UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

AUTOMATICKÉ ROZTŘIĎOVÁNÍ PŘEDMĚTŮ ROBOTEM ABB YUMI

Bc. Dušan Vašek

Diplomová práce

2022

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2021/2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Dušan Vašek**
Osobní číslo:	**I20198**
Studijní program:	**N0714A150005 Automatické řízení**
Téma práce:	**Automatické roztřiďování předmětů robotem ABB YuMi**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování

Bude vytvořena aplikace s robotem ABB YuMi využívající modul strojového vidění pro lokalizaci uchopovaných předmětů. Robot bude odebírat náhodně umístěné předměty ze skupiny předmětů více typů a umísťovat je na pozice vyhrazené každému typu zvlášť, s využitím obou paží. Pro monitorování činnosti robota bude vytvořeno jednoduché uživatelské rozhraní, nejlépe s využitím externího počítače PC, které umožní upravit parametry provádění, sledovat odebrané počty kusů a ošetřit chybové situace. Práce bude obsahovat rozbor jednotlivých kroků řešení, v přiměřeném rozsahu bude popsáno použité zařízení a využité programovací prostředky a softwarové nástroje. Bude popsána struktura vytvořených programů a komentovány jednotlivé procedury. Funkčnost programu bude v textu demonstrována vhodnou formou.

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

Seznam doporučené literatury:

SICILIANO, B., KHATIB, O., ed. *Springer Handbook of Robotics*. Springer-Verlag, 2008.
CVEJN, J. Návod ke zpracování praktické úlohy ke zkoušce z předmětu Modelování, plánování pohybu a řízení robotů. Univerzita Pardubice, FEI, 2019.
Návod k použití IRC5 s jednotkou FlexPendant, RobotWare 6.06. ABB, 2017.
Technical reference manual RAPID overview, RobotWare 6.06. ABB, 2004-2017.
Technical reference manual RAPID Instructions, Functions and Data types, RobotWare 6.06. ABB, 2004-2017.
Product manual IRB 14000 gripper, IRC5. ABB, 2015-2017.
Application manual Integrated Vision, RobotWare 6.06. ABB, 2013-2017.

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

Datum zadání diplomové práce: **8. listopadu 2021**
Termín odevzdání diplomové práce: **20. května 2022**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Daniel Honc, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 16. listopadu 2021

Prohlášení

Prohlašuji:

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

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

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

V Pardubicích dne

Bc. Dušan Vašek

Poděkování

Tímto bych chtěl poděkovat vedoucímu mé práce panu doc. Ing. Janu Cvejnovi, Ph.D. za odborné konzultace, jeho vedení, trpělivost a ochotu, kterou mi v průběhu zpracování diplomové práce věnoval. Dále bych rád poděkoval své rodině a přátelům, kteří mě podporovali při psaní této diplomové práce.

V Pardubicích dne

Bc. Dušan Vašek

ANOTACE

Cílem této diplomové práce je vytvoření aplikace s robotem ABB YuMi využívající modul strojového vidění pro lokalizaci náhodně umístěných předmětů. Robot bude dále odebírat lokalizované předměty a umísťovat je podle typu na určené místo. Navíc je vytvořeno uživatelské rozhraní v podobě PC aplikace, která umožnuje činnost robota sledovat a řídit.

KLÍČOVÁ SLOVA

RobotStudio, Visual Studio, ABB, YuMi, Strojové vidění.

TITLE

AUTOMATIC ASSORTING OBJECTS BY THE ROBOT ABB YUMI

ANNOTATION

A goal of this diploma thesis is to create an application with the ABB YuMi robot, which uses a machine vision module. This module is used for locating irregularly placed objects. Further, the robot tak the objects away and place them according to their type. In addition, a user interface in the form a PC application is created, which enables to monitor the robot operation.

KEYWORDS

RobotStudio, Visual Studio, ABB, YuMi, Machine vision.

OBSAH

SEZNAM ZKRATEK A ZNAČEK

PC	osobní počítač
WPF	Windows Presentation Foundation
CAPI	Controller API
GUI	Grafické uživatelské rozhraní
HMI	Human machine interface

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

ÚVOD

Průmyslové roboty jsou již dlouhou dobu zastoupeny v různých odvětvích průmyslu a často se objevují na propagačních firemních videích jako znak velkého automatizačního pokroku v průmyslové výrobě. Dosud se jednalo hlavně o roboty konstruované na náročnou, únavnou a nebezpečnou činnost, které měly svůj vyčleněný pracovní prostor, do kterého člověk nesměl. Postupem času se otevíraly dveře spolupráce mezi člověkem a robotem, a za určitých podmínek sdíleli svůj pracovní prostor. V posledních několika letech na trh masivně proniká víceméně nový druh robotů, takzvané kolaborativní roboty. Tyto roboty jsou určeny pro přímou spolupráci mezi člověkem a robotem.

Tato diplomová práce se zabývá vytvořením aplikace strojového vidění s použitím kolaborativního robota ABB YuMi. Robot bude odebírat náhodně umístěné předměty z podložky a následně je bude třídit a přesouvat na určená místa. Pro monitorování činnosti robota bude vytvořeno uživatelské rozhraní.

1 KOLABORATIVNÍ PRŮMYSLOVÉ ROBOTY

Pojetí kolaborativních robotů vzniklo již v roce 1995 v rámci výzkumného projektu společnosti General Motor s myšlenkou vytvořit pro lidi bezpečné roboty, s kterými by mohli spolupracovat na jednom místě. Po dvaceti letech se kolaborativní roboty, také nazývané „coboty“, začaly v průmyslu silně rozvíjet. Dostaly se také do povědomí široké veřejnosti.

Klasické průmyslové roboty a pracovníci byli obvykle odděleni, přičemž roboty prováděly úlohy jako je svařování, lisování plastů, manipulace s břemeny a lakováním. Pracují ve vysokých rychlostech, mnohdy s těžkými nebo ostrými předměty. Umísťují se do ohrazených pracovních ploch, kam mají přístup jen proškolení pracovníci. Vstup je navíc zabezpečen snímači, optickými bránami a dalšími bezpečnostními prvky, aby se při neoprávněném vstupu robot uvedl do klidu (Kolíbal, 2016).

Odnoží průmyslových robotů jsou kolaborativní roboty. Lidé mají výborné předpoklady pro řešení úloh, které se nedají přesně definovat. Roboty na druhé straně mají sílu, přesnost, a možnost dlouhé doby provozu. Hlavním cílem kolaborativních robotů je skloubit tyto významné vlastnosti a umožnit pracovat robotům a lidem na společném pracovišti.

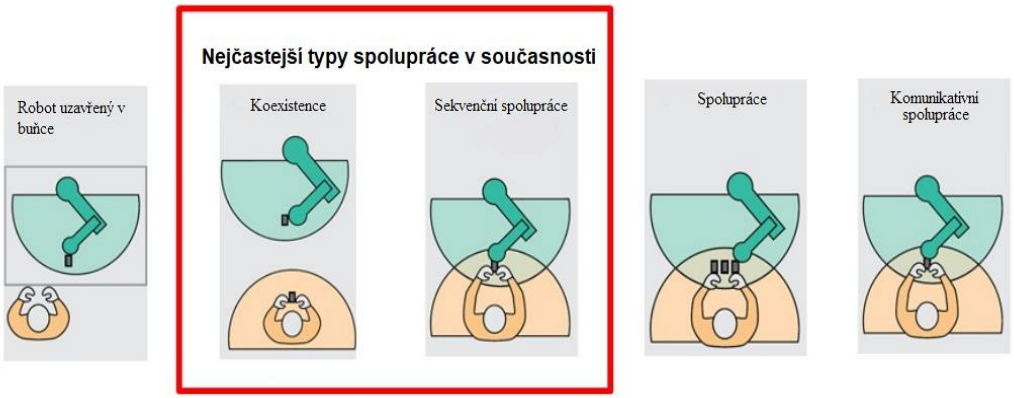
Kolaborativní roboty se nasazují v mnoha odvětvích, ať již se jedná o automobilový průmysl, výrobu elektrotechnických dílů, obrábění kovů nebo tváření plastů. Jsou používány na úlohy jako „Pick and place“, kontrolu kvality, třídění, montáž, manipulaci, balení. Na rozdíl od klasických robotů jsou navrženy tak, aby programování bylo co nejjednodušší. Jsou velmi flexibilní, dají se snadno přemístit na nové pracoviště (Demystifying Collaborative Industrial Robots, 2020).

Výrobou kolaborativních robotů se zabývá celá řada společností. Mezi nejznámější patří společnosti KUKA Robotics, Stäubli, Fanuc, Universal robots a ABB.

1.1 ROBOTIZOVANÁ PRACOVIŠTĚ

Se začleňováním kooperace robotů s člověkem do výroby přišel nový trend v robotice – čím dál tím větší spolupráce člověka s robotem. S tímto trendem se postupně začalo odlišovat, jak takové pracoviště vypadá. Na začátku byly tradiční průmyslové roboty ohraničené klecemi, ke kterým šlo vstoupit pouze při úplném vypnutí robotu. V dnešní době již pracují kolaborativní roboty, které jsou určeny výslovně pro interakci s lidskou obsluhou. Všechny kategorie spolupráce zobrazené na obrázku 1.1 s výhodami a nevýhodami jsou (Demystifying Collaborative Industrial Robots, 2020):

- Robot uzavřený v buňce (Cell) – robot je oddělen od okolního prostředí. Jeho výhodou je bariéra mezi robotem a obsluhou, která zvyšuje bezpečnost a umožnuje robotu pracovat na plný výkon. Nevýhodou je, že robot zabírá i s bariérou více místa a nemůže být flexibilně přesouván, dále je zapotřebí nějakého dalšího procesu mezi spoluprací člověka a robota.
- Koexistence (Coexistence) – robot nemá klec, nemá sdílené pracovní prostředí s obsluhou, využívá bezpečnostní senzory a kamery. Má podobné výhody a nevýhody jako robot uzavřený v buňce. Oproti němu je však snadnější přístup a údržba, ale také vzniká problém s vyhodnocením nebezpečného přiblížení obsluhy.
- Sekvenční spolupráce (Sequential collaboration) – robot i obsluha mají sdílená pracovní prostředí, spolupracují podle stanovených postupů a využívají sdílený prostor v různých časech. Výhody částečné spolupráce ve sdíleném prostředí ale omezují vyšší nároky na bezpečnost.
- Spolupráce (Cooperation) – robot a obsluha pracuje současně na sdílené ploše. Robot musí splňovat různé podmínky bezpečnosti, jako je monitorování síly. Tyto roboty zpravidla nemají takový výkon jako klasické průmyslové roboty a jsou pomalejší, zato však bezpečnější.
- Komunikativní spolupráce (Responsive collaboration) – robot s člověkem pracuje ve sdíleném prostředí na jednom výrobku. Robot interaguje s člověkem v reálném čase. Vznikají minimální doby zastavení, čímž se zvyšuje efektivita (Demystifying Collaborative Industrial Robots, 2020).



Obrázek 1.1 – Typy spolupráce člověk – robot (Demystifying Collaborative Industrial Robots, 2020)

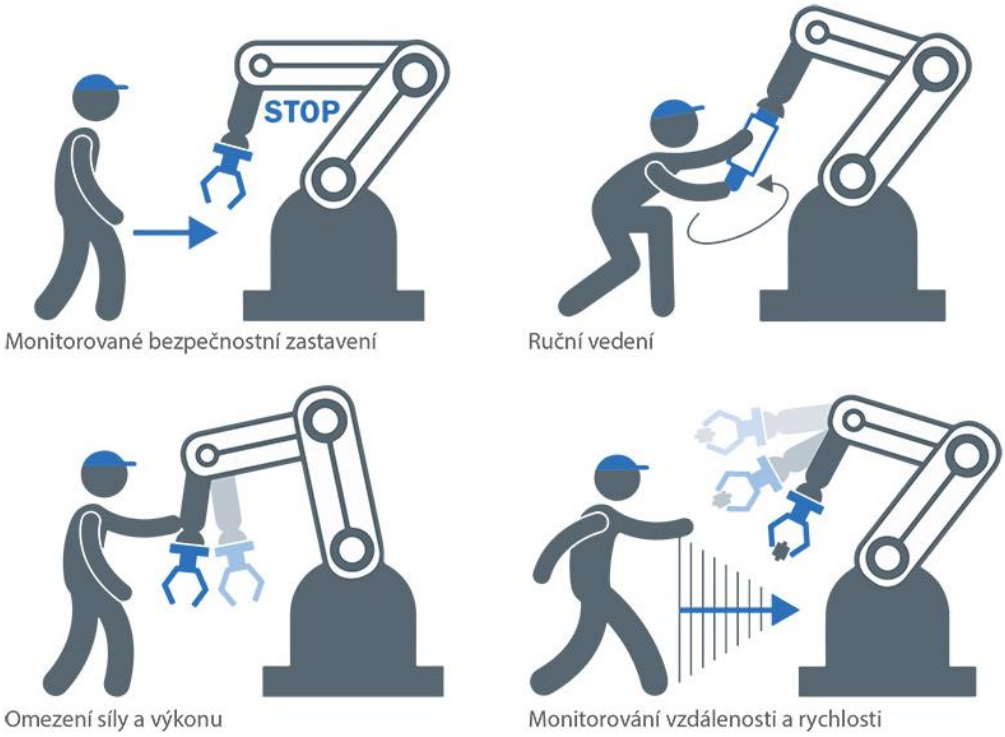
1.2 BEZPEČNOST A NORMY

Kolaborativní roboty jsou z definice všechny průmyslové roboty, které splňují nároky na bezpečnost. Tato bezpečnost je definovaná v mezinárodních normách ISO, kde některé z nich se promítají i do ČSN. Tyto normy lze označit do tří kategorií:

- Norma typu A – stanovuje základní pojmy a zásady pro projektování a konstrukci a obecná hlediska, která mohou být použita u všech strojů. Například norma (ČSN EN 12100, 2011).
- Normy typu B – zabývají se jedním bezpečnostním aspektem, jako jsou bezpečnostní vzdálenosti, rychlost a podobně. Například norma (ČSN EN 13857, 2021). Pro koexistující roboty.
- Norma typu C – řeší požadavky pro jednotlivé stroje nebo skupinu strojů například norma (ČSN EN ISO 10218, 2011). Pro kolaborativní roboty (Kolíbal, 2016).

