důležitějším nastavením je definování počtu hledaných vzorů. Pokud víme předem, kolik jich hledáme, je možné to ve funkci nastavit a tím urychlit její práci. Blokový diagram pro detekci značky za pomoci vyhledání vzoru v obraze je možné vidět na obrázku Obr. 44.



Obr. 44 Blokový diagram pro detekci značky v obraze pomocí funkce *Find Pattern*

Výše je ještě zmiňována nutnost definování vzoru, který se má hledat a jeho způsob vytvoření mimo prostředí *LabVIEW*. Protože vystřihovat vzor pokaždé mimo *LabVIEW*, když má být provedena detekce je neefektivní, tak je vytvořeno vystřižení vzoru přímo ve vývojovém prostředí. Za prvé je tak obstarán přesný výřez z obrazu a za druhé je to efektivnější. Vystřižení obrazu funguje tak, že je nasnímán jeden snímek za pomoci programu pro snímání obrazu. Ten je následně zkopírován pomocí funkce *IMAQ Copy* a pomocí funkce *IMAQ Construct ROI* vytvořena oblast zájmu, kterou chceme vyříznout. Následně je za pomoci funkcí *IMAQ Extract* a *IMAQ Convert ROI to Rectangle* vyříznut. Takto vyříznutý obraz je následně použit ve funkci na detekci vzoru v obraze. Vyříznutí podle definované oblasti je vidět na obrázku Obr. 45.

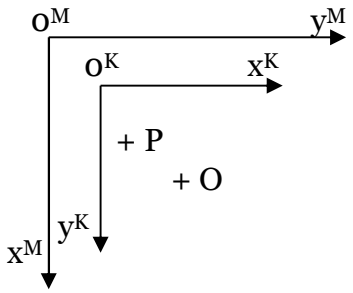


Obr. 45 Blokový diagram pro vystřižení obrazu

7.1.3 Návrh způsobu transformace souřadných systémů

Při návrhu je nejdříve určeno, co je potřeba u obrazu transformovat. V tomto případě se jedná o transformaci souřadného systému v pixelech v obraze do souřadného systému robotu v reálných jednotkách (mm). V ideálním případě, kdy by kamera byla umístěna přesně, tedy kolmá na pracovní prostor a obraz rovnoběžný s osou Y robotu, pak by stačil pouze jeden přesně zaměřený bod v souřadném systému robotu s určitou velkostí. Pomocí těchto hodnot je možné

provést převedení pixelových souřadnicových hodnot na souřadnice v mm. To může být provedeno zjištěním velkosti značky v pixelech a podělením reálné velikosti značky v mm. Tím se získá přepočtový koeficient, který udává kolik pixelů je jedem mm. Pro tento způsob provedení přepočtů na souřadnice v souřadném systému robotu je zapotřebí uchovávat pět hodnot. První dvě jsou souřadnice značky v obraze v pixelech. Druhé dvě hodnoty jsou souřadnice značky v mm v souřadném systému robotu a poslední je koeficient pro přepočet pixelů na mm. Jediným problémem je, že souřadný systém robotu a souřadný systém kamery není stejně orientovaný, jak je vidět na obrázku Obr. 46. Ten lze, ale jednoduše vyřešit obrácením souřadnicových hodnot z kamery.



Obr. 46 Rovnoběžný souřadný systém robotu a kamery

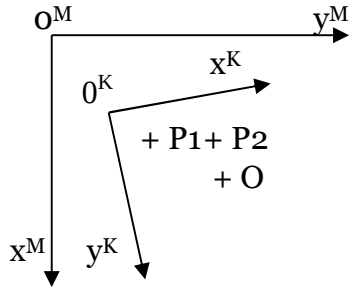
Pokud jsou tedy k dispozici souřadnice nějakého objektu v obraze, přepočet vypadá následovně:

$$x = \frac{y_O^K - y_P^K}{koef} + x_P^M \tag{14}$$

$$y = \frac{x_O^K - x_P^K}{koef} + y_P^M \tag{15}$$

kde x, y jsou přepočtené souřadnice v souřadném systému robotu
x_O^K, y_O^K jsou souřadnice detekovaného objektu v obraze v pixelech
x_P^K, y_P^K jsou souřadnice detekované značky v obraze v pixelech
koef je koeficient přepočtu pixelů na mm
x_P^M, y_P^M jsou souřadnice značky v souřadném systému robotu

Po většinou, ale nemusí být kamera přesně uchycena a tím není možnost zajistit rovnoběžnost obrazu kamery s osou Y robotu. Pak už nepostačuje pouze jedna přesně zaměřená značka, ale značky dvě, jak je možné vidět na obrázku Obr. 47.



Obr. 47 Nerovnoběžný souřadný systém robotu a kamery

Pomocí dvou značek s přesnými souřadnicemi (body P1 a P2 v obrázku Obr. 47) je možné zjistit i natočení kamery. Přesnost zjištění natočení je odvislá od přesnosti zaměření detekčních značek. První co je zapotřebí, je zkontrolovat zdali nejsou pootočeny zaměřené body vůči souřadnému systému robotu. To je provedeno výpočtem pomocí goniometrických funkcí. Stejným způsobem je zjištěno, v jakém úhlu se na značky dívá kamera. Úhel značek je odečten od úhlu pohledu kamery a tím je zajištěno otočení pouze kamery. Z úhlu, který je zjištěn, se vytvoří transformační matice pro otočení obrazu kolem počátku. Koeficient pro přepočet pixelů na milimetry se řeší podobně, jako u jednoho bodu. Jako velikost se ale bere vzdálenost mezi body z důvodu větší přesnosti. Pomocí Pythagorovy věty je zjištěna vzdálenost mezi body jak v souřadném systému robotu, tak souřadném systému kamery. Tyto hodnoty jsou jako v předchozím způsobu poděleny, čímž získáme přepočtový koeficient.

7.1.4 Implementace programové části pro zjištění transformačních hodnot

Při implementaci programu pro zjištění transformačních hodnot jsou použity předchozí dva programy pro snímání obrazu a detekci značky. Tyto dva programy jsou rozšířeny o výpočet úhlu pro transformační matici a zjištění souřadnic detekčních značek v obraze kamery. Snímání je nastaveno na pořízení jednoho snímku. Z toho je následně vystřižen vzor, který je předán funkci na nalezení vzoru v obraze (*IMAQdx Find Pattern*). Tato funkce je nastavena na techniku hledání *rotation invariat*, která nalezne vzory, které jsou pootočeny i o více jak 5°. Počet hledaných vzorů je nastaven na 2 s tím, že se kontroluje, zdali jsou detekovány opravdu dvě detekční značky. Pokud ne, je tato chyba indikována červenou diodou ve Front Panelu a musí se proces zopakovat. Blokové schéma pro výpočet a sestavení je z důvodu velkosti v příloze C.

Při testování výpočtů se vyskytuje jeden problém. Tím je detekování přesně zaměřených značek v obráceném pořadí. Jednou je detekován nejprve bod P1 a poté P2 a jindy nejdříve P2 a pak P1. Tím vznikají velké chyby ve výpočtu. Řešení tohoto problému ale není nijak složité. Protože jsou známé souřadnice bodu v souřadném systému robotu, tak porovnáním se souřadnicemi v souřadném systému kamery, je možné zjistit, zdali jsou body detekovány správně. Například, pokud bod P1 je dále v ose x souřadného sytému robotu než bod P2, musí být bod P1 také dále v ose y souřadného systému kamery než bod P2. V případě, že tak není, jsou body detekovány obráceně a je nutné je

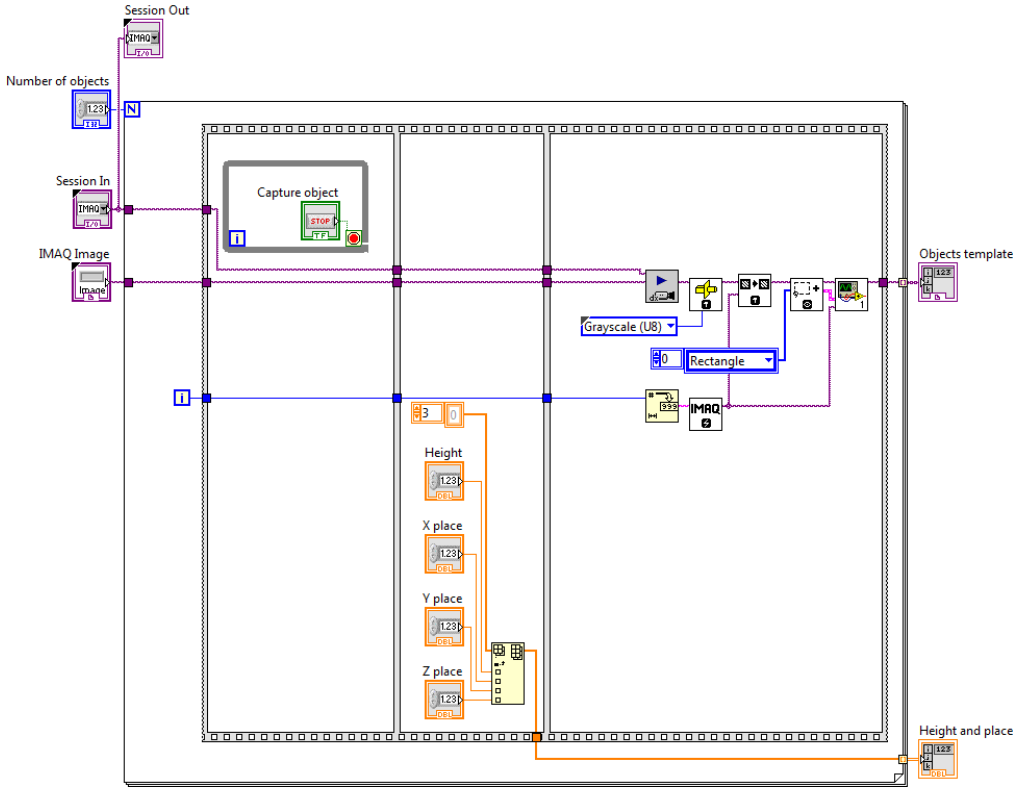
prohodit. V programu je detekce prohození implementována řadou hlídajících podmínek kontrolující stavy popsané výše.