Norma ČSN EN ISO 10218 Roboty a robotická zařízení rozděluje roboty podle čtyř definic zobrazeny na obrázku 1.2, a to:

- Bezpečnostní monitorované zastavení – roboty tohoto typu využívají řadu senzorů, které zastaví robota, když člověk vstoupí do dráhy robota. Využívá se pro aplikace, kde mají roboty minimální interakci s člověkem.
- Monitorování rychlosti a polohy – tento typ používá k zamezení kolizí a zvýšení bezpečnosti pokročilejší systém kamerových systémů.
- Omezení síly a příkonu vlastní konstrukcí nebo ovládáním – roboty tohoto typu jsou postaveny se zaoblenými rohy a řadou inteligentních kolizních senzorů, které rychle detekují kontakt s člověkem a zastaví robota. Robot má také omezení síly, aby se předešlo jakémukoliv zranění.
- Ruční vedení – obsluha přímo řídí robot v automatickém režimu. Pomocí ručního vedení může obsluha za pomoci robota přesouvat těžké či rozměrné věci, čímž se vyhne namáhavé a opotřebovávací činnosti (ČSN EN ISO 10218, 2011).



Obrázek 1.2 – Režimy spolupráce (Bezpečná robotika, 2018)

K těmto čtyřem základním rozdělením je přiřazena ještě jedna ochrana:

- Ochrana singularity – trajektorie definované v kartézské soustavě souřadnic se mohou v některém bodě přiblížit k bodu singularity. Toto přiblížení může zapříčinit nečekané zvýšení rychlosti os robota, a tedy i vystavit člověka riziku střetu.

Bezpečný robot sám o sobě v praxi ještě nezaručuje bezpečnou spolupráci s obsluhou. Proces, při němž robot, který splňuje bezpečnostní normy, drží ostrý nástroj, není bezpečný, ať již robot pracuje jakkoli pomalu. Při každé aplikaci musí být vypracované zhodnocení rizik, aby bylo jisté, že v žádném případě nehrozí porušení bezpečnosti. Platí to nejen pro robot, ale také pro předměty, se kterými manipuluje, okolní prostředí a tak dále. Posuzuje se:

- Bezpečnost kontrakce – eliminace nebezpečí skřípnutí nebo nebezpečného dotyku.
- Bezpečnostní prostředky – kryty, klece, například pro manipulaci s nebezpečnými nástroji. Nouzové zastavení, bezpečné odpojení momentu.
- Opatření na straně obsluhy – školení bezpečnosti práce, vhodný pracovní úbor (Kolíbal, 2016).

1.3 PROGRAMOVÁNÍ ROBOTŮ

Tradiční průmyslové roboty jsou mnohdy programovány v nízko-úrovňových jazycích, což je složitý a zdlouhavý proces, u něhož jsou potřeba určité odbornosti, na rozdíl od kolaborativních robotů, kde se očekává jednoduchost a flexibilita. Programování robotů je velmi různorodá záležitost. Vesměs každý výrobce robotů vytvořil pro své roboty vlastní programovací jazyky a své vývojové prostředí. Například společnost ABB vyvinula jazyk RAPID, společnost KUKA má Kuka Robot Language, roboty Stäubli používají VAL3. Všechny tyto jazyky umožnují dva základní přístupy pro programování, a to online a off-line programování.

Online programování se provádí přímo na pracovišti, kdy programátor přímo interaguje s robotem, který programuje. K tomu může využít funkce a rozhraní robota, jako jsou teachpendant nebo možnost ručního navádění robota. Obě funkce slouží k tomu, aby programátor uložil body nebo trajektorie pohybu robota a nemusel je vypočítávat. Výhoda této online metody je v tom, že programátor pouze navede robota do požadovaných míst, kde uloží polohu a po přidání dalších instrukcí se plnohodnotný program může na místě odzkoušet a ihned zařadit do provozu. Nevýhodou je, že po celou dobu práce programátora je robot odstaven z provozu.

Off-line programování se provádí ve vytvořeném virtuálním 3D modelu robotizovaného pracoviště. Programátor připraví program, který optimalizuje a zkouší na modelu. Po jeho dokončení nahraje program do reálného robota. Tímto přístupem neblokuje na zbytečnou dobu robota, který mezitím může dále pracovat. K této metodě programování je zapotřebí vhodný software, který modelování umožnuje. Opět různé společnosti vyvíjejí své systémy podle svých potřeb. Zpravidla platí, že každý robot je kompatibilní se softwarem vlastního výrobce. Naleznou se ale i výjimky (Kolíbal, 2016).

Simulační software čtyř největších společností v robotice jsou následující:

- ABB – RobotStudio.
- KUKA – KUKA.Sim.
- Yaskawa – MotoSim.
- Fanuc –RoboGuide.

1.4 ROBOTSTUDIO

RobotStudio je simulační a programovací software od společnosti ABB. Umožňuje programovat roboty bez přerušení výrobního systému v off-line i online režimu. Poskytuje nástroje k navrhování stanic, ovládání robota, simulaci provozu a programování. Využívá vysoko-úrovňový jazyk RAPID určeného k programování industriálních robotů společnosti ABB. Ukrývá v sobě mnoho technologií a možností, jako jsou:

- 3D import a modelování – umožňuje import dat do RobotStudia z formátu CAD.
- Smart components – k vytvořeným komponentám lze přiřadit různé vlastnosti chování.
- AutoPath a AutoConfiguration – umožňuje automatické generování trajektorií robota v závislosti na geometrii 3D modelu.
- Path tools – sada nástrojů pro optimalizaci trajektorie nástroje.
- Editace a ladění programu – zvýraznění chyb a nápověda v RAPID editoru.
- Jobs – dokáže komunikovat mezi kontroléry a provádět zálohy dat, synchronizace stanic a přeposílat data.
- Visual SafeMove – je nástroj pro vytváření bezpečnostních zón, parametrů a podmínek.
- Vizualizace dráhy TrueMove – vykresluje trajektorii, po které se bude robot pohybovat.
- Signal Analyzer – zobrazuje a zaznamenává signály z řídicího systému.
- Visualization FlexPendant – umožňuje zvirtualizovat jednotku FlexPendantu.
- ScreenMaker – tento nástroj umožňuje vytvářet uživatelské aplikace na jednotce FlexPendantu.
- Station Viewer – vytváření záznamů simulace.
- Transfer – přímý přenos programu mezi prostředím RobotStudia a reálným kontrolérem robota.
- Robot Vision – nástroj pro programování kamer strojového vidění, který umožňuje simulaci a práci off-line.

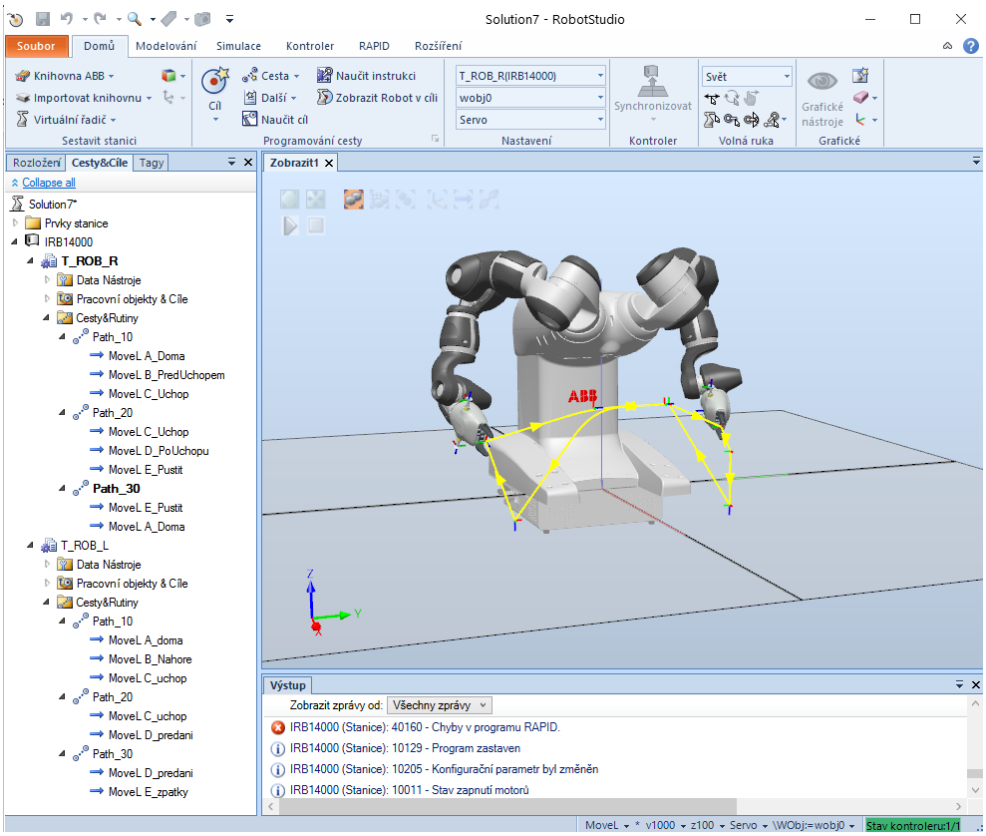
Jedná se o prostředí, které je jednoduché, přehledné a snadno pochopitelné. A s přidanými funkcemi je velmi silným nástrojem pro programování robotů. RobotStudio pracuje paralelně s virtuálním kontrolérem. Obě jednotky spolupracují, fungují ovšem nezávisle na sobě. Na tomto virtuálním kontroléru je RobotStudio založeno. Je to přesná kopie opravdového software, který řídí skutečné roboty. Umožňuje proto velmi realistické simulace.

Programy vytvářené v tomto prostředí je možné dělat v jedné nebo druhé jednotce, nebo také část modifikovat v RobotStudiu a jinou část ve virtuálním kontroléru. Při slučování částí do jednoho funkčního celku je na místě obezřetnost. Může se stát, že jedna část může přepsat

část druhou. Toto slučování neprobíhá automaticky, synchronizace proběhne ručně. Při spuštění synchronizace je umožněno uživateli si vybrat, jaké části chce importovat a jaké ne. Synchronizace funguje v obou směrech, jak z RobotStudia do virtuálního kontroléru, tak naopak.

Program je v RobotStudiu tvořen jako souhrn informací tvořený:

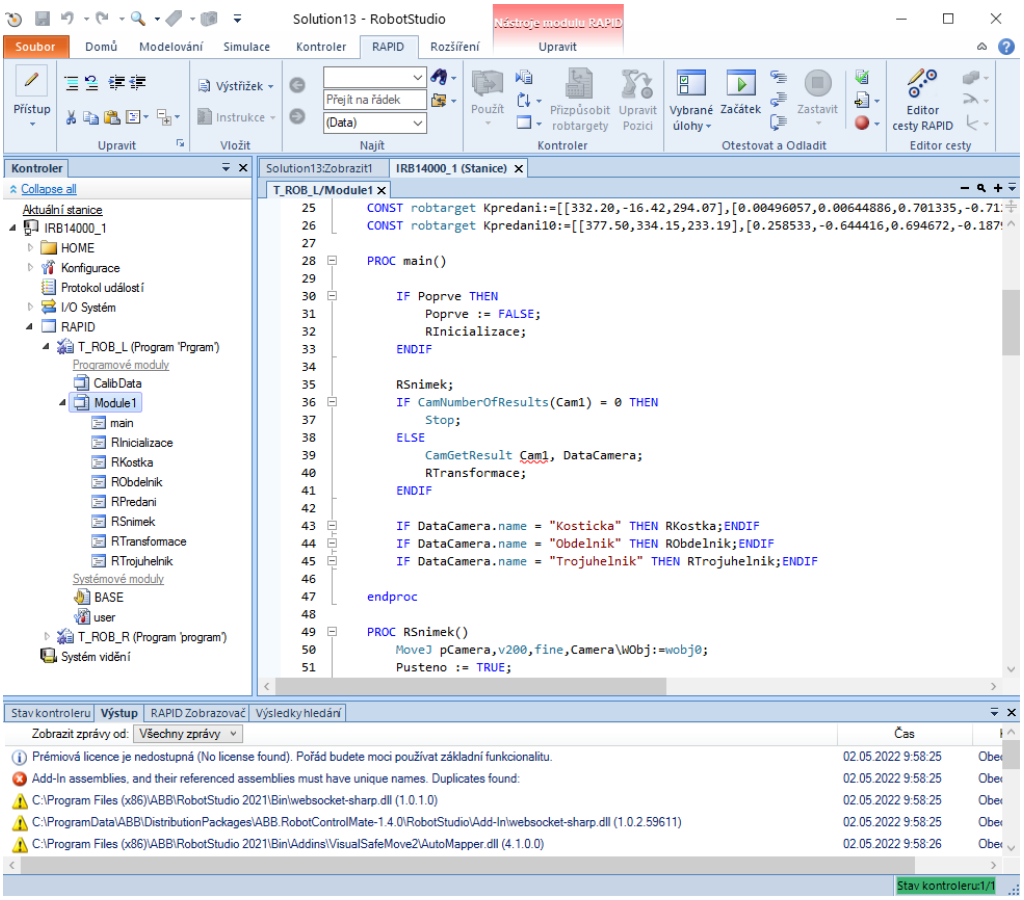
- Objekty – vytvořené jednoduché objekty základních geometrických tvarů. K objektům se mohou přiřazovat fyzikální vlastnosti, jako je hmotnost, momenty setrvačnosti. Objekty lze importovat ve standardních formátech CAD, nebo využít knihovny RobotStudia pro vložení předem vytvořených objektů, jako jsou pracovní stoly, zábrany, jednoduché předměty a nástroje. K vytvoření objektu nástroje je k dispozici funkce CreateTool. S touto funkcí si uživatel nastaví vlastnosti nástroje. Již vytvořené objekty a nástroje se dají přidat do knihoven a využívat je i v jiných projektech.
- Cestami, pohyby a targety – cesty se tvoří v simulovaném prostředí RobotStudia, skládají se z jednotlivých targetů. Pohyb mezi dvěma targety je definovaný jako lineární, obecný či kruhový. Ke každému pohybu se ještě přiřazuje rychlost a zóna, což je minimální požadovaná přesnost v jednotlivých bodech pohybu. Pro naprogramování cest lze použít také nástroj AutoPath a pro nastavení orientace nástroj AutoConfiguration. Cesty nemusí tvořit pouze targety, ale také mohou obsahovat události, například pro vyčkání, synchronizaci či zápis do proměnné v RAPIDu.
- I/O signály – uživatel má možnost definovat nové I/O signály. Signály se dají využít pro komunikování s okolím (tlačítka, kontrolky) nebo je pomocí nich možné v simulaci ovládat nástroje vložené do simulace. Nově založený kontrolér některé signály obsahuje. Jsou to hlavně signály určené pro chod kontrolérů, jako jsou signály nouzového zastavení, ovládání brzd, identifikace stavu motorů a podobně. Tyto stavy uživatel nesmí měnit (IRC5 a RobotStudio, 2018).