Protože ne pokaždé je nutné vytvářet transformační matici a zjišťovat detekční značky pro přepočty souřadnic, je v programu část zabývající se uložením konkrétních hodnot pro pozdější využití. Ta ukládá hodnoty potřebné pro transformaci souřadných systému do souboru, odkud mohou být při dalším spuštění načteny. Tato část programu se volá pokaždé, když proběhne detekce značek a přepočet správně s dotazem, zdali chceme transformační hodnoty uložit, či ne. Uložení hodnot je provedeno stylem zápisu klíč-hodnota do textového souboru. Načtení pak probíhá tak, že jsou dle klíčů nalezeny hodnoty a použijí v programu. Bloková schémata pro uložení a načtení hodnot je možné vidět na obrázcích v příloze D.

7.2 Detekce objektů pro manipulaci

V této kapitole je popisován způsob detekce objektů určených k manipulaci. Podprogram řešící detekci objektů vychází z předchozí programové části pro detekci značek pro zajištění transformačních hodnot. Tak jako v programové části pro detekci značek, je zapotřebí nejdříve určit objekt, který se má v obraze vyhledávat. V tomto případě je nasnímán daný objekt, udělán výřez (vzor) a uložen do paměti, tak jako u detekčních značek. Vzor je poté použit ve vyhledávací funkci `IMAQdx Find Pattern`. Funkce je však tentokrát ve smyčce a provádí se, dokud není ukončena uživatelem. Počet detekovaných objektů je nastaven na jeden. Do budoucna je ale možné upravit celý program pro detekci více stejných objektů najednou a tím urychlit práci robotu. Vyhledávací technika je stejně jako při detekci značek nastavena na *Rotation Invariant*. Jakmile je tedy vzor v obraze nalezen, jsou jeho souřadnice zpracovány programovou částí na přepočet souřadnic do souřadného systému robotu. Výstupem jsou reálné souřadnice v souřadném systému robotu, které budou použity v programové části řešící řízení robotického ramene. Tato programová část je popsána v kapitole 7.3. Po zkombinování s programovou částí pro zjištění transformačních hodnot, je zkoušeno, zda je program schopen správně detekovat objekt a určit jeho přesné reálné souřadnice. Pokud jsou dobře zaměřeny detekční značky a zjištění transformačních hodnot proběhne správně, je přesnost zjištění reálných souřadnic v souřadném systému robotu velmi dobrá.

Protože je zapotřebí často detekovat více než jeden objekt, je k programové části přidána možnost načtení do paměti více objektů. V této části se nejprve zvolí počet objektů určených k detekci. Poté se každý objekt musí nasnímat a vytvořit vzor vystřižením z obrazu. K tomu jsou použity programové části popsané v předchozích kapitolách. Pro budoucí použití jsou každému objektu ještě přiřazeny další data, která jsou potřebná při manipulaci za pomoci robotického ramene. Jedná se o výšku objektu, která je zapotřebí kvůli uchopení objektu a o souřadnice, kam se má objekt přesunout. Blokové schéma je možné vidět na obrázku Obr. 48.



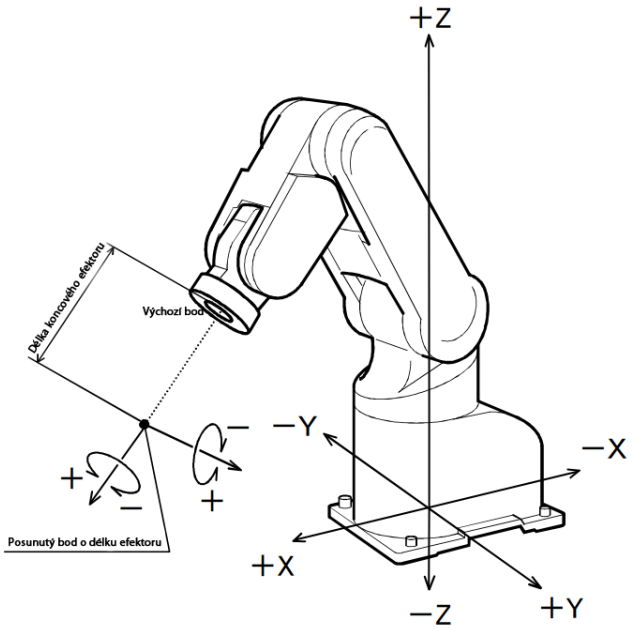
Obr. 48 Blokové schéma pro načtení více objektů k detekci

7.3 Manipulace s objekty

Před samotnou implementací programové části pro řízení robotického ramene Mitsubishi Melfa RV-2AJ, je zapotřebí zkontrolovat jeho funkčnost. Nejprve je zkontrolováno zapojení mezi řídící jednotkou a robotickým ramenem. Po zapnutí řídicí jednotky je zjištěna chyba týkající se baterií v robotickém rameni, za pomoci kterých je uchovávána poloha číslicových snímačů polohy robotického ramene. Po prostudování dokumentace k robotu je zjištěno, že se jedná o překročení intervalu výměny baterií. Ty jsou následně vyměněny a robot nově zkalibrován. Po těchto úkonech je zkontrolováno, zda souhlasí souřadnice poskytované řídicí jednotkou s reálnou polohou koncového efektoru. Veškeré ovládání je řešeno za pomoci ovládacího panelu *Teach box R28TB*, který je popsán v kapitole 4.1.2. Řídicí jednotka je při tomto typu ovládání nastavena do režimu *Teach* a ovládací panel musí být přepnutý do stavu *Enable*.

Při kontrole je zjištěna odchylka ve všech třech osách *x, y* a *z*. Jednou z možných příčin je špatně provedená kalibrace, která je ale hned vyloučena opětovnou kalibrací robotu, která nepomohla. Po dalším studování dokumentace je zjištěno, že je možné nastavit posunutí souřadného systému robotu. Při kontrole je zjištěno, že je souřadný systém posunut ve všech třech osách. Nastavení posunutí souřadného systému se provádí pomocí změny parametru *MEXBS*. Ten se nastavuje v menu řídicí jednotky za pomoci ovládacího panelu *Teach box R28TB*. Po nastavení výchozího umístění souřadného systému, jsou souřadnice koncového efektoru stále posunuty. Je zjištěno, že není nastavena délka koncového efektoru. Řídicí jednotka udává ve výchozím nastavení souřadnice vztažené k rovině upnutí koncového efektoru, viz obrázek Obr. 49. Aby byly vztaženy ke konci efektoru, je nutné nastavit

posun v ose *z* pomocí parametru *MEXTL1*. Délka koncového efektoru je zjištěna z technické dokumentace robotu. Po změně tohoto parametru a kontroly souřadnic je již vše v pořádku.



Obr. 49 Poloha souřadnic koncového efektoru

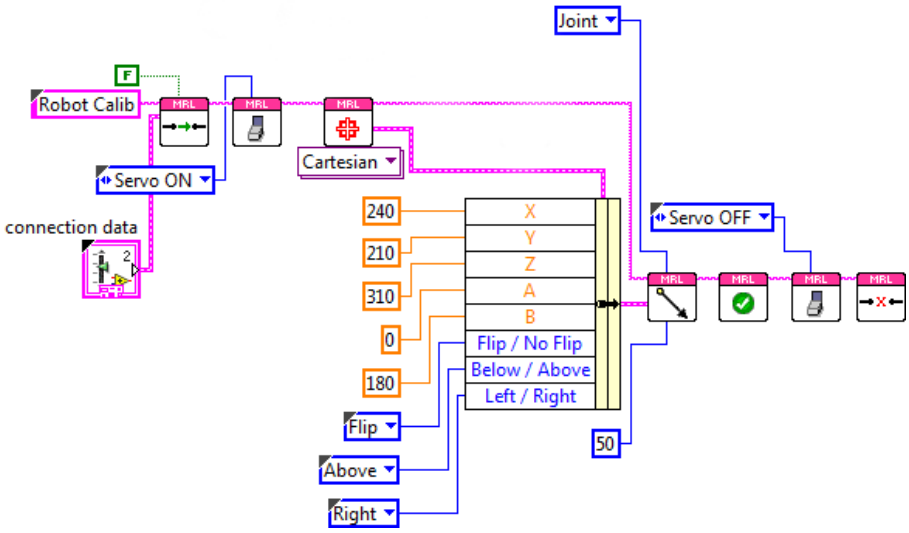
Po vyřešení problému s posunutým souřadným systémem, je zapotřebí propojit počítač z řídicí jednotkou robotu. Řídicí jednotka používá sériovou komunikaci skrze port RS-232. Ten je pomocí převodníku převeden na rozhraní USB a pomocí něj zapojen do počítače.

7.3.1 Zkušební programová část pro řízení robotu

Nejprve je vytvořen program pro navázání spojení s řídicí jednotkou robotu a vypsání nějakých údajů o stavu robotu. To je provedeno za pomoci knihovny *ImagingLab*, která je pospána v kapitole 5.2. Při sestavování programu pro řízení robotu pomocí knihovny *ImagingLab* je nejdříve zapotřebí nastavit a spustit relaci robotu. To je zařízeno pomocí funkce `Open Session`, kde je nutné nastavit data pro připojení. Jedná se o typ řídicí jednotky, typ robotu, IP adresu a port (v případě komunikace přes ethernetovou síť) a příslušný sériový port, na který je řídicí jednotka napojena. Ten je možné nalézt ve správci zařízení v operačním systému počítače. Dále je nutné zvolit jméno relace, kterým se inicializují ostatní použité funkce. Nejprve je zkoušeno vypsání údajů o poloze robotu za pomoci funkce `Get Current Position`. Zde je možné volit z několika možností. Jednou z možností jsou kartézské souřadnice, které vypisují souřadnice x, y a z, úhel rotace koncového efektoru, úhel směru koncového efektoru, zdali se může překlápět, jestli vrchem či spodem a zdali má dojet do koncové pozice zleva či zprava. Ta je také v programu použita. Další možnosti jsou například výpis natočení jednotlivých kloubů robotu, nebo kartézské souřadnice s posledními čtyřmi úhly kloubů atd. Program je ukončen funkcí `Close Session`, která pouze ukončuje relaci robotu. Dříve, než se program spustí, je nutné přepnout řídicí jednotku do režimu *Auto Ext.* a ovladač

do režimu *Disable,* jinak program není schopen komunikovat s řídicí jednotkou. Po spuštění programu se na *front panelu* objeví vypsané souřadnice, které souhlasí jak s reálnými souřadnicemi, tak souřadnicemi na ovladači. Program tedy je schopen komunikace s řídicí jednotkou a může vypisovat požadovaná data.

V dalším kroku je program upraven tak, aby bylo možné vyzkoušet pohyb robotickým ramenem na zadané souřadnice. To je provedeno tak, že po otevření komunikace s řídicí jednotkou pomocí funkce `Open Session`, se použije funkce `Servo ON/OFF,` která zapne servomotory a tím je robot připraven k pohybu. Dále je použita, jako v předchozím případě funkce `Get Current Position,` odkud je převzata struktura kartézských souřadnic. Této struktuře jsou přiřazeny nové souřadnice, které určují, kam se má koncový efektor robotu přemístit. Takto upravená struktura je pak dále použita ve funkci `Move.` Ta se stará o pohyb robotického ramena. Pro její funkčnost je zapotřebí nastavit trajektorii pohybu robotu, souřadnice, na které se má přemístit, jméno relace ve které daný robot pracuje a rychlost jeho pohybu. V tomto případě je trajektorie nastavena na *Joint*, což je způsob pohybu, kde se vypočítají jednotlivé úhly natočení kloubů robotu pro zadané souřadnice. Souřadnice jsou zadány pomocí struktury popisované výše. Jméno relace je předáno z předchozích funkcí a rychlost robotu je nastavena na 50 % výkonu. Aby se program neukončil dříve, než se pohyb vykoná, je zde ještě funkce `Wait Motion` Stop, která čeká, dokud se pohyb nedokončí a poté program pokračuje dále. Nakonec stačí jen vypnout servomotory, aby se aktivovaly brzdy a program ukončit. Program je poté zkoušen s různými souřadnicemi a je kontrolována správnost provedení. Blokové schéma pro pohyb s robotickým ramenem je vidět na obrázku Obr. 50.



Obr. 50 Blokové schéma pro základní pohyb robotu

7.3.2 Návrh a implementace řízení robotu

Při návrhu programové části na přemisťování objektů je vycházeno z pravidel pro pohyb robotu, která je možné najít v dokumentaci k robotu (18). Robot vždy nejdříve najede přibližně 15 cm nad detekovaný objekt, poté k němu pomaleji dojede pohybem v jedné ose kolmé na úchop. Objekt uchopí a ve stejné ose opačným směrem vyjede zase cca. 15 cm nad souřadnice uchopeného

objektu. Odtud jej přesune cca. 15 cm nad souřadnice odložení a poté dojede pomaleji na upouštěcí výšku. Objekt upustí a vyjede na bezpečnou výšku cca. 15 cm, ve které zůstává a čeká na další objekt.

Implementace programové části vychází z předchozí zkušební programové části pro řízení robotu. Ta je rozšířena o ovládání úchopu koncového efektoru, což je řešeno funkcí `Open/Close Hand.` Postupně jsou za sebe kladeny funkce `Move` a `Open/Close Hand`, tak jak je popsáno výše. Za každou jednou funkcí je poté funkce `Wait Motion Stop,` aby se nezačal provádět další příkaz, aniž by byl dokončen předchozí pohyb. Souřadnice detekovaného objektu a místa, kam se má objekt umístit, jsou získány přes vstupy do programové části. Výška, v jaké se má objekt uchopit, je počítána z definované výšky objektu, který se posílá současně se souřadnicemi. Pokud je výška objektu větší než hloubka kleštin koncového efektoru, je uchopovací výška dána odečtem hloubky kleštin od výšky objektu. Blokové schéma je z důvodu velikosti na přiloženém CD v adresáři /obrázky/move_robot_orig.png.

Pro testování je tato programová část spojena s ostatními programovými částmi. Programová část pro detekci objektů poskytuje řídicímu programu robotu souřadnice detekovaných objektů. Tím je možné vyzkoušet, zdali řídicí program funguje. Bohužel je při testování zjištěno několik chyb. První chybou je vzdálenost prostoru zabíraného kamerou. Když se objekt položí na nejvzdálenější místo, které je kamera schopna nasnímat, robot na objekt již nedosáhne. V takové situaci řídicí jednotka robotu zahlásí příslušnou chybu zvukovým signálem a výpisem chybového kódu na displej. To je ihned upraveno posunutím snímaného prostoru blíže k robotu. Při této operaci je nutné znovu provést zaměření souřadnic a zjištění transformačních hodnot. Tím je znovu odzkoušena správná funkčnost této programové části. Další chybou, která je zjištěna, je neustálé snímání pracovního prostoru. To znamená, že jsou řídicímu programu neustále posílány souřadnice detekovaného objektu. Jakmile pro něj ruka dojede, překryje jej a kamera ho nevidí. To není problém v případě jednoho objektu v obraze. Pokud tam jsou ale dva, je detekován ten druhý a nastává chyba. Tento problém je vyřešen pozastavením detekce objektů při vykonávání manipulace s objekty. Pozastavení je vyřešeno kontrolou stavu robotu. Tedy dokud má robot zapnuté servomotory, je detekce pozastavena. Po vyřešení těchto dvou problémů již program funguje, jak má.

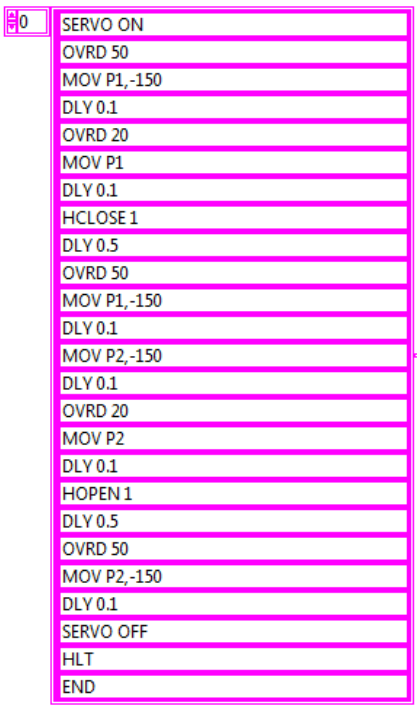
Při testování se však vyskytuje jeden nedostatek. Tím je rychlost robotu. Ten po každém dojetí na příslušné souřadnice dlouho čeká, než provede další pohyb. Nejvíce je to znát při nastavené maximální rychlosti robotu, kdy přesun z jedněch souřadnic na druhé provede plnou rychlostí, ale mezi jednotlivými kroky zase čeká stejnou dobu, jako při menší rychlosti. Oprava nedostatku je nejprve zkoušena pomocí eliminace funkcí `Wait Motion Stop.` Tady je ale naraženo na problém přeskakování kroků v pohybu robotu a, i když se některé odstraněny, k vyšší rychlosti to moc nepomáhá. Dalším pokusem o odstranění je pevné časování. Místo funkcí `Wait Motion Stop` jsou vloženy funkce na pozastavení programu, jak je možné vidět na blokovém schématu na přiloženém CD v adresáři /obrázky/move_robot_wait.png. Robot tedy dostává příkaz, který začne vykonávat, ale program se na určitou dobu pozastaví. Jakmile uvedená doba uplyne, program pokračuje. Postupným snižováním času jednotlivých úseků je zjištěna minimální hranice než robot začne určité kroky přeskakovat. I při zjištění těchto minimálních čekacích dob je vykonávání velmi podobné, jako

v předchozím případě. Problém tedy nastává nejspíš při komunikaci s řídicí jednotkou robotu. Nabízí se tedy ještě jeden způsob, a to nahrání celého sestaveného programu pro konkrétní objekt do řídicí jednotky. Tento způsob je popsán v další kapitole.