Obrázek 1.3 – Ukázka grafického off-line programování v software RobotStudio

1.4.1 Jazyk RAPID

RAPID je jazyk vytvořený pro programování robotů ABB. Program je psán v textovém editoru integrovaném do RobotStudia. Náhled textového editoru je ukázaný na obrázku 1.4. Program RAPID se skládá z řady instrukcí, dat a rutin, které popisují práci robota. Existují tři druhy rutin: procedury, funkce a trap rutiny. Trap rutiny poskytují reakci na přerušení, například při změně vstupního digitálního signálu vykonají určitou proceduru.



Obrázek 1.4 – Ukázka textového editoru RAPID v software RobotStudio

Základní dělení programu je na programové a systémové moduly. Programové moduly se skládají z dat a rutin. Hlavní programový modul musí obsahovat vstupní proceduru s názvem *main*. Systémové moduly se vytvářejí pro definování společných dat a rutin. Jednotlivé moduly jsou znázorněné položkami v seznamu po levé straně okna na obrázku 1.4.

Programy jsou rozděleny na jednotlivé procedury nebo funkce. Začátek procedury je deklarován klíčovým slovem *PROC*, konec procedury pak *ENDPROC*. Ukázka zápisu lokální procedury se vstupní proměnou a s konstrukcí vícenásobného větveného programu:

```
LOCAL PROC Nazev(num vstupni_promena)
    TEST vstupni_promena
    CASE 1:
RETURN;
      !Další instrukce
    CASE 2:
      !Další instrukce
    DEFAULT:
      !Další instrukce
    ENDTEST
```

```
ENDPROC
```

Ve všech rutinách může být definovaná oblast, ve které je daná rutina viditelná, pokud se před rutinu nenapíše klíčové slovo *LOCAL*, je rutina globální. Procedura se ukončí instrukcí *RETURN* nebo nepřímo, když je dosaženo konce.

Na rozdíl od procedur, funkce vrací hodnotu. Ukázka zápisu funkce se vstupní/výstupní proměnou a podmíněnou konstrukcí *IF – THEN*:

```
FUNC num Nazev(num vstupni_promena)
    IF vstupni_promena < 0 THEN
      RETURN Abs(vstupni_promena);
    ENDIF
ENDFUNC
```

Trap rutina umožnuje reaguje na vyvolané přerušení. Ukázka zápisu trap rutiny:

```
TRAP nazev_preruseni
    !Další intrukce
    RETURN;
ENDTRAP
```

1.4.2 Datové typy

RAPID pracuje se třemi druhy dat. Jejich klíčová slova a vlastnosti jsou následující:

- VAR – hodnotu proměnné je možné v programu měnit, při ukončení programu se hodnota nastaví na původní hodnotu.
- PERS – hodnotu proměnné je možné v programu měnit, avšak při ukončení programu se aktuální hodnota uloží.
- CONST – hodnota je přiřazena při deklaraci proměnné, v programu již nelze změnit.

Pro přiřazení hodnoty k proměnné se v jazyku RAPID používají znaky přidělení „:=“.

Jazyk dále disponuje sadou strukturovaných datových typů, mezi nejužívanější patří robtarget:

Robtarget Nazev := [[Pos],[Orient],[Confdata],[Extjoint]];

Kde jednotlivé složky mají následující význam:

- Pos – definuje pozici středu nástroje.
- Orient – definuje otočení nástroje pomocí čtveřic neboli quaternion.
- Confdata – definuje konfigurace os robota.
- Extjoint – definuje pozice až šesti externích os.

1.4.3 Instrukce pro ovládání robota

RAPID je speciálně navržen pro práci s roboty, a poskytuje speciální instrukce pro pohyb. Pro pohyb jsou tři základní instrukce:

- MoveL – slouží pro lineární pohyb nástroje.
- MoveC – slouží pro kruhový pohyb nástroje.
- MoveJ – slouží pro efektivní pohyb k cílovému bodu po nedefinované křivce.

Celý zápis instrukce MoveL je:

```
MoveL P10,v30,fine,Servo\WObj:=wobj0;
```

Kde jednotlivé parametry značí:

- P10 - definuje target, ke kterému se má robot vydat.
- v30 – uvádí rychlost, jakou se robot má pohybovat.
- fine – uvádí, že robot musí projít daným targetem.
- Servo – označuje, jaký nástroj je zvolen.
- WObj:=wobj0 – použitý souřadnicový systém, volitelný parametr.

2 STROJOVÉ VIDĚNÍ

Pojem strojové vidění je užíváno pro průmyslové systémy, automaticky zpracovávající snímky z kamer. Na základě tohoto vyhodnocení systémy vykonávají automatizovanou činnost. V této kapitole jsou uvedeny některé z používaných kamer a dostupný software.

Základní komponentou u strojového vidění je kamera. Kamera se skládá ze čtyř hlavních komponentů: objektiv s osvětlením, senzor, procesor a komunikace. U většiny kamer či senzorů je k dispozici celá škála možností, jak je sestrojit. Základní koncept robota řízeného strojovým viděním spočívá v tom, že kamera pořídí snímek předmětu, analyzuje obraz a přepošle získané souřadnice s dalšími daty robotu. Robot řízený strojovým viděním využívá čtyři druhy kamer:

- Kamerové senzory – Jedná se o nejlevnější a nejjednodušší variantu strojového vidění. V kamerovém senzoru jsou integrovány všechny prvky systému strojového vidění. Tato varianta se používá například pro zjištění přítomnosti předmětu, jeho orientace nebo pozice.
- Inteligentní kamery – Podobná kamerovému senzoru, na rozdíl od něj je inteligentní kamera vyráběna s výkonnější jednotkou, která zvládne sofistikovanější analýzy strojového vidění. Používá se pro náročnější aplikace. Mezi tyto aplikace může patřit rozeznávání textu, čtení tištěných kódů. Programování inteligentních kamer je složitější, programuje se ve specializovaných prostředích. Ke kamerám je k dispozici sortiment příslušenství, jako externí ovládací přísvit nebo objektivy.
- Systémy s průmyslovými kamerami – Jedná se o systém s více kamerami, obraz z těchto kamer zpracovává jedna centrální jednotka. Takto se dají řešit náročné úlohy, hodí se také pro aplikace, kde je potřeba zpracovávat obraz pro hlavní řídicí systém. Kamera je založena pouze na pořízení obrazu, jeho digitální transformaci a odeslání dat do řídicí jednotky ke zpracování.
- Kamerový systém s umělou inteligencí – Jedná se o systém s jednou kamerou, kde se při zpracování obrazu využívá umělá inteligence. Umělá inteligence se naučí na vzorku správně vyrobených součástek. Poté v provozu dokáže odhalovat i nenápadné defekty a vady podobně jako člověk (Havle, 2008).

Kromě kamer se používají 3D skenery a laserové radary zvané LIDAR. Strojové vidění vnáší do robotiky možnost provádět přesnější a komplikovanější úlohy. Klasickými úlohami jsou:

- Detekce vad,

- navádění robota pro přesné svařování, vrtání, lepení a další operace,
- k měření výrobků,
- třídění nebo počítání neorientovaných dílů,
- testování a kalibrace.

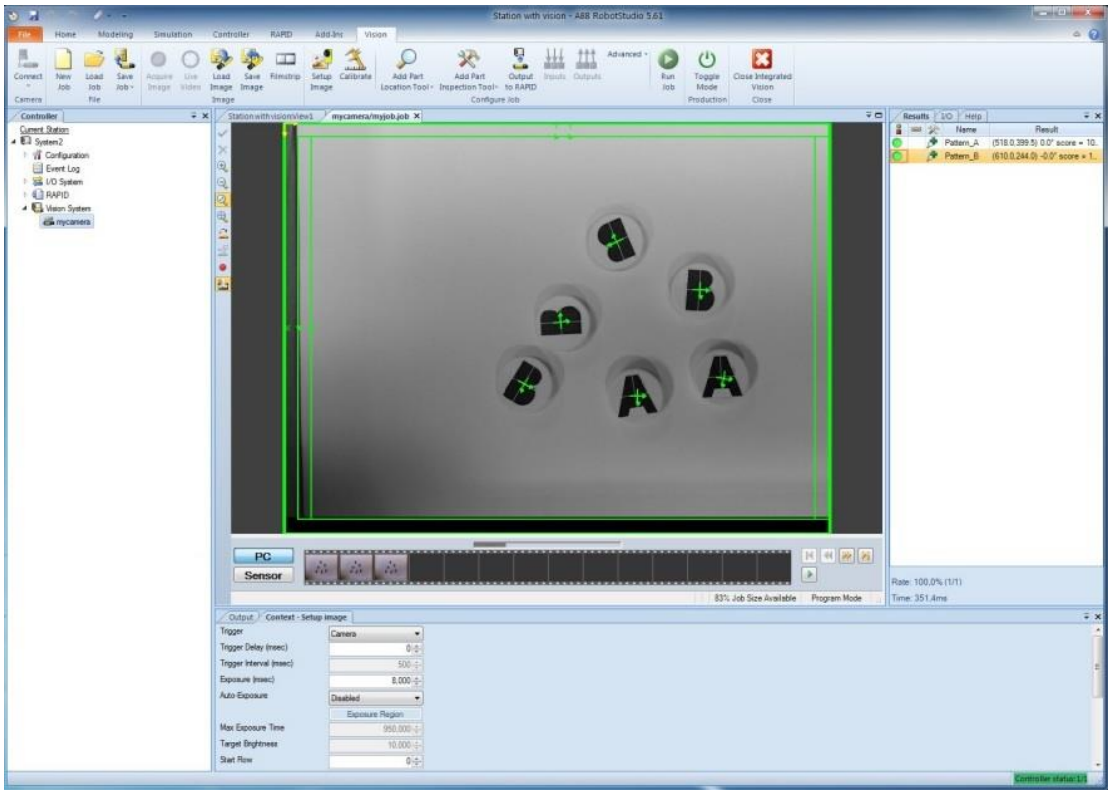
Kamery mohou být dodávány samostatně nebo mohou být integrovány na průmyslových robotech. Samostatně umístěná kamera přináší univerzální řešení s velkým výběrem požadovaných parametrů a jednoduchým začleněním do výroby bez dalších nákladů. Trendem je však integrování kamer do robota. Nevýhodou integrovaných kamer je, že ve většině případu má výrobcem dané parametry objektivu a přísvitu, které jsou neměnné. Pokud je kamerou naváděn robot, jedná se o aplikaci Vision Guided Robot. Takové vedení robotů strojovým viděním umožnuje větší míru flexibility výrobních procesů (Co je strojové vidění a jak může pomoci?, 2019).

Jedním z předních dodavatelů komplexních systému strojového vidění je firma Cognex. Jejich řešení implementují přední výrobci robotů do svých výrobků. Software strojového vidění

Cognex VisionPro je například integrován do systému Integrated vision od společnosti ABB nebo VisionTech od společnosti KUKA.

2.1 INTEGRATED VISION

Jedná se o systém společnosti ABB určený pro poskytnutí snadného používání aplikací Vision Guided RobotICS. Systém obsahuje celé softwarové a hardwarové řešení strojového vidění, které je integrované do řídicího kontroléru robota a prostředí RobotStudio. Strojové vidění je založeno na řadě chytrých kamer Cognex In-Sight s integrovaným zpracováním obrazu. Integrated Vision je vybaven nabídkou nástrojů Cognex EasyBuilder pro umístění, kontrolu a identifikaci předmětů. Pro práci s Integrated Vision je jazyk RAPID rozšířen o příslušné instrukce. K systému se dají připojit celkem tři kamery. Připojují se přes Ethernet switch ke kontroléru, napájené jsou 24 V z kontroléru. Vzhled systému je na obrázku 2.1.



Obrázek 2.1 – Vzhled systému Integrated Vision (Application manual – Integrated Vision, 2019)

V horní části je umístěna hlavní lišta s funkcemi pro práci se strojovým viděním. Vlevo je umístěné okno s již existujícími nastavenými nástroji pro identifikaci a kontrolu předmětů. Uprostřed je umístěn snímek s předměty. Zelené ohraničení okolo snímku je prostor, kde nástroje strojového vidění konají svou činnost. Každý nástroj může mít tento prostor vyhrazen samostatně. Pokud toto ohraničení je červené, znamená to, že nástroj nesplnil svůj cíl. Identifikované předměty jsou označeny dvěma šipkami, které značí orientaci předmětu. Pod snímkem je takzvaný Filmstrip bar, ten se používá k záznamu série snímků pro pozdější analýzu a optimalizaci parametrů nástrojů nebo kamery. Pod položkou Filmstrip bar je umístěno okno pro správu dostupných vlastností a různých nastavení. Na pravé straně je k dispozici seznam použitých nástrojů a digitálních vstupů/výstupů. Nejdůležitějšími prvky hlavní lišty jsou:

- Setup Image – nastavení parametrů obrazu.
- Calibrate – kalibrace obrazu k souřadnému systému.
- Add Part Location Tool – přidává nástroj ze sady nástrojů pro lokalizaci a identifikaci.
- Add Part Inspection Tool – přidává inspekční nástroj.
- Output to RAPID – zde se vybírají parametry, které mají být exportovány do RAPIDu.

- Run Mode – Přepínání mezi programovacím a běhovým režimem kamery.

2.1.1 Nastavení obrazu

Správné nastavení obrazu je jedním z předpokladů pro účinné využití strojového vidění. Kvalitu pořízeného snímku nejvíce ovlivňují následující parametry:

- Intenzita přísvitu,
- ohnisková vzdálenost,
- doba expozice,
- podmínky vnějšího prostředí.