7.3.3 Řídicí program za pomoci sestaveného podprogramu v jazyce *Melfa Basic*

Pro vytvoření programové části s využitím jazyku *Melfa Basic,* je nutné nejdřív sestavit v tomto jazyku řídicí program. Jazyk je ale celkem jednoduše pochopitelný a přepsání navrženého řešení do tohoto jazyku netrvá dlouho. Sestavený program je možné vidět na obrázku Obr. 51. Dále je nutné zajistit možnost změny jednotlivých souřadnic. Jednou možností je nahrazování příkazů jednotlivých řádků novými. Další a lepší možností je použití připravených funkcí z knihovny *ImagingLab*.

Aby bylo možné program editovat, je použita funkce `Edit Open.` U této funkce je zapotřebí připojit relaci, která je vstupem do programové části. Dále je potřeba pojmenovat program a nastavit, ve kterém slotu se bude pracovat. Tímto je možné program editovat. Sestavený program v *Melfa Basic* je připraven pomocí funkce `Edit line`, který ji připraví pro nahrání do slotu. Zde se musí definovat slot, relace robotu a hlavním vstupem je pole příkazů. Dále už stačí upravit pozice dané vstupními proměnnými do programové části. To je provedeno pomocí funkce `Edit Position`, kde se nejprve určí relace robotu a slot. Poté musíme definovat, jakou pozici (závisí na pojmenování v sestaveném programu v *Melfa Basic*) chceme editovat a pak, jaké souřadnice se mají nastavit. V tomto případě upravujeme dvě pozice. Jednou je pozice objektu a druhá místo, kam se má umístit. Zbytek je v programu napevno a nemusí se měnit. Po úpravě těchto dvou pozic je možné tedy editaci ukončit pomocí funkce `Edit Close.` Nyní je možné program nahrát do řídící jednotky, což umožňuje funkce `Load Program.` U této funkce se musí definovat jméno programu, relace robotu a slot. V této chvíli je program připraven na spuštění funkcí `Start Program,` kde je zase nutné zadat relaci robotu, jméno programu a slot. Navíc je tu nastavení, zda se má řídicí jednotka před spuštěním resetovat nebo ne. V tomto případě je nastaveno resetování. Po tomto příkazu je program spuštěn a provádí se naprogramované úkony. Za funkcí `Start Program` je ještě funkce `Wait Program End`, která čeká na provedení všech příkazů.



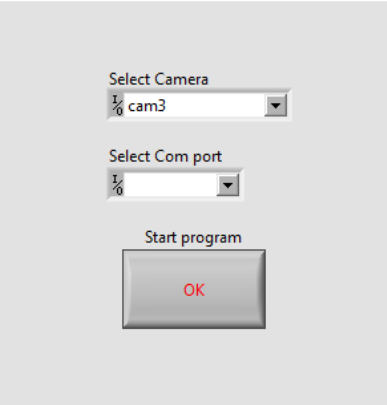
Obr. 51 Sestavený program v jazyku *Melfa Basic*

Při testování takto vytvořeného programu již robot mezi kroky nečeká a pracuje plynuleji. Plynulost je ještě vylepšena změnami hodnot v příkazu *DLY*. Ten zajišťuje čekání mezi kroky robotu. Celé blokové schéma je možné vidět na obrázku na přiloženém CD v adresáři /obrázky/move_robot_program.png

7.4 Hlavní program

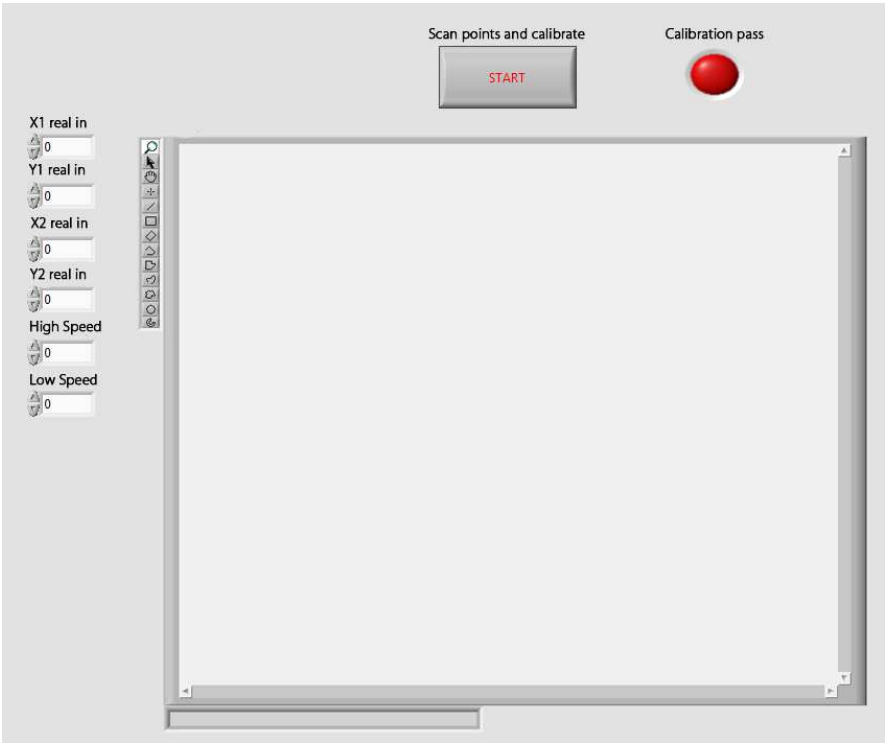
V této kapitole je popsáno, jak program funguje jako celek a slouží tak jako návod, jak program ovládat. Program je možné nalézt na přiloženém CD v adresáři /program/. Hlavní spouštěcí program se nazývá *Grab.vi*.

Při spuštění programu je nejprve výzva k výběru sériových portů kamery a robotu, ke kterým je máme připojeny, viz obrázek Obr. 52. Pokud je ve výběru více zařízení a není možné určit, které je které, je možné to zjistit ve správci zařízení v PC. V případě použití ovladačů z vývojového prostředí *LabVIEW,* jako je v případě kamery, je možné zjistit jméno kamery v programu *Measurement & Automation Explorer*.



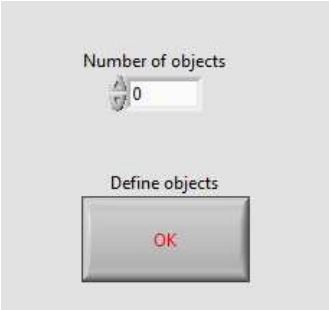
Obr. 52 Úvodní obrazovka pro výběr portů

Po stisknutí tlačítka start se provádí operace, při které robot najíždí do výchozí pozice, ze které bezpečně dojede k objektům a nezasahuje do obrazu. Po této operaci je zobrazena výzva zdali chceme provést konfiguraci, tedy zjištění transformačních hodnot, či načíst předchozí. V případě, že je zvolena konfigurace, je zobrazeno nové okno pro konfiguraci, viz obrázek Obr. 53. Zde je možné vidět několik polí pro vložení souřadnic bodů pro detekci a rychlosti robotu. Dále je zde prvek pro zobrazení obrazu z kamery, indikátor stavu konfigurace a tlačítko pro spuštění rozpoznání, a tedy zjištění transformačních hodnot. V tomto kroku je nejdříve vyzváno k označení detekční značky, která by měla být označena co nejpřesněji. Po potvrzení označení se spustí detekční a výpočetní programová část a výsledek indikuje dioda. V případě chybné detekce svítí červeně v případě správné zeleně a současně se zobrazí výzva k uložení dat.



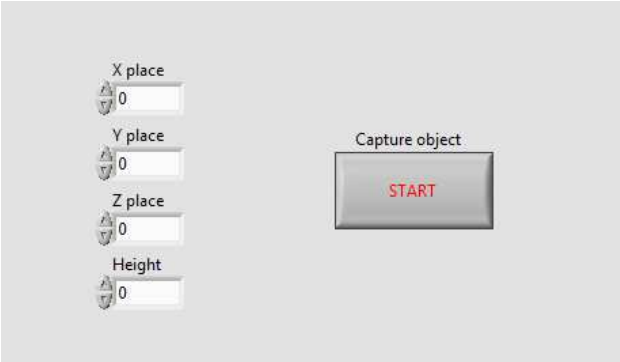
Obr. 53 Okno pro vložení souřadnic detekčních značek a spuštění výpočtu transformačních hodnot

Pokud je zvolena možnost, uložení provede se operace na pozadí a pokračuje se v programu. Dalším krokem je výběr počtu objektů určených k detekci. Okno se zde skládá pouze z pole pro zvolení hodnoty a tlačítka pro potvrzení, viz obrázek Obr. 54.



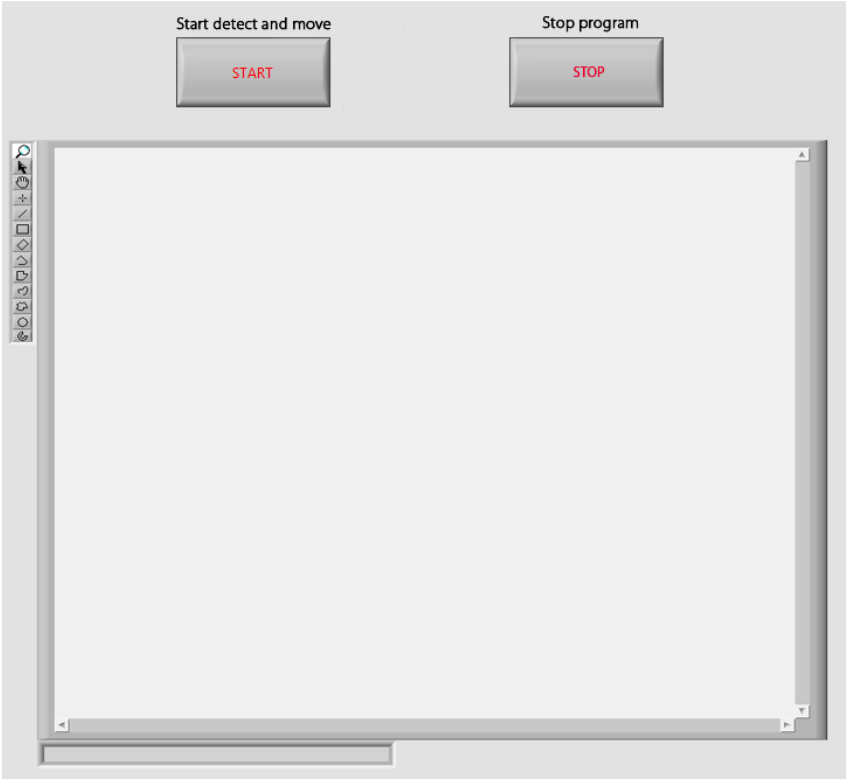
Obr. 54 Okno pro definování počtu objektů určených k detekci

V dalším kroku je otevřeno okno pro definování souřadnic odložení daného objektu a jeho výšky. Okno obsahuje ještě tlačítko pro spuštění programové části na nasnímání objektu. Po zadání hodnot a následného spuštění se tedy otevře okno pro označení objektu, tak jako u detekční značky. Objekt je nutné vybrat přesně, aby byla zaručena vysoká přesnost. Tento proces se zopakuje tolikrát, kolik bylo navoleno počtu objektů. Po potvrzení je tedy buď znovu zobrazeno okno pro definování dat o objektu, nebo se přejde do dalšího kroku. Okno pro definování objektů je na obrázku Obr. 55.



Obr. 55 Okno pro definování dat o objektu

V posledním kroku je zobrazeno okno pro zobrazování obrazu z kamery, tlačítko Start pro spuštění detekce a manipulace a tlačítko Stop pro ukončení programu. Po spuštění je zobrazen obraz a v něm červeně orámovaný detekovaný objekt. Ten se následně přemístí pomocí robotického ramene na určené souřadnice. Tato část pracuje, dokud není ukončena stisknutím tlačítka Stop. Okno je možné vidět na obrázku Obr. 56.

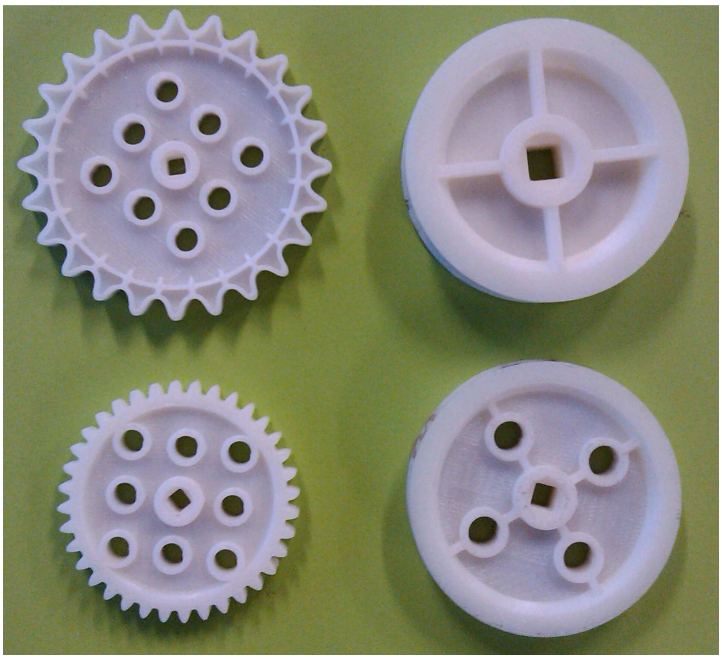


Obr. 56 Okno pro detekci a manipulaci

7.5 Realizace testovacích úloh

Tato podkapitola se zabývá vytvořením testovacích úloh, za pomoci kterých je možné posoudit vhodnost řešení zadané problematiky. Pro testování jsou nejdříve vybrány objekty pro manipulaci. Z důvodu zjednodušení úchopu objektů jsou zvoleny kruhové objekty. Jedná se o čtyři typy ozubených kol, které jsou vytisknuty na 3D tiskárně. Dva z objektů jsou vždy podobné, aby byla testována správnost detekce. Od každého typu je vytisknuto 5 kusů, aby bylo možné zkoušet detekci více stejných kusů najednou. Objekty pro testování je možné vidět na obrázku Obr. 57.

Testy probíhají tak, že při různých natočeních je zkoušena opakovaná konfigurace a následně testováno, jak jsou objekty detekovány. Objekty jsou kladeny v různém pořadí a je kontrolována správnost přemístění a samozřejmě zdali je objekt vůbec přemístěn. Celkově proběhl pětkrát test na jinak natočenou kameru, kde při tomto testu je použito všech dvacet objektů pro detekci.



Obr. 57 Objekty pro testování

V tomto testu, až na čtyři špatně uchopené objekty, kde je chyba způsobena ne úplně rovným podkladem (klíčovací plátno), proběhlo vše v pořádku. To odpovídá úspěšnosti 96%. Robotické rameno je ale určeno k dopravníkovému pásu, kde bude nejspíše zaručen lepší podklad. Při konfiguracích se vyskytl dvakrát problém s nedetekovanými body. Program ale tuto chybu kontroluje a nutí k nové konfiguraci, takže to není bráno jako závažná chyba. Co se týče špatného detekování objektu, nebo špatného přemístění, tak zde není zjištěna při testech žádná chyba.

8 Shrnutí

Tato diplomová práce pojednává o návrhu programu schopného detekovat objekty za pomoci kamery. Tyto objekty jsou následně přemisťovány za pomoci robotického ramene Mitsubishi Melfa RV-2AJ.

Kapitola 2 se zabývá dvěma algoritmy pro detekování objektu v obraze. Jedním je detekce hrany a druhým vyhledání vzoru v obraze. Další komponentou v celém systému je kamera, která je popsána v kapitole 3 společně kartou pro připojení kamer s rozhraním USB 3.0 k počítači. V kapitole 4 je popsáno robotické rameno Mitsubishi RV-2AJ a jeho příslušenství. Společně s ním jsou popsány běžně používané jazyky pro jeho řízení *Melfa Basic* a *Movemaster*. Následující kapitola 5 pojednává o vývojovém prostředí *LabVIEW* a jeho významných funkcích. Zejména se jedná o knihovny IMAQ, IMAQdx a *ImagingLab*. Současně jsou popsány některé softwarové moduly pro práci s knihovnami. Teoretickou část ukončuje metodika použitá při řešení praktické části.

V praktické části je nejdříve v kapitole 7.1 pojednáváno o řešení snímání a zjištění transformačních hodnot pro přepočtení souřadnic ze souřadného systému robotu do souřadného systému kamery. V kapitole 7.2 je řešena detekce objektů v obraze a zjišťování jejich souřadnic. V další kapitole 7.3 je řešeno řízení robotického ramene za pomoci knihovny *ImagingLab*. V předposlední kapitole 7.4 je pak popsán způsob ovládání kompletního programu. V závěru praktické části je nakonec popsán způsob testování programu a jeho výsledky.

9 Diskuze

Výsledkem práce je program vytvořený za pomoci vývojového prostředí *LabVIEW*. Program je schopen detekovat a přemisťovat objekty položené do pracovního prostoru robotu na předem definované souřadnice. Systém je uzpůsoben tak, aby i při nepřesně ustavené kameře mohl bezchybně pracovat. To je dáno částí programu pro zaměření kontrolních značek, díky kterým je schopen vypočítat transformační hodnoty. Ty jsou následně používány pro přepočet souřadnic. Tyto hodnoty je možné i ukládat a později načítat. Práce je použitelná jako návod pro řízení robotického ramene, s možností externího zásahu do programu. Prozatím bylo možné pouze řízení za pomoci specializovaných programů, kde se program předem sestavil a příkazy mohly být posílány jen s jeho pomocí a nebylo možné do něj zasahovat externě, například posílat cílové souřadnice získané jiným programem. Dále může sloužit jako prezentační manipulační úloha pro studenty.

Při návrhu a implementaci programu se samozřejmě vyskytly chyby a problémy. Program byl v průběhu implementace neustále modifikován a vylepšován, aby se dosáhlo co nejspolehlivějšího systému.

Problémem při implementaci byla nepříliš okomentovaná knihovna pro řízení robotického ramene *ImagingLab*. V případě problému s čekáním robotu mezi kroky, se nedaly dohledat možné důvody. Nalezené řešení pomocí předem sestaveného programu za pomoci jazyku *Melfa Basic* může být ještě vylepšeno. Možným vylepšením by mohlo být poslání sestaveného programu do jednotky a měnit pak pouze souřadnice. Zdali by to pomohlo, je jen otázkou. Při dohledávání informací bylo také zjištěno, že brzy bude nová verze knihovny pro řízení robotického ramene, kde bude i detailní dokumentace.