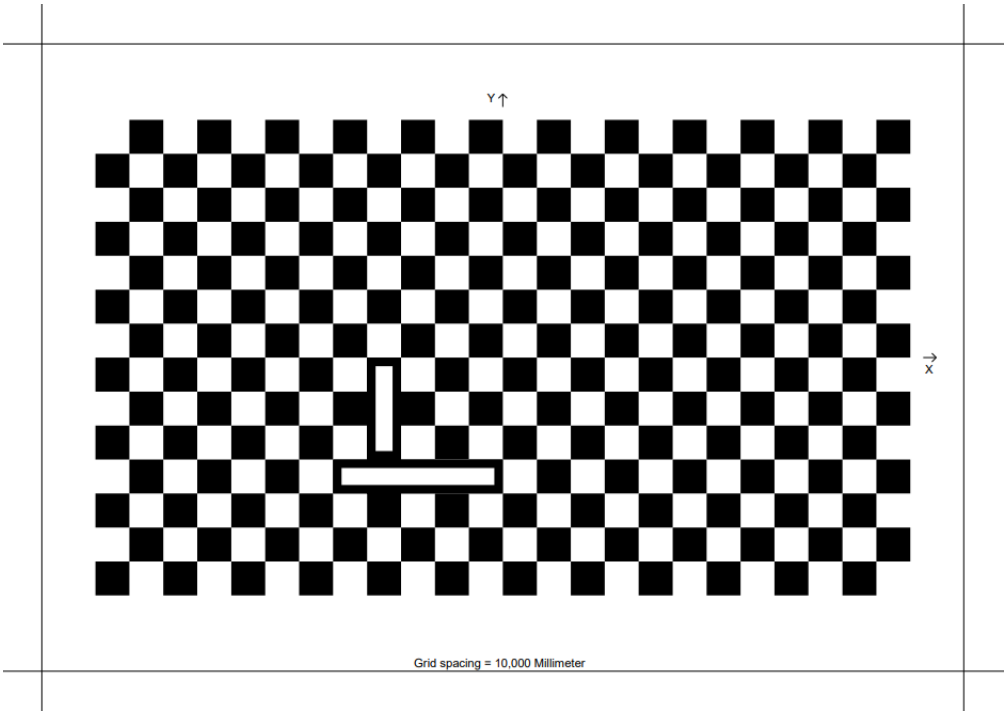
Podmínky vnějšího prostředí lze eliminovat jejich stálostí. Pokud jsou tyto podmínky proměnné, například záblesky, změna intenzity světla či stíny, pak při vyhodnocování pořízených snímků se mohou objevovat anomálie, které zhorší kvalitu u požadovaného výsledku zpracování obrazu. Ostatní parametry se nastavují podle možností při dané aplikaci.

2.1.2 Kalibrace obrazu

Získaný snímek se skládá z pixelů, pro převod na metrickou soustavu je třeba specifikovat vztah mezi pixelem a jednotkou délky. Kalibrace se skládá ze dvou kroků. Prvním krokem je kalibrace obrazu, tedy převod z pixelů na mm. Jako druhý krok je kalibrace kamery na souřadný systém robota. Druhý krok není nutný, je-li kamera integrovaná. Možnosti kalibrace jsou následující

- X/Y Scale – zadávají se rozměry pořízeného snímku.
- Edge to Edge – udává se rozměr mezi dvěma identifikovanými hranami obrazu v jedné ose.
- X/Y Edge to Edge – udávají se rozměry mez dvěma páry hran. Jeden pár hran ve vertikální ose a druhý v horizontální ose.
- Circle – kalibruje se podle známého průměru kruhu.
- Grid – kalibrace mřížkou. Na papír se vytiskne obrázek, na kterém je šachovnicový nebo tečkovaný mřížkový vzor. Obrázek mřížky se generuje při zvolení kalibrace přes Grid. Při vytváření se definuje typ mřížky a rozestup mřížky. Vytištěná mřížka se umístí pod kameru a přejde se ke kalibraci. Náhled na vygenerovanou kalibrační mřížku je na obrázku 2.2.

- Import – importování již vytvořeného kalibračního souboru.



Obrázek 2.2 – Vygenerovaná kalibrační mřížka (Application manual– Integrated Vision, 2019)

Pro velmi přesné aplikace strojového vidění je vhodné použít metodu kalibrace přes Grid. Dále je třeba zkontrolovat, zda kamera je kolmo k povrchu. Pokud tomu tak není, kalibrace bude nepřesná. Integrated Vision nemá k dispozici nástroje pro nelineární kalibraci.

2.1.3 Nástroje identifikace a lokalizace

Nejčastěji používané nástroje pro identifikaci a lokalizace jsou:

- PatMax Pattern – Tato metoda vyhledává geometrické obrazce. Pro správnou funkci je třeba určit vzor, ten se uloží, a následně se porovnává. Je-li vzor stejný, nástroj vrátí polohu v souřadnicovém systému pořízeného obrazu, úhel natočení oproti vzoru a skóre podobnosti.
- Blob – Vyhledává skupiny tmavých nebo světlých spojených pixelů a vrátí polohu v souřadném systému.

2.1.4 Příkazy pro obsluhu v jazyce RAPID

Pro ovládání kamery jsou v jazyce RAPID připraveny instrukce. Nejpotřebnějšími pro obsluhu kamery jsou:

- CamSetProgramMode – Nastaví kameru do programovacího režimu.
- CamStartLoadJob – Nahraje do kamery úlohu strojového vidění.
- CamWaitLoadJob – vyčkává, dokud se nenahraje nahrávaná úloha.
- CamSetRunMode – nastaví kameru do běhového režimu.
- CamReqImage – přikáže kameře pořídit snímek.
- CamGetResult – vyčte zpracovaná data z pořízeného snímku.

3 ROBOT ABB YUMI

Robot YuMi, zkráceně „You and Me" (tedy „ty a já"), byl první na trhu kolaborativních robotů, které mohou úzce spolupracovat s člověkem. Společnost ABB tohoto robota s celým názvem YuMi – IRB 14000 oficiálně uvedla na trh 13. dubna 2015 v německém městě Hannover. Byl to první krok k otevření obrovského potenciálu, který nabízí přímá kooperace mezi člověkem a strojem. Robot je určen k práci ve sdíleném prostředí s člověkem bez klecí a optických závor. YuMi přinesl do tohoto odvětví nové pojetí bezpečnosti.

Jedná se o dvouramenný robot s integrovaným kontrolérem, vybavený dvěma sedmiosými pažemi zakončenými efektorem zvaným Smart Gripper. Jeho hmotnost je 38 kg, šířka 339 mm a hloubka 497 mm. Smart Gripper je modulární efektor, který je osazen různými kombinacemi prvků. Jedním z prvků může být integrovaná kamera. Účelem tohoto systému integrovaného vidění je poskytovat robustní a snadno použitelný systém pro všeobecné použití aplikací Vision Guided Robotics (VGR). Součástí robota je kompletní softwarové a hardwarové řešení programovatelné v prostředí RobotStudio.

Robot Yumi pracuje s přesností 0,02mm a vykonává pohyb rychlostí 1,5 m/s se zrychlením 11 m/s^2. Jedno rameno uzvedne 500 gramů. Byl konstruován pro automatizované procesy při přesné výrobě s kooperací člověka. Například při montáži drobných součástek při výrobě spotřební elektroniky. Vzhled robota je znázorněn na obrázku 3.1.

Obrázek 3.1 – Robot ABB YuMi IRB 14000 (IRB 14000 YuMi, 2020)

Jak již bylo řečeno, robot YuMi se skládá ze dvou robotických ramen. Každé z ramen je programováno samostatně, a je na ně pohlíženo jako na dva roboty se společným kontrolérem.

3.1 BEZPEČNOST ROBOTA YUMI

Hlavním bezpečnostním prvkem robota YuMi je robot sám. Je navržen tak, aby eliminoval veškeré nebezpečí. Splňuje bezpečnostní normy všech kategorií, které jsou potřeba pro zavedení tohoto kolaborativního robota do výroby. Robot YuMi má skelet vyroben z magnezia, což je lehký, ale velmi pevný materiál. Skelet je obalený plastovými kryty s polstrováním, které mají za cíl pohltit co nejvíce síly a minimalizovat dopady kolize. Už ze své konstrukce minimalizuje riziko skřípnutí citlivých částí lidského těla, jako jsou prsty mezi dvěma protilehlými povrchy. Pokud robot zaznamená sebemenší kolizi, která by mohla být pro člověka nebezpečná, je schopný zastavit pohyb během milisekundy, aby se zabránilo zranění. Pokud robot takto pohyb zastaví, vyhlásí chybu kolize a čeká na opětovné spuštění. Opětovné spuštění lze provést přes dálkový ovladač FlexPendant, či uživatelskou aplikaci. Po spuštění robot pokračuje v programu, kde byl zastaven, a to vše s přesností 0,02mm. Robot plně splňuje normu ČSN EN ISO 13849.

3.2 FLEXPENDANT

FlexPendant je ruční terminál s barevným dotykovým panelem, určeným k ovládání a programování robota YuMi. Displej umožňuje zobrazovat informace o stavu robota, programovat robota online a přístup k uživatelskému rozhraní. Je opatřen joystickem a několika tlačítky pro ovládání. Součástí je také tlačítko nouzového zastavení, které je možné použít, když je zapotřebí systém nouzově zastavit. Všechny prvky FlexPendantu jsou zobrazeny na obrázku 4.2. Robot se ovládá ve dvou režimech – v ručním a automatickém režimu.

Obrázek 3.2 Vzhled FlexPendantu (IRC5 a RobotStudio – Návod k použití, 2018)

V ručním režimu, jak už z názvu vyplývá, je robot pod ruční kontrolou. Používá se při programování a ověřování programu robota. K dispozici je ruční režim se sníženou rychlostí a ruční režim s plnou rychlostí. Režim s plnou rychlostí umožňuje robotu pohybovat se naprogramovanou rychlostí, jinak je pohyb omezen na 250 mm/s. V ručním režimu je k dispozici:

- Kalibrace,
- ruční přestavení,
- editor programu,
- editor dar,
- krokový režim.

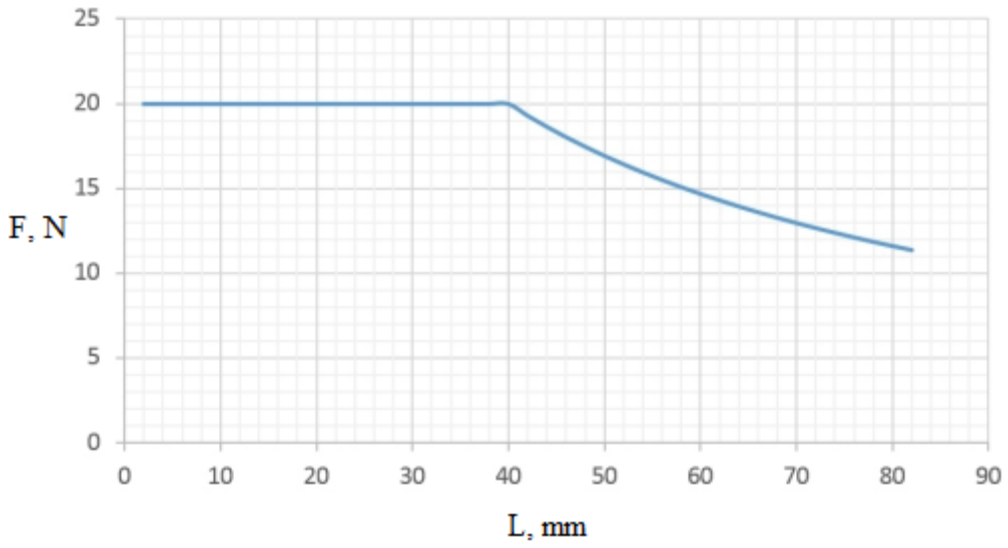
Automatický režim slouží ke spuštění programu robota v provozu. V tomto režimu není možné provádět jakékoliv programové změny, ale je možné upravit rychlost pohybu. Nabídka funkcí je uzamčena.

3.3 SMART GRIPPER

Jedná se o zápěstí robota, které je dodáváno v několika variantách. Mezi nejčastěji používané patří například úchopové čelisti, kamera, přísavky na bázi podtlaku a jejich kombinace. Vzhled a dostupné kombinace jsou znázorněné na obrázku 4.2.

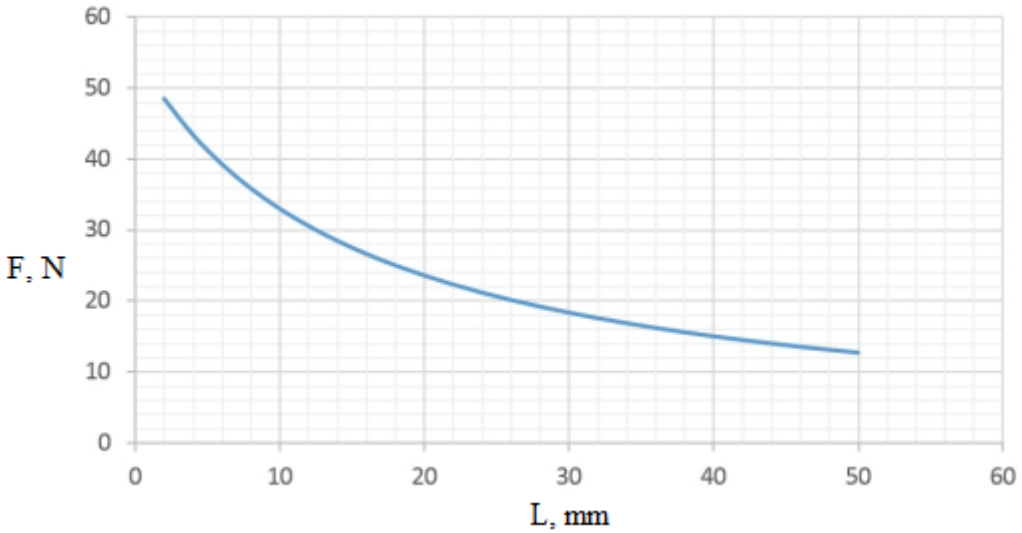
Obrázek 3.3 – Varianty SmartGripu (IRB 14000 YuMi, 2020)

Servomodul, neboli uchopovací čelisti, jsou umístěny na konci SmartGripperu, v něm je také integrováno řízení polohy čelistí, s opakovatelnou přesností 0,05mm. Modul je možné kalibrovat pomocí příkazu v programu RAPIDu nebo pomocí jednotky FlexPendantu. Maximální rozsah roztáhnutých čelistí je 50 mm, roztahovat se dokážou rychlostí 25 mm/s. Maximální síla, kterou čelisti dokáží vyvinout pro uzavření, je 20 N. V programu RAPID je možno nastavit požadované roztáhnutí čelistí, i zvolit velikost uchopovací síly. Nastavení uchopovací síly je s přesností 3 N. Uchopovací síla je závislá na délce čelistí podle obrázku 4.3.



Obr. 3.1 – Kalibrační přímka DAC převodníku

Následující graf na obrázku 4.4 ukazuje maximální povolenou externí sílu, kterou je možné zatížit čelisti před jejich destrukcí.



Obrázek 3.4 – Maximální přípustná síla (F) působící externě na čelisti o délce (L) (Product manual – IRB 14000 gripper, 2021)

Vakuový modul má integrovaný generátor podtlaku, který je závislý na přivedeném tlakovém vzduchu. Je navržen na maximální užitečné zatížení 150 g. Nutno podotknout, že toto užitečné zatížení závisí na:

- Konstrukci přísavky a jejího materiálu,
- struktuře povrchu přisávaného předmětu,
- na poloze přisávaného předmětu,
- na tlaku přivedeného vzduchu.