Jedním z dalších možných vylepšení by mohlo být automatická kontrola ustavení kamery. Pomocí dvou přesně zaměřených bodů, které by byly vždy viditelné, by se mohla pokaždé zkontrolovat jejich poloha s uloženými hodnotami. V případě, že by se neshodovali, bylo by provedeno nové zaměření a vypočtení nových transformačních hodnot.

Program sice není komplexním řešení dané problematiky a existuje mnoho vylepšení, která by jej posunula dále, ale splňuje daná kritéria a pracuje s vysokou přesností a spolehlivostí. Program je stavěn tak, aby bylo možné používat jeho jednotlivé části. Je tedy možné tyto části dále používat při řešení jiných manipulačních, či detekčních úloh, nebo celý program dále rozšiřovat, či vylepšovat.

10Závěr

Cílem práce bylo navrhnout a následně implementovat způsob detekce objektů v obraze a řídící program pro robotické rameno, které bude s objekty manipulovat. To vše mělo být implementováno pomocí vývojového prostředí *LabVIEW*, jeho knihoven a modulů.

Cíl práce byl splněn a robotické rameno je schopné přemisťovat detekované objekty na předem dané souřadnice, a to i v případě pootočené kamery. Testy je potvrzena použitelnost řešení, které poskytuje dostatečnou přesnost a opakovatelnost.

11 Literatura

1 FAUST, Russell A. *Robotics in surgery: history, current and future applications*. New York: Nova Science Publishers, c2007, xiv, 317 p. ISBN 9781600213861.

2 VYTEČKA, Marcel. *Šachový automat s využitím manipulátoru Katana* [online]. Brno, 2013, 2015-11-22 [cit. 2015-11-22]. Dostupné z: http://www.acm-spy.cz/wp-content/uploads/2014/05/acmspy2013_submission_11.pdf

3 JOHN C. RUSS. *The image processing handbook*. 6th ed. Boca Raton, FL: CRC Press, 2011. ISBN 978-143-9840-634.

4 HAMMERSCHMIEDT, Michal. *Analýza obrazu pro potřeby řízení mobilního robota* [online]. Brno, 2013, 2015-11-22 [cit. 2015-11-22]. Dostupné z: http://www.acm-spy.cz/wp-content/uploads/2014/05/acmspy2013_submission_61.pdf

5 STRAKA, Stanislav. *Segmentace obrazu* [online]. Brno, 2009, 2015-11-22 [cit. 2015-11-22]. Dostupné z: is.muni.cz/th/72784/fi_m/dp.pdf

6 TRUCCO, Emanuele a Alessandro VERRI. Introductory techniques for 3-D computer vision. Upper Saddle River, NJ: Prentice Hall, c1998, xvii, 343 p. ISBN 01-326-1108-2.

7 MOESLUND, Thomas B. *Introduction to video and image processing: building real systems and applications*. London: Springer, c2012, xi, 227 s. Undergraduate topics in computer science. ISBN 978-1-4471-2503-7-.

8 BRIECHLE, Kai a D. HANEBECK. *Template Matching using Fast Normalized Cross Correlation* [online]. 2015-11-25, : 8 [cit. 2015-11-25]. Dostupné z: http://i81pc23.itec.uni-karlsruhe.de/Publikationen/SPIE01_BriechleHanebeck_CrossCorr.pdf

9 RAJAGOPAL, Ram. *Pattern Matching Based on a Generalized Transform* [online]. 2015-11-25, : 9 [cit. 2015-11-25]. Dostupné z: http://users.ece.utexas.edu/~bevans/courses/ee381k/projects/fall00/rajagopal/litSurveyReport.pdf

10 SZELISKI, Richard. *Computer vision: algorithms and applications*. New York: Springer, c2011, xx, 812 p. Texts in computer science. ISBN 18-488-2934-5.

11 *Basler AG - Industries* [online]. [cit. 2015-11-25]. Dostupné z: http://www.baslerweb.com/en

12 *NI PCIe-8242 - National Instruments* [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://sine.ni.com/nips/cds/view/p/lang/cs/nid/211213

13 GOUBEJ, Martin, Martin ŠVEJDA a Miloš SCHLEGEL. *Úvod od mechatroniky, robotiky a systémů řízení pohybu* [online]. Plzeň, 2012 [cit. 2015-04-20]. Dostupné z: http://home.zcu.cz/~msvejda/URM/materialy/Uvod%20do%20mechatroniky.pdf. Skriptum pro studenty doktorských programů v oboru automatické řízení. Západočeská univerzita v Plzni.

14 *Automa: Nový, menší a výkonnější robot MELFA RV-2AJ* [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://automa.cz/index.php?id_document=28493

15 *MELFA Industrial Robots Specifications Manual: RV-1A/RV-2AJ Series* [online]. [cit. 2015-11-28]. Dostupné z: http://www.rixan.com/Portals/0/RV-1A-2AJ/1n2specs.pdf

16 *Mitsubishi Movemaster robot RV-2AJ* [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://www.roboex.com/nn3.html

17 *April|2011| favnservice* [online]. 2015 [cit. 2015-11-28]. Dostupné z: https://favnservice.wordpress.com/2011/04/

18 *MELFA Industrial Robots Instruction Manual: CR1/CR2/CR3/CR4/CR7/CR8/CR9 Controller* [online]. [cit. 2015-11-28]. Dostupné z: http://www.roboex.com/Mitsubishi%20*Movemaster*%20Robot%20Manual/CR1%20Controller%20Operation.pdf

19 *Robot programming - Mitsubishi - Provendor Ltd.* [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://www.provendor.fi/english/index.php?page=mitsurobotprogram

20 *HW.cz: Začínáme s LabVIEW* [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://vyvoj.hw.cz/teorie-a-praxe/knihovnicka/zaciname-s-*LabVIEW*.html

21 *Průmyslové-kamery.cz: Vše pro strojové vidění* [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://www.prumyslove-kamery.cz/o-produktech/merici-hardware-a-software-national-instruments/software/

22 *National Instruments: Product Manuals* [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://search.ni.com/nisearch/app/main/p/ap/tech/lang/cs/pg/1/sn/catnav:pm/

23 *National Instruments: ImagingLab Robotics Library* [online]. 2015 [cit. 2015-11-28]. Dostupné z: http://www.ni.com/white-paper/12395/en/

24 ImagingLab srl. *ImagingLab Robotics Library for MITSUBISHI* [software]. [přístup 2015-11-28]. Dostupné z: http://www.imaginglab.it/eng/MitsuLib_DataSheet.php.

Přílohy

A Tabulka základních příkazů programovacího jazyka MELFA BASIC s vysvětlením

Instrukce vztažené k řízení pohybu robotu	
Příkaz	**Popis**
"MOV (Move)"	Pohyb za pomoci spojové interpolace
"MVS (Move S)"	Pohyb za pomoci lineární interpolace
"MVR (Move R)"	
"MVR2 (Move R2)"	
"MVR3 (Move R 3)"	
"MVC (Move C)"	
"MVA (Move Arch)"	Pohyb za pomoci obloukové interpolace
"OVRD (Override)"	Specifikování celkové rychlosti
"SPD (Speed)"	Specifikování rychlosti při lineárním nebo kruhovém pohybu
"JOVRD (J Override)"	Specifikování rychlosti při spojovém pohybu
"CNT (Continuous)"	Specifikování průběžné cesty
"ACCEL (Accelerate)"	Specifikace akcelerace/zpomalení
"CMP JNT (Comp Joint)"	Specifikace dodržování kloubového souřadného systému
"CMP POS (Composition Posture)"	Specifikace dodržování souřadného systému XYZ
"CMP TOOL (Composition Tool)"	Specifikace dodržování souřadného systému chapadla
"CMP OFF (Composition OFF)"	Vypnutí dodržování souřadného systému
"CMPG (Composition Gain)"	Specifikování síly dodržování
"OADL (Optimal Acceleration)"	Specifikování míry optimalizace akcelerace
"LOADSET (Load Set)"	Specifikování optimální podmínky pro chapadlo
"PREC (Precision)"	Specifikování módu vysoké přesnosti
"TORQ (Torque)"	Specifikování točivého momentu každé osy
"JRC (Joint Roll Change)"	Povolení vícenásobné rotace hrotu osy
"FINE (Fine)"	Specifikace rozsahu polohování robotu
"SERVO (Servo)"	Zapnutí/vypnutí servomotorů
"SPDOPT (Speed Optimize)"	Optimalizování rychlosti při lineárním pohybu
"WTH (With)"	Přídavné instrukce polohování robotu
"WTHIF (With If)"	Přídavné podmínkové instrukce polohování robotu
Instrukce vztažené k řízení programu	
"REM (Remarks)"	Komentář
"IF...THEN...ELSE...ENDIF (If Then Else)"	Podmíněné větvení
"SELECT CASE (Select Case)"	Povolení vícenásobného větvení
"GOTO (Go To)"	Skok v programu
"GOSUB (RETURN)(Go Subroutine)"	Skok v podprogramu
"RESET ERR (Reset Error)"	Resetování chyb
"CALLP (Call P)"	Volání program
"FPRM (FPRM)"	Volání program s definovanými argumenty
"DLY (Delay)"	Časování
"HLT (Halt)"	Pozastavení program
"END (End)"	Ukončení program
"ON ... GOSUB (ON Go Subroutine)"	Skok v podprogramu podle hodnoty
"ON ... GOTO (On Go To)"	Skok v programu podle hodnoty
"FOR - NEXT (For-next)"	Opakovaní
"WHILE-WEND (While End)"	Opakování podle podmínky
"OPEN (Open)"	Otevření souboru nebo komunikačního kanálu
"PRINT (Print)"	Výstupní data
"INPUT (Input)"	Vstupní data
"CLOSE (Close)"	Zavření souboru nebo komunikačního kanálu