Tlak vzduchu přísavky je možné sledovat díky vestavenému senzoru tlaku uvnitř přísavky. Je tedy možné zjistit, jestli je předmět uchopen nebo zdali upadl.

Kamerový modul je založen na kameře z rodiny Cognex In-Sight s integrovaným zpracováním obrazu a komunikačním rozhraním přes síťové připojení. Přesné jméno kamerového modulu je Cognex AE3. Tento modul je výkonný a spolehlivý nástroj pro strojové vidění a identifikaci. Kamera má následující parametry:

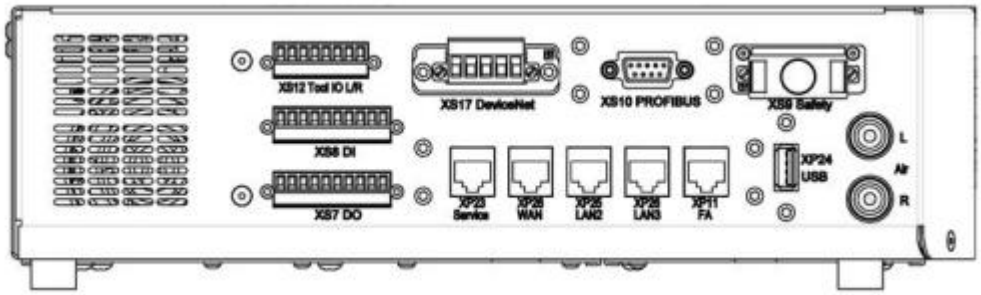
- Rozlišení 1,3 Mpx,
- objektiv 6,2 mm f/5,
- integrované osvětlení s nastavitelnou intenzitou,
- možnost naprogramovat kameru v ABB Integrated vision nebo v Cognex In-Sight Explorer.

Jazyk Rapid je rozšířen o funkce pro ovládání a komunikaci s integrovaným strojovým viděním, jak je podrobněji popsáno v kapitole 2.1.

3.4 ROZHRANÍ ROBOTA YUMI

Jedním ze základních přístupových rozhraní u robota YuMi je jednotka FlexPendant popsaná v kapitole 3.2. Ke komunikaci mezi robotem, uživatelskými rozhraními a dalšími zařízeními najdeme u robota YuMi celou paletu možností. Jsou k dispozici:

A. I/O porty – K dispozici je osm digitálních vstupů a výstupů, umístěných na boční straně kontroléru. Tyto I/O se mohou použít pro jednoduchou komunikaci mezi obsluhou a robotem. Například kontrolka signalizuje připravenost robota na odebrání dílu a obsluha potvrdí tlačítkem založení toho dílu pro odebrání. Tato komunikace může stačit na menší a jednodušší aplikace. Dostupné I/O jsou zobrazeny na obrázku 3.6.

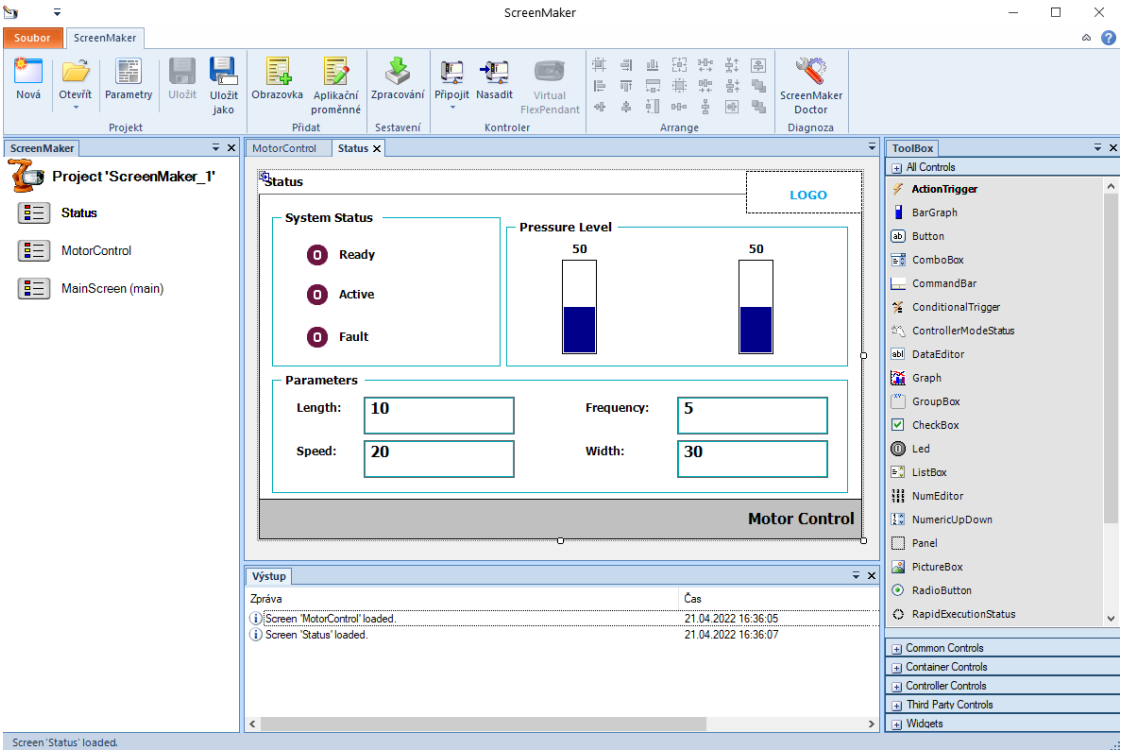


Obrázek 3.5 – Ukázka vstupně výstupních portů kontroléru robota YuMi (Operating manual – IRB 14000, 2019)

B. Podpora protokolu TCP/IP v jazyku RAPID – v jazyku RAPID jsou k dispozici příkazy pro naprogramování serveru, který může odesílat a přijímat data z TCP klienta přes protokol TCP/IP. Posílat data do klienta lze prakticky kdykoliv. Příjem dat musí probíhat buď v jeden konkrétní okamžik, nebo musí vyvolat přerušení, ve kterém se následně zpracují přijatá data. Tímto propojením je možné komunikovat mezi robotem a aplikací PC. K této metodě je třeba také vytvořit aplikaci v libovolném prostředí, které umožnuje komunikaci TCP/IP.
C. FlexPendant – tento HMI může sloužit kromě programování robota také k vytvoření uživatelského prostředí. Nejčastěji se používá pro drobné úpravy v programu a pro jednoduchou manipulaci. Uživatelské prostředí lze programovat dvěma způsoby. Prvním způsobem je programování aplikace pro FlexPendant v prostředí RobotStudio

pomocí software ScreenMaker se sadou FlexPendant SDK. Druhým způsobem je možnost programovat v prostředí Microsoft Visual Studio 2008. Sada FlexPendant SDK, která se pro programování používá, je založena na Windows CE 6 a .NET Compact Framework, které už nejsou podporovány v novějších verzích Visual Studio.

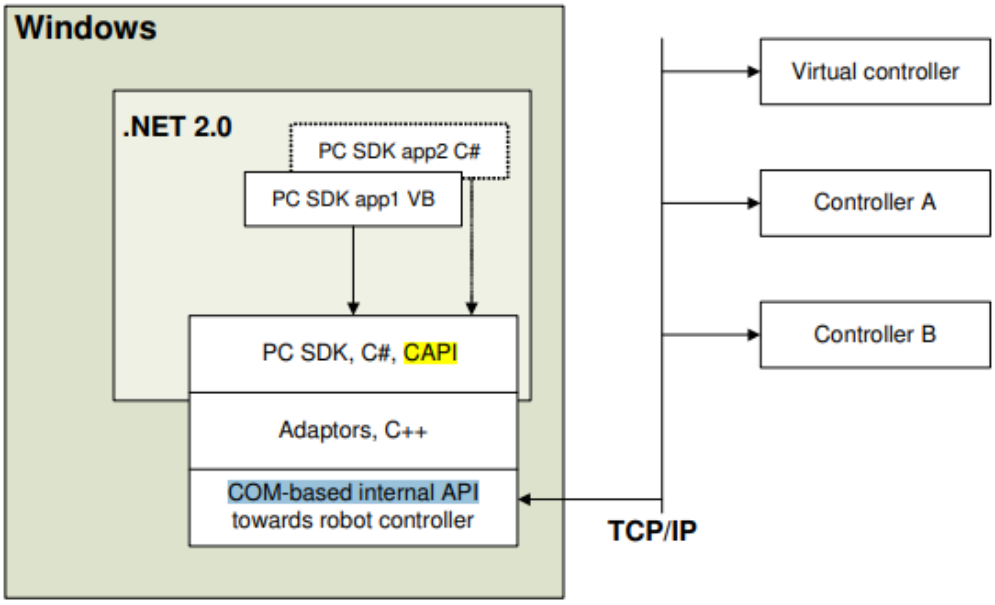
Prostředí ScreenMaker znázorněné na obrázku 3.7 je nástroj v RobotStudiu, který je určený pouze pro vývoj aplikací s grafickým uživatelským rozhraním pro FlexPendant. Nástroj je pro pochopení velmi jednoduchý a nevyžaduje žádné hlubší znalosti pro programování v prostředí, jako je Visual Studio. Aplikace je tvořena ze sady grafických komponent, jako jsou tlačítka, textová pole nebo obrázky, umístěných na dotykové obrazovce FlexPendantu, pomocí nichž uživatel aplikaci ovládá. Ke každé grafické komponentě je možné přiřadit událost, kterou daný prvek vyvolává. Jednotlivé události obsahují instrukce, které mohou reagovat na sepnutý I/O signál, zmáčknutí tlačítka nebo změnu proměnné v RAPIDu. Aplikace se skládá ze dvou částí, grafické části a části, která vyvolává jednotlivé události.



Obrázek 3.6 – Ukázka prostředí ScreenMaker

D. Interní Controller API s použitím PC SDK– tento softwarový nástroj, který je volně ke stažení na webu společnosti ABB, umožňuje vytvoření počítačové aplikace na OS

Windows nebo na systémech s podporou .NET 2.0. Počítačová aplikace komunikuje s robotem pomocí protokolu TCP/IP. Architektura software PC SKD je znázorněna na obrázku 3.8. Hlavní výhodou použití PC SDK, je úplná kontrola aplikace nad robotem. Omezením je, že k robotu může být přihlášena pouze jedna relace aplikace současně, a jakýkoliv další přístup robot zamezí, dokud se neukončí původní relace. Toto omezení je z důvodů bezpečnosti a ochrany dat před přepsáním. Menší nevýhodou je, že potenciál takto vytvořené aplikace se dá využít jen u automatického chodu. V manuálním režimu je uživatel nucen každý příkaz, který zapisuje do kontrolérů, ručně potvrdit na jednotce FlexPendant.



Obrázek 3.7 – Architektura software PC SDK (Application manual– PC SDK, 2012)

E. Robot web services – u tohoto nástroje je používáno webové rozhraní REST API. Možnosti webového rozhraní jsou zobrazené na obrázku 3.9. Rozhraní je poskytované samotným kontrolérem. Architektura REST pro API je datově orientovaná, nespouští tak vzdálené procedury, ale pracuje přímo s daty. Pro přístup k datům se používají čtyři metody: Create, Retrieve, Update a Delete. Těmito metodami může kompletně spravovat data na serveru. Všechny metody vždy vrací určitý návratový HTTP stav.

Obrázek 3.8 – Robot web Services (Application manual– Robot Web Services, 2020)

Hlavní rozdíl mezi těmito dvěma platformami je v tom, že PC aplikace vystupuje v roli vzdáleného klienta, zatímco aplikace FlexPendant je klientem lokálním. Vzdálený klient nemá všechna oprávnění místního klienta. Obě Aplikace PC a FlexPendant mohou resetovat ukazatel programu, spustit kód nebo přepisovat proměnné. Pro vzdáleného klienta toto platí jen při automatickém režimu. Při manuálním režimu se musí každý přístup do RAPIDu potvrdit na FlexPendantu.

4 PRAKTICKÉ ŘEŠENÍ ZADANÉ ÚLOHY

V následující kapitole bude popsán návrh a tvorba PC aplikace, která realizuje uživatelské rozhraní a program pro robot v jazyku RAPID. Jak bylo v úvodu řečeno, cílem diplomové práce je využít strojového vidění pro lokalizaci předmětů náhodně umístěných na podložce. Robot bude dále odebírat lokalizované předměty a umísťovat je podle typu na určené místo. K tomu je vytvořeno jednoduché uživatelské rozhraní.

Pro vytvoření programu v jazyce RAPID bylo využito prostředí RobotStudio 2021 popsané v kapitole 1.4. Pro vytvoření uživatelského rozhraní jsem použil prostředí Microsoft Visual Studio 2022, do kterého byl instalován nástroj PC SDK.

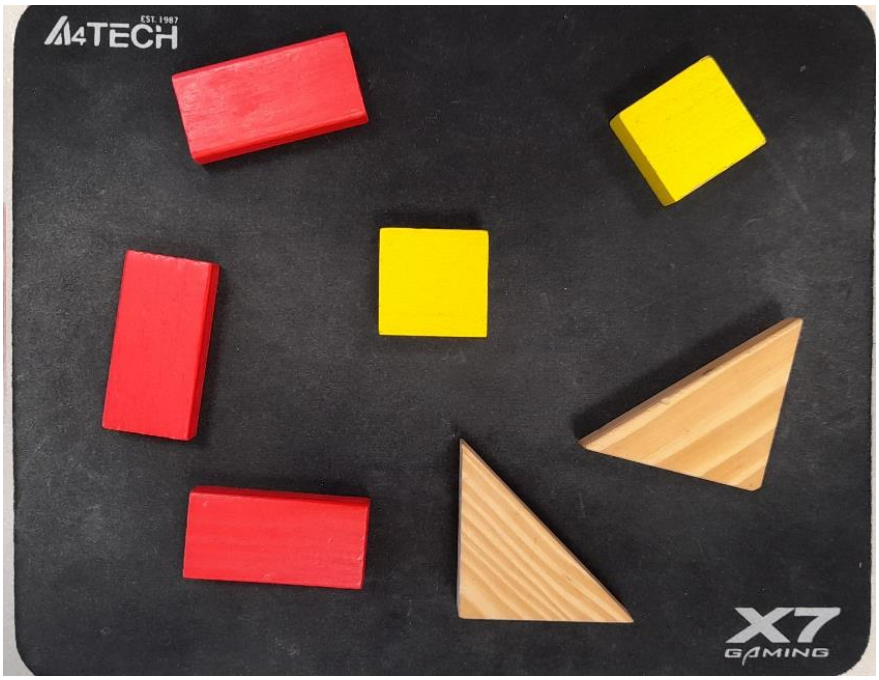
Výsledné řešení bylo aplikováno na robotickém pracovišti, umístěném v jedné z laboratoří Fakulty elektrotechniky a informatiky Univerzity Pardubice. Na robotizovaném pracovišti se nachází pracovní stůl s robotem YuMi. K dispozici je ovladač robota FlexPendant. Na pracovišti byla vyvinuta velká část programu RAPID, dále zde byla testována funkčnost uživatelské PC aplikace.

4.1 POUŽITÉ PŘEDMĚTY

Modul strojového vidění má lokalizovat a identifikovat náhodně umístěné předměty na podložce. Pro tento účel byly vybrány tři typy komponent různých tvarů, přičemž každý typ má odlišnou barvu. Jedná se o tvar rovnoramenného trojhranu, kvádru a kostky o rozměrech dle tabulky číslo 4.1, dále jsou vidět na obrázku číslo 4.1.

Tabulka 4.1 – Soupis předmětů

Typ	Rozměry	Barva
Kostka	Šířka: 30 mm Hloubka: 30 mm Výška: 15 mm	Žlutá
Kvádr	Šířka: 50 mm Hloubka: 25 mm Výška: 12 mm	Červená
Rovnoramenný trojhran	Základna: 70 mm Ramena: 50 mm Výška: 12 mm	Hnědá



Obrázek 4.1 – Scéna pro snímek strojového vidění s použitými součástkami

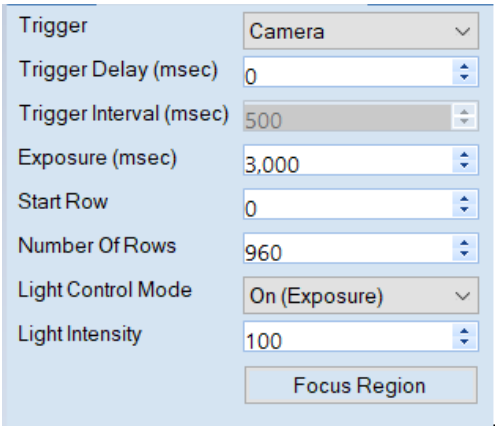
4.2 STRUČNÝ POPIS ČINNOSTI PROGRAMU PRO ROBOT

Funkce programu je navržena následovně. Koncový efektor levého ramene se přesune na místo nad podložkou. Z tohoto místa pořídí kamera snímek zachycující podložku s předměty, který se následně v prostředí Integrated Vision vyhodnotí. Vyhodnocená data se importují do příslušné proměnné v jazyce RAPID. Proměnné v tomto bodě obsahují polohu a typ předmětu. Podle těchto dat se provede transformace souřadnic targetu, umístěného kolmo k podložce v jejím pravém horním rohu. Koncový efektor najede nad součástku, následně se přiblíží lineárním pohybem směrem k součástce a uchopí ji. Uchopenou součástku předá buď druhému rameni, nebo ji přesune na místo, odkud součástku pravé rameno převezme. Pravé rameno následně roztříděné předměty rovná do sloupců podle typu předmětu. Jednotlivé kroky tvorby programu jsou popsány v kapitolách 4.3, 4.4 a 4.6.

4.3 LOKALIZACE PŘEDMĚTŮ S POUŽITÍM STROJOVÉHO VIDĚNÍ

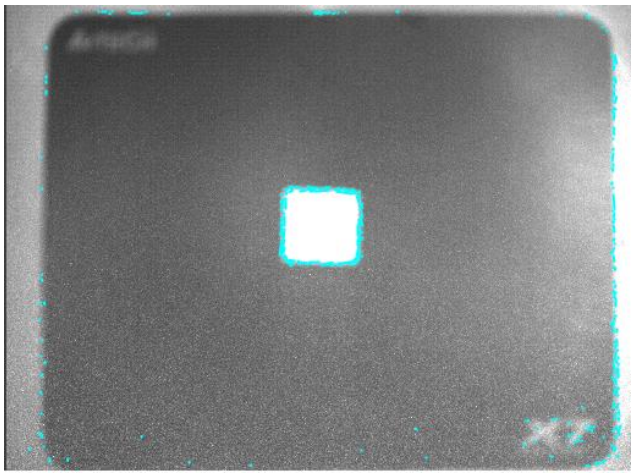
Stěžejní částí této práce je lokalizování součástek s použitím strojového vidění. K tomuto účelu je použita integrovaná kamera umístěná v koncovém efektoru robota. Parametry kamery jsou popsány výše v podkapitole 3.3 SmartGripper.

Robot je připevněný na kovovém pracovním stole s lesklým povrchem. Tento lesklý povrch by při pořízení obrazu zcela pohltil jakékoliv tvary do jednoho světlého celku. Aby byly na pořízených snímcích součástky zřetelně viditelné, byla použita matná pracovní podložka černé barvy. Pro zlepšení kvality snímku je jako první třeba nastavit Image Setup. Nabídka Image Setup je k vidění na obrázku 4.2. K dispozici je řada nastavitelných parametrů. Pro zadanou aplikaci bylo experimentováno s nastavením expozice a přísvitu. Nejostřejší hrany se jevily při expoziční hodnotě tři milisekundy a nastavení přísvitu na 100 %. Při této intenzitě přísvitu se nejvíce zviditelnily hrany předmětů.



Obrázek 4.2 – Nabídka Image Setup

Pro nenáročnost aplikace byla zvolena kalibrační metoda Edge To Edge, která je popsána v kapitole 2.1.2. U této metody je třeba je třeba zvolit dva protilehlé okraje, u kterých je známa vzdálenost. Jako kalibrační předmět byla zvolena kostka s šířkou 30 mm. Následný postup začíná u zvolení kalibrační metody a pořízení snímku, na kterém se bude kalibrace provádět. Při zvolení Edge To Edge se objeví výzva k označení již zmíněných dvou hran. Program detekuje možné hrany, ze kterých je třeba vybrat. Detekce hran je znázorněna na obrázku 4.3.

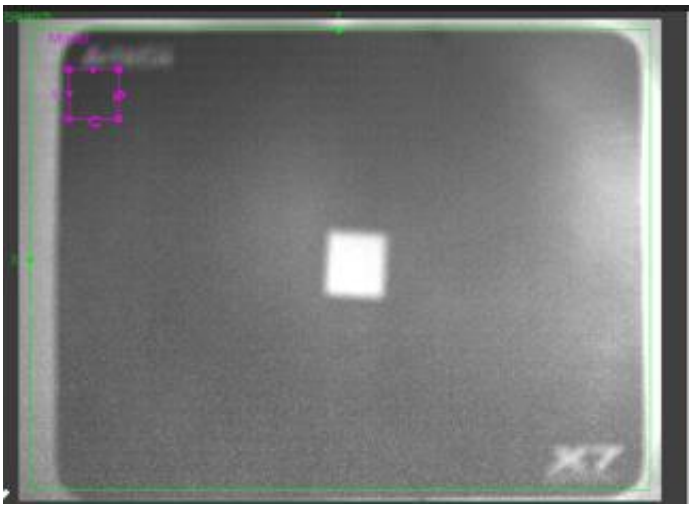


Obrázek 4.3 – Detekované hrany při kalibrační metodě Edge To Edge

Mezi označenými hranami se vygeneruje úsečka. Dále je třeba zadat délku úsečky, což kalibraci dokončí.

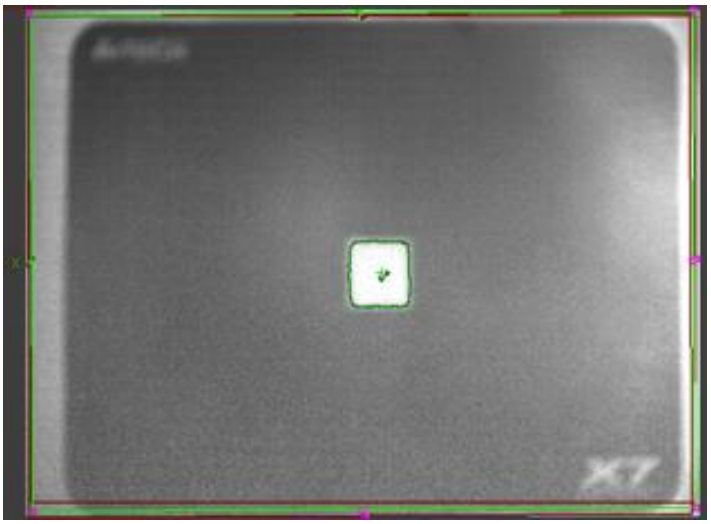
Pro lokalizaci tří typů předmětů byl vybrán nástroj PatMax Patterns. Tento nástroj je krátce popsán v podkapitole 2.1.3. Dále je popsán průběh nastavení nástroje u jednoho typu součástky.

Pro ukázku nastavení je zvolena součástka stejná jako v ukázce postupu kalibrace. Nástroj PatMax Patterns se aplikuje na nově pořízený snímek. Na snímku se objeví dvě oblasti, snímek s oblastmi je na obrázku 4.4. Oběma oblastem je možno přiřadit v nabídce k nástroji čtyři následující tvary: čtverec, kruh, polygon a mezikruží. První oblast s názvem model se umístí na součástku, která má být později lokalizována. Druhá oblast znázorňuje prostor, ve kterém se daný model bude lokalizovat.



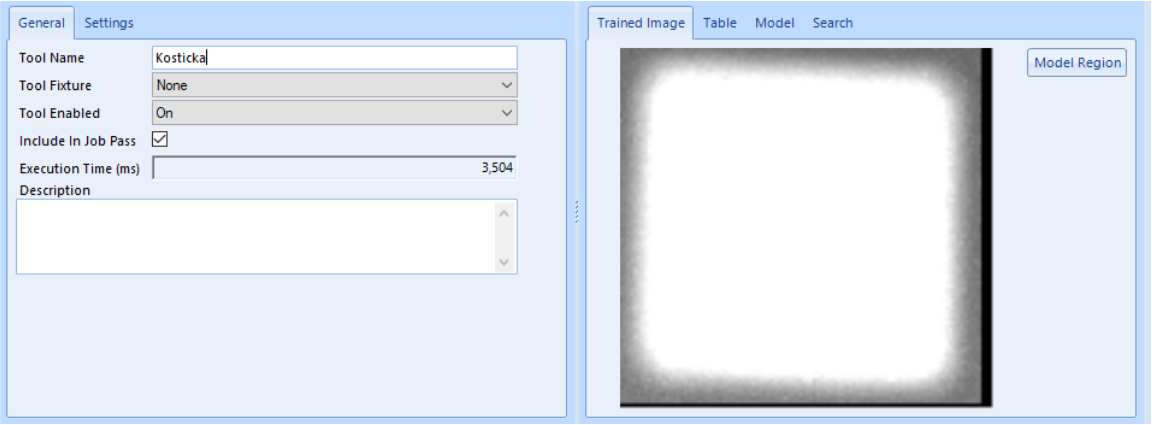
Obrázek 4.4 – Snímek po vybrání nástroje PatMax Patterns

Po ukončení nastavení se nástroj uloží do seznamu použitých nástrojů. Pokud je vše nastavené v pořádku, při spuštění úlohy by měl být předmět lokalizován jako na obrázku 4.5.



Obrázek 4.5 – Snímek s lokalizovaným předmětem

Dalším krokem je nastavení dalších parametrů vytvořeného nástroje. Nastavení se objeví po vybrání nástroje z nabídky vytvořených nástrojů. Po vybrání nástroje se objeví nabídka, která je vyobrazena na obrázku 4.6. V této nabídce v listu General je nejprve pojmenován nástroj. Dále je možné udělat vazbu nástroje s dalším nástrojem (Tool Fixture), a zvolit ovládací vstupní signál k zapnutí nebo vypnutí nástroje (Tool Enabled).

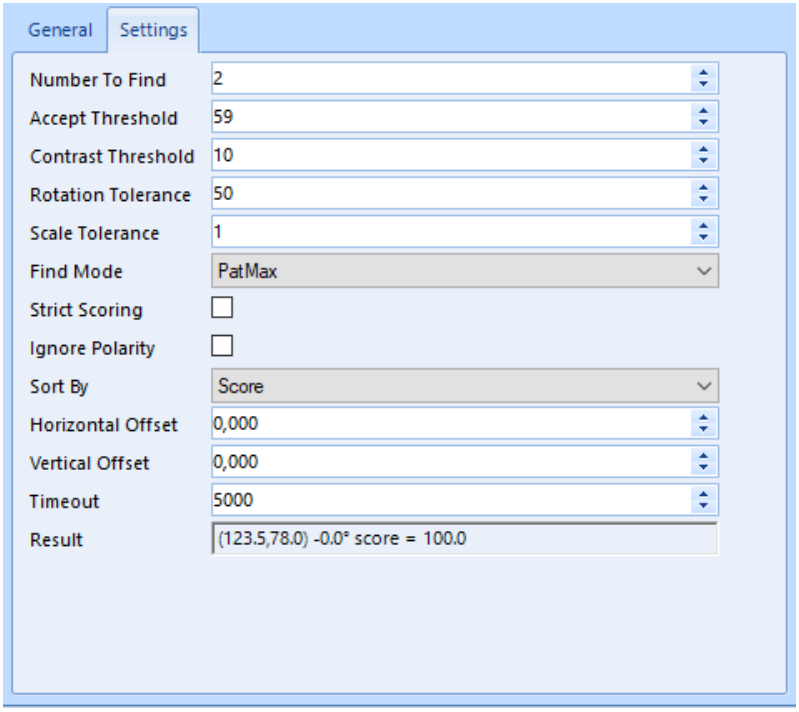


Obrázek 4.6 – Nabídka zvoleného nástroje

V dalším listu s názvem Settings je možné nastavovat další parametry. Nabídka Settings, vyobrazena na obrázku 4.7, nabízí následující parametry:

- Number To Find – kolik předmětů má být lokalizováno.
- Accept Threshold – nastavuje přijatelnou procentuální hladinu bodového označení podobnosti Score, při které bude předmět vyhodnocen jako shodný. Rozsah je od nuly do sta procent.
- Contrast Threshold – nastavuje přijatelnou procentuální hodnotu přijatelného kontrastu. Kontrast je zde myšlen mezi předmětem a podložkou. Rozsah hodnoty je od nuly do dvou set padesáti pěti.
- Rotation Tolerance – nastavuje hodnotu tolerance pootočení předmětu vůči modelu. Rozsah hodnoty je od nuly do sto osmdesáti pěti stupňů.
- Scale Tolerance – nastavuje toleranci měřítka v rozsahu padesáti procent. Rozsah hodnoty je tedy od nuly do padesáti.
- Find Mode – nabídka mezi algoritmem porovnávání vzorů, PatMax a PatQuick. PatMax nabízí vyšší přesnost, PatQuick vyšší rychlost. Algoritmus PatQuick nevrací hodnotu Score.
- Strict Scoring – možnost přísného ohodnocení podobnosti, časově náročnější.
- Ignore Polarity – možnost zpracovávat barevně polarizované vzory oproti vytvořeným modelům.
- Sort By – možnost řadit výsledky podle Score a vzálenosti od počátku v ose x a y.
- Horizontal Offset.
- Vertical Offset.