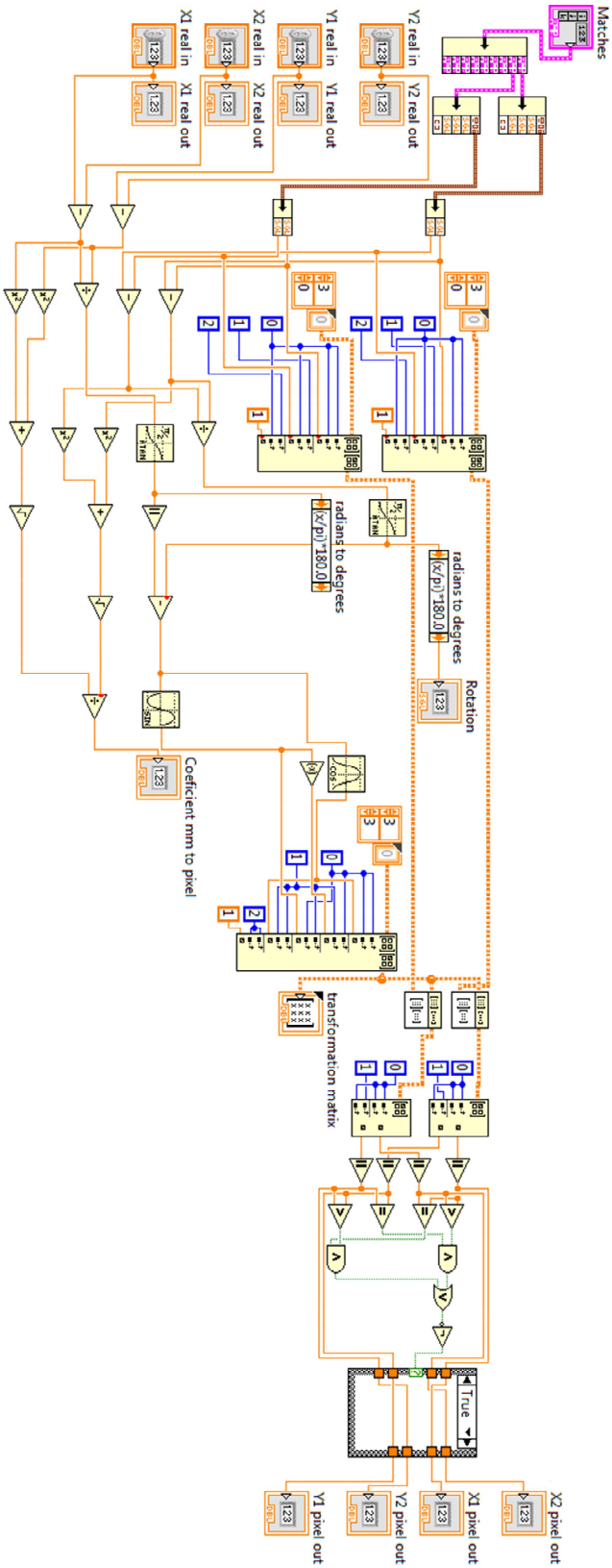
"COLCHK (Col Check)"	Povoluje nebo zakazuje funkci detekci nárazu
"ON COM GOSUB (ON Communication Go Subroutine)"	Přerušení komunikace při skoku v podprogramu
"COM ON/COM OFF/COM STOP (Communication ON/OFF/STOP)"	Povolení/zakázání/ukončení přerušení komunikace
"HOPEN / HCLOSE (Hand Open/Hand Close)"	Otevření/zavření chapadla
"ERROR (error)"	Uživatelské chyby
"SKIP (Skip)"	Přeskočení při pohybu
"WAIT (Wait)"	Čekání na podmínku
"CLR (Clear)"	Vyčištění signálu
Definování instrukcí	
"DIM (Dim)"	Deklarace variabilního pole
"DEF PLT (Define pallet)"	Deklarace palety
"PLT (Pallet)"	Výpočet pozice palety
"DEF ACT (Define act)"	Definování přerušení
"ACT (Act)"	Začátek nebo konec monitorování přerušení
"DEF ARCH (Define arch)"	Definování obloukového tvaru pro obloukový pohyb
"DEF JNT (Define Joint)"	Definování poziční proměnné spojového typu
"DEF POS (Define Position)"	Definování poziční proměnné typu XYZ
"DEF INTE/DEF FLOAT/DEF DOUBLE (Define Integer/Float/Double)"	Definice číselné proměnné typu integer nebo real
"DEF CHAR (Define Character)"	Definice znakové proměnné
"DEF IO (Define IO)"	Definice signálové proměnné
"DEF FN (Define function)"	Definování uživatelské funkce
"TOOL (Tool)"	Nastavení délky koncového efektoru
"BASE (Base)"	Nastavení posunutí základny robotu
Více-úlohové příkazy	
"XLOAD (X Load)"	Načtení program do dalšího úkolového slotu
"XRUN (X Run)"	Spuštění programu v dalším úkolovém slotu
"XSTP (X Stop)"	Zastavení programu v dalším úkolovém slotu
"XRST (X Reset)"	Resetování programu v dalším úkolovém slotu
"XCLR (X Clear)"	Zrušení načítání programu v dalším úkolovém slotu
"GETM (Get Mechanism)"	Získání mechanické řízení robotu
"RELM (Release Mechanism)"	Povolení mechanické řízení robotu
"PRIORITY (Priority)"	Změna priority úkolového slotu
"RESET ERR (Reset Error)"	Resetování chyb
Ostatní příkazy	
"CHRSRCH (Character search)"	Vyhledání textového řetězce ve znakovém poli
GET POS (Get Position)	Rezervováno.