- Timeout – nastavení maximální doby činnosti nástroje. Údaj je v milisekundách.
- Result – zobrazení prvního výsadku nástroje.



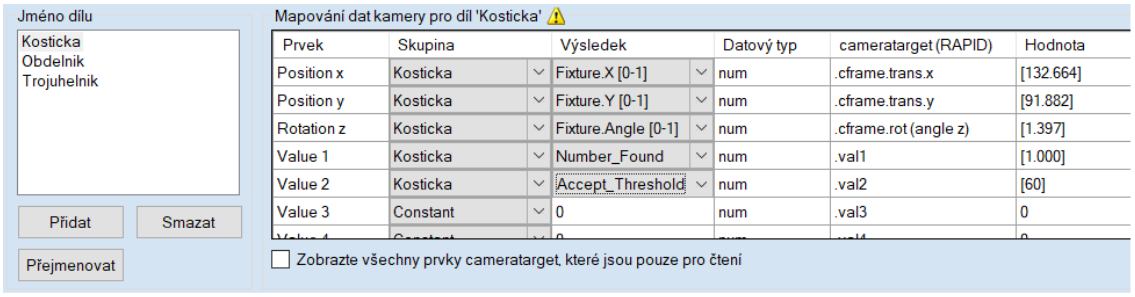
Obrázek 4.7 – Nabídka Settings

Hodnoty nastavené pro jednotlivé prvky jsou získané postupným optimalizováním parametrů při výskytu některé chyby. Výsledné parametry z tabulky 4.2 byly vyhodnoceny jako nejlepší.

Tabulka 4.2 – Vybrané parametry pro vytvořené nástroje jednotlivých tvarů

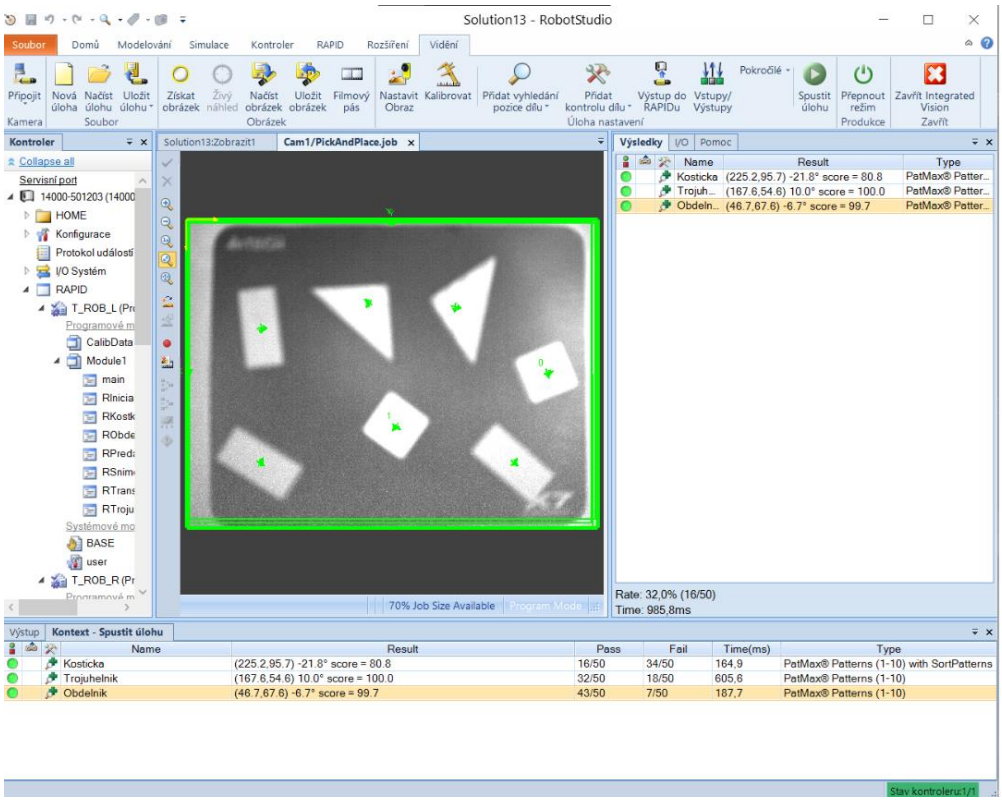
	Kostka	Hranol	Trojhran
Number To	3	3	3
Accept	90	90	90
Contrast	50	50	50
Rotation	45	90	180
Scale	2	2	2
Find Mode	PatMax	PatMax	PatMax
Timeout	200ms	200ms	200ma

Posledním krokem je nastavení požadovaného výstupu do jazyka RAPID. Zde se pro jednotlivé vytvořené nástroje definují výstupní parametry, které mají být odeslány. Názorná ukázka nastavení je zobrazena na obrázku 4.8.



Obrázek 4.8 – Výstup do jazyka RAPID

Vytvořená úloha pro modul strojového vidění s vyobrazenými výstupními daty je znázorněna na obrázku 4.9. Výstupní data v seznamu v pravé části obrázku udávají polohu v ose x, y a rotaci objektu. Rozsah rotace je dána od -90 st. do 90 st..



Obrázek 4.9 – Konečný výsledek úlohy pro modul strojového vidění

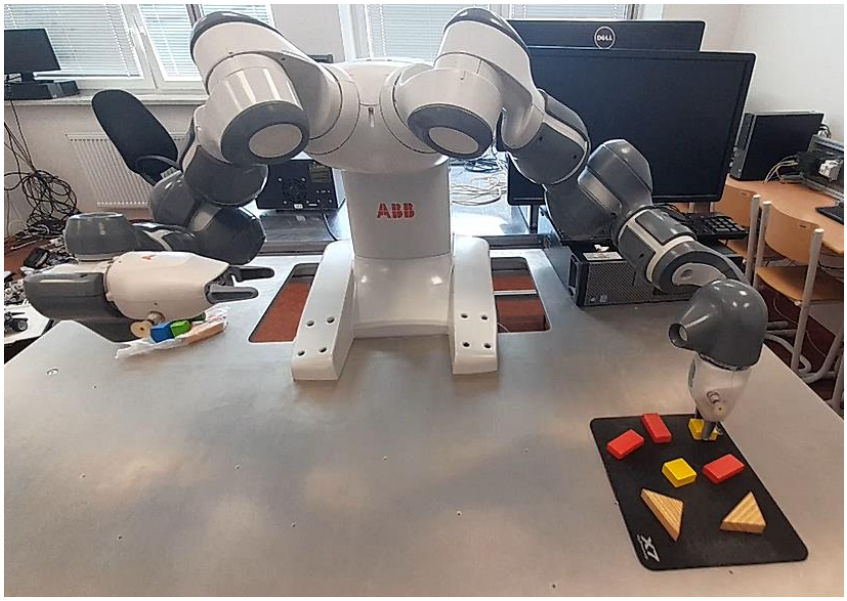
4.4 PROGRAM PRO ROBOT V JAZYCE RAPID

Jak je vysvětleno kapitole 3, robot YuMi se skládá ze dvou samostatných ramen a k jejich obsluze je tedy zapotřebí dvou programů. V této kapitole jsou tyto dva programy blíže popsány a vysvětleny. Programu odpovídá jeden programový modul. V tomto případě moduly T_ROB_L a T_ROB_P. Modul s názvem T_ROB_L je program pro levé rameno robota, modul T_ROB_P je pro pravé rameno.

Start programu začíná inicializací proměnných a modulu strojového vidění. Tato inicializace probíhá při prvotním spuštění programu. Pokud program již běží, tzn. že následuje druhý a další cyklus, inicializace proměnných se nespouští. Následuje pořízení snímku, které je zobrazené na obrázku 4.10. Není-li součástka lokalizována, program se zastaví, opětovné spuštění programu se provádí z uživatelského rozhraní.

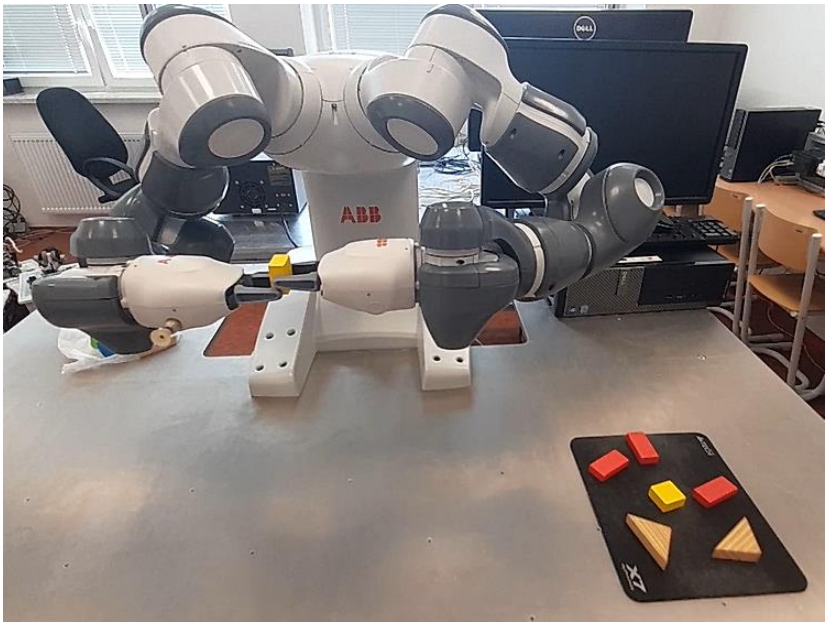
Obrázek 4.10 – Robot v poloze pro pořízení snímku

V případě nalezení součástky na podložce nastává přepočet souřadnic dle referenčního bodu umístěného v rohu pořízeného snímku. Následuje rozhodnutí programu, o jaký tvar součástky se jedná v podobě víceúrovňového větvení a provedení následných procedur pro uchopení součástky. Uchopení součástky je zaznamenáno na obrázku 4.11. Po vyhodnocení následuje další cyklus programu.



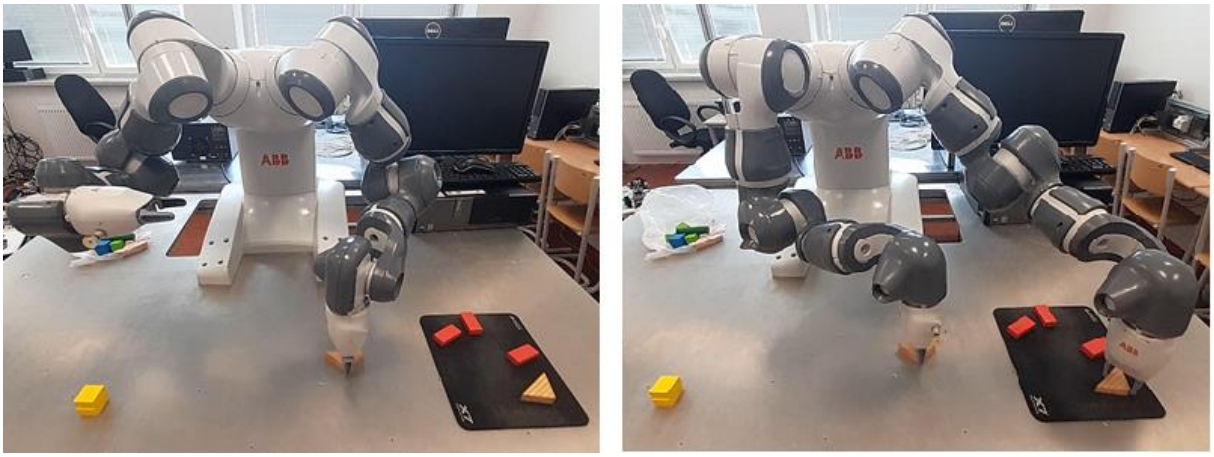
Obrázek 4.11 - Leva paže uchopující součástku

Program pravé paže je velmi podobný k programu levé paže. Po inicializaci se rozhoduje o povolení odebrání součástky. V případě, že povolení odebrání je nastaveno na FALSE, tak pravá paže čeká, dokud povolení nezíská nebo dokud nebude z uživatelského rozhraní pokyn k odebrání připravených trojhranů. V případě splnění podmínek následuje víceúrovňové větvení programu a rozhodnutí, o jaký typ součástky se jedná. Podle typu součástky se provede její odebrání. V případě, že se jedná o součástky typu kostka nebo hranol, je předání součástky mezi pažemi provedeno přímo. Pro přehlednost je tato situace znázorněna na obrázku 4.12.



Obrázek 4.12 - Přímé předání součástky mezi pažemi

V případě součástky typu trojhran je předání rozděleno na dvě části. V první levá paže přemístí součástku na definovanou pozici. Pravá paže pak z této pozice trojhran odebere. Postup je znázorněn na obrázku 4.13.



Obrázek 4.13 - Nepřímé předání součástky typu trojhran

4.4.1 Procedury programu RAPID pro levé rameno

Tato podkapitola je věnována rozboru programu, na jehož základě robot pracuje. Kód lze pro snadné vysvětlení rozdělit na část pro levé a pravé rameno. Každá z těchto částí obsahuje jednotlivé procedury programového kódu, které jsou popsané níže. U vytvořených programů

se přepokládá zapnutý cyklický režim provádění. Následuje popis procedur programu pro levé rameno.