B Příkazy *Movemaster* s vysvětlením

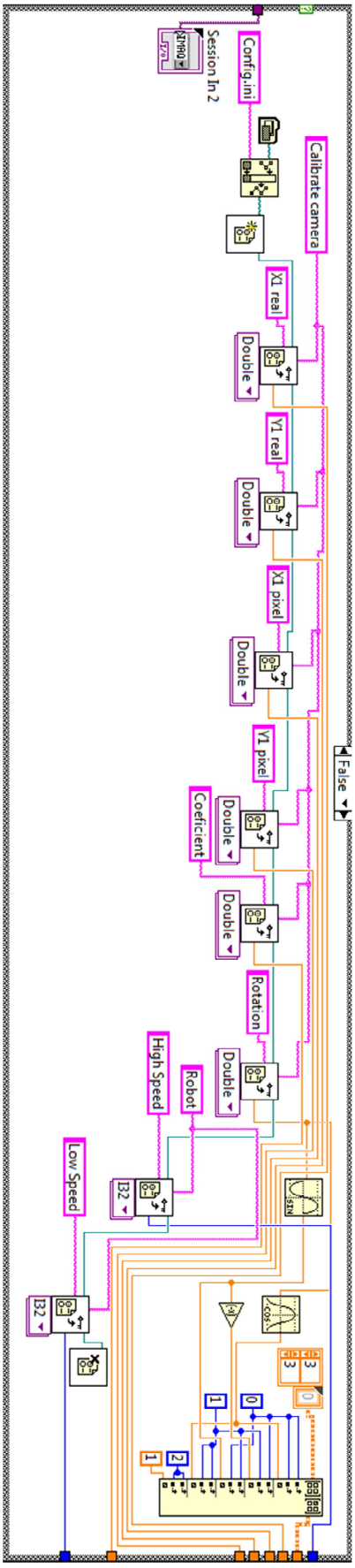
Příkaz	Popis
Příkazy pro řízení pohybu do určité pozice	
CFa[,[R/L][,[A/B]]]	Změna postavení robotu do pozice
DP	Pohyb do předchozí pozice podle čísla pozic
DS [x],[y],[z]	Pohyb o zadanou vzdálenost v osách X, Y a Z, z aktuální pozice
DW [x],[y],[z]	Pohyb o zadanou vzdálenost v osách X, Y a Z, z aktuální pozice
HE a	Zapamatování aktuální pozice pod číslem „a“
HO [a]	Zapamatování aktuální pozice jako počátek
IP	Pohyb do následující pozice podle čísla pozic
MAa1,a2[,[o/C]]	Pohyb do pozice z dané pozice „a1“ o vzdálenost danou pozicí „a2“
MCa1,a2[,[o/C]]	Pohyb z dané pozice „a1“ do dané pozice „a2“ za pomoci lineární interpolace
MJ [J1],[J2],[J3], [J5],[J6][J7],[J8]	Otočení klouby o daný úhel ve stupních z aktuální pozice
MO a [,[o/C]]	Pohyb do dané pozice „a1“
MP [X],[[Y][,[Z] [,[A][,[B][„[L1] [,[L2]]]]]]] [,[<R/L>][,[<A/B>]]]	Pohyb do dané pozice specifikované parametry (X, Y, Z, A, B)
MRa1,a2,a3 [,[o/C]]	Pohyb po kružnici dané třemi body „a1“, „a2“, „a3“ za pomoci kruhové interpolace
MRA a [,[o/C]]	Pohyb po kružnici kde je aktuální i koncový bod „a“ daný kruhovou interpolací
MS a [,[o/C]]	Pohyb na pozici „a“ za pomoci lineární interpolace
MT a,[b][,[o/C]]	Pohyb na pozici z pozice „a“ o vzdálenost „d“ ve směru koncového efektoru za pomoci kloubové interpolace
MTS a,[b][,[o/C]]	Pohyb na pozici z pozice „a“ o vzdálenost „d“ ve směru koncového efektoru za pomoci lineární interpolace
NT	Pohyb do uživatelem specifikovaného počátku
OG	Pohyb do uživatelem specifikovaného počátku Kloubová interpolace
OVRa	Nastavení rychlosti ramene
PC a1 [,a2]]	Vymaže data pozic mezi pozicemi „a1“ a „a2“
PD a,[X],[[Y][,[Z] [,[A][,[B][„[L1] [,[L2]]]]]][,[<R/L>] [,[<A/B>]]][,<o/C>]	Nahradí pozici „a“ pozicí zadanou parametry
PWa	Čekání v aktuální pozici na pulz „a“
SD a [,b[,c,d[,e]]]	Definuje rychlost „a“, časovou konstantu „b“, rozsah akcelerace „c“, rozsah zpomalení „d“ a nastavení CNT „e“
SP a [,[H/L] [,b]]	Definuje rychlost „a“, akceleraci/zpomalení, nastavení CONT „b“
Tla	Časování pro zastavení pohybu
TL[a]	Nastavení délky koncového efektoru
Příkazy pro řízení programu	
CLa	Načtení hodnoty z registru do určeného počítadla „a“
CPa	Načtení hodnoty/řetězce z počítadla „a“ do interního registru
DA a	Deaktivování ukončení programu bitem „a“ externího vstupního signálu
DL [a1][,[a2] [,[b1][,[b2]]]]	Vymazání řádků programu od „a1“ do „a2“nebo kroků od „b1“ do „b2“
EA [+/-] a,b[,[c]]	Aktivuje ukončení bitem „a“ z externího signálu, specifikuje řádek programu „b“, kde je určené co se stane při výskytu ukončovacího signálu, specifikování skoku
ED	Ukončení programu
EQ a, b	Způsobuje skok na řádek „c“ pokud hodnota v interním registru je stejná jako hodnota „a“
GS [a][,[b]]	Zavolání subprogramu začínajícího na řádku číslo „a“ programu „b“
GTa	Způsobí skok na řádek číslo „a“
HLT	Zastavení programu
LG a ,c	Způsobuje skok na řádek „c“ pokud hodnota v interním registru je větší jako hodnota „a“
NEa.c	Způsobuje skok na řádek „c“ pokud hodnota v interním registru se nerovná hodnotě „a“

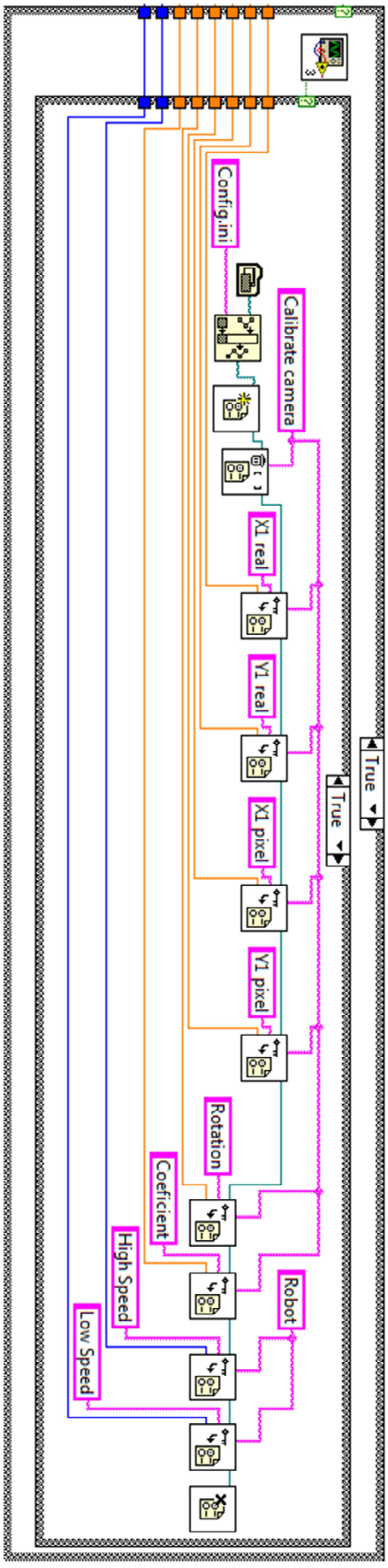
NW	Vymaže všechny řádky a pozice zvoleného programu
NX	Určuje rozsah cyklus programu spuštěnému příkazem RC
RCa	Opakuje cyklus určený příkazem NX, „a“ krát
RN [a1][,a2[,b]]]	Spuštění programu „b“ od řádku „a1“ do řádku „a2“. (a2 neobsahuje)
RT[a]	Dokončí podprogram spuštěný příkazem GS a vrátí se k hlavnímu programu
SM a, c	Způsobuje skok na řádek „c“ pokud hodnota v interním registru je menší jako hodnota „a“
Příkazy pro ovládání chapadla	
GC[a]	Zavírá chapadlo
GFa	Definuje otevření/zavření chapadla používané s příkazem PD
GO [a]	Otevření chapadla
GP a1, a2, a3	Nastavení uchopovací síly „a1“ v čase „a3“ a uchopovací síly při rozpěru „a2“
Příkazy pro	
ID [a]	Přijmutí externího signálu bez podmínky
OB [+/-] a	Nastavení výstupního stavového bitu
OC a [,[a1][,[a2]]]	Výpis hodnoty počítadla „a“ na externí výstupní signál bez podmínky
OD a (or &b) [,[a1][,[a2]]]	Výpis dat „a“ na externí výstupní signál bez podmínky („a1“ startovní bit, „a2“ bitová délka)
TB [+/-] a,b	Způsobuje skok na řádek číslo „b“ na základě stavového bitu „a“ z interního registru
TBD [+/-] a,b	Způsobuje skok na řádek číslo „b“ na základě stavového bitu „a“ z externího registru
Operace, substituce, výměnné příkazy	
ADD a	Sečtení hodnoty „a“ a hodnoty v interním registru
SUB a	Odečtení hodnoty „a“ a hodnoty v interním registru
MULa	Vynásobení hodnoty „a“ a hodnoty v interním registru
DIV a	Vydělení hodnoty v interním registru a hodnoty „a“
ICa	Hodnota v počítadle „a“ se zvětší o 1
DC a	Hodnota v počítadle „a“ se zmenší o 1
AN a	Logická operace AND mezi hodnotou v interním registru a hodnotou „a“
OR a	Logická operace OR mezi hodnotou v interním registru a hodnotou „a“
XO a	Logická operace XOR mezi hodnotou v interním registru a hodnotou „a“
SC a,[b]	Hodnota nebo řetězec „b“ se přiřadí počitadlu „a“
PLa1,a2	Souřadnice pozice „a2“ je přehozena na pozici „a1“
SFa1,a2	Souřadnice pozice „a1“ jsou posunuty o souřadnice pozice „a2“
PXa1,a2	Souřadnice pozice „a1“ a souřadnice pozice „a2“ jsou prohozeny
RS-232C komunikační příkazy	
CRa	Přečte obsah počítadla „a“
DR [a]	Přečte hodnotu z interního registru, stav koncového efektoru, a stav hlavního výstupu. „a“ číslo výstupního bitu
ER[a]	Přečte chybový kód. „a“ není určeno Aktuální chybový kód „a“ Přečte historii chybových kódů
LR[a]	Přečte řádek „a“ aktuálního programu
PR [a]	Přečte všechny souřadnicové hodnoty pozice „a“
QN [a]	Přečte informace o programu číslo „a“ nebo vybraného programu
STR [a]	Přečte program krokového čísla „a“
VR	Přečte jméno verze ROM
WH	Přečte veškeré souřadnicové hodnoty aktuální pozice
WT	Přečte aktuální délku koncového efektoru
WTM	Vypíše aktuální souřadnicový systém koncového efektoru
Ostatní příkazy	
INP a,[b],[,[c]]	Přečte hodnotu počítadla „b“, nebo pozice „b“, řetězce „b“ z kanálu „a“
Na	Vybere program „a“
OPN a,b	Otevře vstup/výstup „b“ pro kanál „a“
PRN a,b or c	Odešle obsah počitadla „a“, nebo pozice „b“, nebo řetězec „c“ z počítače ve spojení s příkazem INP
RS [a]	Resetování stavu určeného „a“

C Blokové schéma pro výpočet transformačních hodnot



D Blokové schéma pro uložení a načtení transformačních hodnot





E Obsah přiloženého CD

Přiložené CD obsahuje:

- Rozměrné obrázky blokových schémat
- Zdrojové kódy programu pro vývojové prostředí LabVIEW
- Video s ukázkou běhu programu