Program začíná voláním procedury *main* ukázané níže, ve které je první instrukcí podmínka pro zjištění, zda se už program nachází v konkrétním cyklu předávání součástek, nebo nastává samotné spuštění programu. Tato podmínka je vyhodnocena na základě datové proměnné *Poprve*, která je standardně nastavená na hodnotu *true*. Podmínka se tedy při spuštění programu vykoná. Následně je proměnné *Poprve* nastavena hodnota *false* a nastává inicializace všech použitých proměnných v běhu programu. Po vykonání tohoto podmíněného větvení je volána procedura *Rsnimek*, která bude probrána níže. Další instrukce je také ve formě podmínky, která ošetřuje případ, kdy na podložce již nejsou žádné součástky. Argumentem této podmínky je vnitřní metoda kamery *CamNumberOfResults*, která vrací počet nalezených součástek na podložce. V případě, kdy *CamNumberOfResults* je rovna nule, program zastavuje obě ramena pomocí příkazu *stop*. Pro stav, kdy je *CamNumberOfResults* větší než nula, je provedeno získání hodnot z kamery pomocí *CamGetResult* a uložení do *DataCamera*. Dále je provedena procedura *RTransformace*, která slouží pro přepočet souřadnic pro uchopení lokalizované součástky a jejíž popis bude probrán níže. Nakonec je provedeno vícenásobné větvení, při kterém se přiřadí do proměnné *S* hodnota, která odpovídá typu lokalizované součástky (1 – Kostka, 2 – Kvádr, 3 – Trojhran) a provede se jedna z následných procedur *RPredani* nebo *RTrojran*. Tyto procedury budou probrány níže.

```
PROC main()
    IF Poprve THEN
      Poprve := FALSE;
      RInicializace;
    ENDIF
    RSnimek;
    IF CamNumberOfResults(Cam1) = 0 THEN
      Stop;
    ELSE
      CamGetResult Cam1, DataCamera;
      RTransformace;
    ENDIF
    IF DataCamera.name = "Kosticka" THEN S:=1; RPredani;ENDIF
    IF DataCamera.name = "Obdelnik" THEN S:=2; RPredani;;ENDIF
    IF DataCamera.name = "Trojuhelnik" THEN S:=3; RTrojhran;ENDIF
  endproc
```

Procedura *RInicializace* při prvním spuštění inicializuje proměnné, kde jsou zaznamenány odebrané díly, proměnné k signalizaci pro spolupráci mezi pažemi robota, kalibrace čelistí efektoru a nahrání vytvořené úlohy do modulu strojového vidění.

```
PROC RInicializace()
    !Počet odebraných dílů
    K:=0;       !Kostek
    O:=0;       !Kvádrů
    T:=0;       !Trojhranů k odebrání
    TP:=0;      !Předaných trojhranů
    PT:=0;      !Povolení odebrání trojhranů

    Tzmena := Tpoloz;              !Proměnná robtarget

    PovoleniPredani := FALSE;      !Signalizuje povolení předání
    PovoleniPusteni := FALSE;      !Signalizuje povolení puštění při předávání
    Pusteno := True;               !Signalizuje manipulaci se součástkou

    g_Calibrate\Jog;               !Kalibrace serva
    g_Init \holdForce:=5;          !Nastavení síly úchopu

    CamSetProgramMode Cam1;                    !Přepnutí kamery do programového
režimu
    CamStartLoadJob Cam1,"PickAndPlace.job";!Nahrání vytvořené úlohy
    CamWaitLoadJob Cam1;                       !Počká na úspěšné nahrání úlohy
    CamSetRunMode Cam1;                !Přepnutí kamery do běhového režimu
ENDPROC
```

Procedura *RSnimek* níže slouží pro zachycení obrazu stavu podložky, se kterým se následně pracuje. Jako první je zde instrukce *MoveJ*, která realizuje přesun integrované kamery nad podložkou. Po nastavení logických hodnot do proměnných *Pusteno*, *PovoleniPredani* a *PovoleniPusteni*, které ošetřují správný postup předání součástek, následuje zpoždění programu instrukcí *WaitTime 1*, pořízení snímku *CamReqImage* a opět zpoždění programu příkazem *WaitTime 1*. Zpoždění je přidáno z důvodu kvality pořízení snímku a jeho případného rozmazání.

```
PROC RSnimek()
    MoveJ pCamera,v200,fine,Camera\WObj:=wobj0;
    Pusteno := TRUE;
    PovoleniPredani := FALSE;
    PovoleniPusteni := FALSE;
    WaitTime 1;
    CamReqImage Cam1;
    WaitTime 1;
```

```
ENDPROC
```

Jako první instrukce v proceduře *RTransformace* je *g_gripOut*, která slouží pro otevření čelistí efektoru. Následně je proměnná *Pretoceni* nastavená na *false*. Tato proměnná slouží ke správnému natočení trojúhelníku při pokládání. Další část kódu procedury je přepočet souřadnic z kamer do souřadného systému robota. Pokud kamera vrátí záporný úhel natočení, úhel je přepočítán do kladného směru a *Pretoceni* je nastaveno na *true* v instrukci *IF*. Přepočet ošetřuje stav, kdy by efektor dosáhl při najíždění do požadovaného targetu své maximální rotace. Při tomto stavu se snaží robot dostat do požadovaného targetu nežádoucím přenastavením své paže. Poté následuje přetočení efektoru do požadované rotace a je naveden nad součástku. Efektor je přemístěn na výšku součástky a tu uchopí. V poslední části procedury se vzdálí od podložky ve vertikálním směru. Pokud pravá paže není připravená, vyčká. Toto vyčkání obstarává instrukce *WaitUntil Pusteno* a nastavení proměnné *Pusteno* na *false*. Pokud je proměnná nastavena na *true*, jsou obě paže připraveny předat si součástku. *Pusteno* je poté nastaveno na false.

```
PROC RTransformace()
    g_gripOut;
    Pretoceni := FALSE;
    ! Nastavení polohy na uchopovaný předmět
    PDataCamera.trans.x := pPodlozka.trans.x+ DataCamera.cframe.trans.x+10;
    PDataCamera.trans.y := pPodlozka.trans.y- DataCamera.cframe.trans.y-9.5;
    PDataCamera.trans.z := pPodlozka.trans.z-50;
    uhel  := EulerZYX(\Z, DataCamera.cframe.rot );

    IF uhel<0 THEN
       uhel := 180 + uhel;
       Pretoceni := TRUE;
    ENDIF

    PDataCamera.rot := OrientZYX( -uhel , 0 , 180);
    MoveL PDataCamera,v200,z5,Servo\WObj:=wobj0;

    PDataCamera.trans.z := 27;
    MoveL PDataCamera,v30,fine,Servo\WObj:=wobj0;
     g_GripIn;
    WaitTime 1;

    PDataCamera.trans.z := 100;
    MoveL PDataCamera,v50,z30,Servo\WObj:=wobj0;
    WaitUntil Pusteno;
    Pusteno := FALSE;
```

```
ENDPROC
```

Procedura *RTrojhran* níže začíná inicializováním proměnných typu robtarget *TZmena* a *TPretoceni*. Proměnná TZmena určuje, kde a v jaké výšce bude trojhran umístěn. Jsou tedy přepočítány souřadnice v ose z pro položení součástky:

(Přepočítaný target).trans.z := (Počáteční target).trans.z + A + (B * C);

Kde:

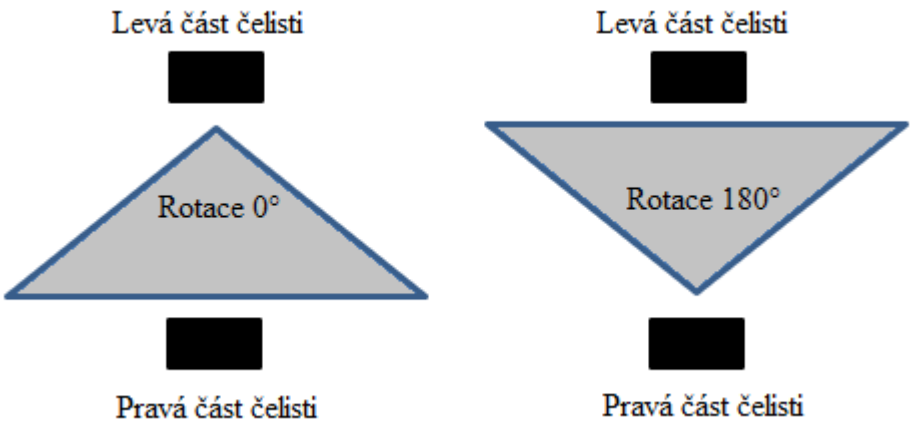
- A – offset,
- B – počet již umístěných součástek,
- C – výška stohované součástky.

Proměnná *TPretocení* slouží jako pomocná proměnná při možném přetočení trojhranu. Dále v instrukci *IF* vyhodnotí, zda byl trojhran zrcadlově přetočený. Tento případ je zobrazen na obrázku 4.14. Vzniká poté, co se přepočte úhel u součástky, která není rovinově (zrcadlově) souměrná. Pokud ano, přeorientuje proměnnou *TPretoceni*, aby při stohování byl trojhran správně umístěn. Následně levá paže uchopený trojhran přemístí na místo předání a položí. S tím je do proměnné *T* přičtena hodnota jedna a signalizováno povolení předání.

```
PROC RTrojhran()
    Tzmena.trans.z := Tpoloz.trans.z + 0 + (T * 15);
    Tzmena.rot := OrientZYX( 180 , 0 , 180);
    TPretoceni.rot := OrientZYX( 180 , 0 , 180);

    IF Pretoceni THEN
      TPretoceni.rot := OrientZYX( 0 , 0 , 180);
      Tzmena.rot := OrientZYX( 0 , 0 , 180);
      Pretoceni := FALSE;
    ENDIF

    MoveJ TPretoceni,v150,fine,Servo\WObj:=wobj0;
    MoveL Tzmena,v30,fine,Servo\WObj:=wobj0;
    g_GripOut;
    WaitTime 1;
    Tzmena.trans.z := Tzmena.trans.z + 40;
    MoveL Tzmena,v40,fine,Servo\WObj:=wobj0;
    T := T + 1;
    PovoleniPredani := TRUE;
  ENDPROC
```



Obrázek 4.14 Situace přetočení součástky

První dvě instrukce procedury *RPredani* navedou levou paži k bodu předání. Poté, co je paže na místě, je signalizováno povolení předání pomocí nastavení proměnné *PovoleniPredani* na hodnotu *TRUE*. Poté paže vyčkává, dokud nedostane signál k uvolnění úchopu. Tedy čeká, dokud druhá paže neuchopí součástku. Poté co součástku pravá paže uchopí, signalizuje přes proměnnou *PovoleniPustení,* kterou nastaví na *TRUE*. Levá paže pokračuje v programu instrukcí *GripOut* a signalizuje puštění. Tím je procedura ukončena.

```
PROC RPredani()
    MoveJ Kpredani10,v200,z200,Servo\WObj:=wobj0;
    MoveJ Kpredani,v200,fine,Servo\WObj:=wobj0;
    PovoleniPredani := TRUE;
    WaitUntil PovoleniPusteni;
    g_GripOut;
    Pusteno := TRUE;
    WaitTime 1;
ENDPROC
```

4.4.2 Procedury programu RAPID pro pravé rameno

Zde jsou popsány jednotlivé procedury programu pravé paže. Jak již bylo v kapitole 4.4.1 napsáno, u programu je předpokládán zapnutý cyklický režim provádění.

```
PROC main()
    IF Poprve THEN
        Poprve := FALSE;
        RInicializace;
    ENDIF
```

```
    MoveJ Predani10,v200,fine,Servo\WObj:=wobj0;
    g_GripOut;

    WaitUntil PovoleniPredani or (PT = 1 and T > 0);
    IF S = 1 THEN
       RKostka;
    ELSEIF S = 2 THEN
       RKvadr;
    ELSEIF PT = 1 or S = 3 THEN
       WHILE Pusteno = TRUE and PT = 1 and T > 0 DO
         Pusteno := False;
         RTrojhran;
       ENDWHILE
    ENDIF
    S := 0;
    Pusteno := TRUE;
  ENDPROC
```

Obdobně jako u programu levé paže začíná procedura *Main* instrukcí *IF*, která, pokud je program spuštěn poprvé, spustí proceduru *RInicializovat*. Procedura *RInicializovat* je obdobná proceduře z levé paže s rozdílem, že inicializace programu pravé paže neobsahuje inicializaci modulu programového vidění. Dále se paže přemístí na pozici *Kpredani10* a otevře čelisti efektoru. *Kpredani10* je prvotní pozice, ze které se paže přemísťuji k přebrání součástky. Následuje instrukce *WaitUntil*, která čeká, dokud nebude signalizované povolení předávání nebo nebude pravá paže vyzvána z uživatelského rozhraní k odebrání trojhranu, přičemž se musí minimálně jeden trojhran nacházet na předávacím místě. Dále procedura obsahuje vícenásobné větvení, ve kterém se vyhodnotí, jaký typ součástky bude pravá paže přebírat. Pro typ součástek kostka a kvádr následuje pouze procedura pro daný typ součástky. Pro typ součástku trojhran je procedura *RTrojhran* v cyklu *WHILE*. V tomto cyklu odebere pravá paže všechny trojhrany umístěné na předávacím místě, pokud nebude v uživatelském rozhraní změněno povolení odebrání trojúhelníku *PT*.

```
PROC RKostka()
    MoveJ Predani20,v50,fine,Servo\WObj:=wobj0;
    g_GripIn;
    WaitTime 1;
    PovoleniPusteni := TRUE;
    WaitTime 1;
    MoveL Predani10,v100,z40,Servo\WObj:=wobj0;
    MoveJ Kpoloz20,v100,z40,Servo\WObj:=wobj0;
    Kzmena.trans.z := Kpoloz10.trans.z + 0.5 + ((K) * 14.95);
    MoveL Kzmena,v30,fine,Servo\WObj:=wobj0;
```

```
    g_GripOut;
    K := K +1;
    WaitTime 1;
    MoveL Kpoloz20,v100,z30,Servo\WObj:=wobj0;
ENDPROC
```

Procedura RKostka slouží k odebrání součástky a k jejímu stohování. Začíná přemístěním pravé paže na pozici Predani20, kde sevře součástku, kterou zatím svírá levá paže. Po sevření vyčká jednu sekundu a signalizuje povolení uvolnění úchopu levé paži, ta otevře své čelisti. Nyní je součástka v držení pravé paže a přemísťuje jí na místo pro stohování. Jsou přepočítány souřadnice v ose z pro položení součástky. Následně je součástka na tuto pozici umístěna. Pravá paže se poté oddálí od položené součástky, tím je procedura ukončena. Shodný postup platí pro součástku typu kvádr, kdy je rozdíl v pozicích umístěné součástky.

```
PROC RTrojhran()
    g_GripOut;
    MoveJ Tnahore,v100,fine,Servo\WObj:=wobj0;
    Tzmena := Tdole;
    Tzmena.trans.z := Tzmena.trans.z + 0 +((T-1) * 15);
    MoveL Tzmena,v50,fine,Servo\WObj:=wobj0;
    g_GripIn;
    T := T-1;
    WaitTime 1;
    MoveL Tnahore,v50,fine,Servo\WObj:=wobj0;
    MoveL Tnahore10,v100,fine,Servo\WObj:=wobj0;
    Tzmena := Tdole10;
    Tzmena.trans.z := Tzmena.trans.z + 0 + (TP * 15);
    MoveL Tzmena,v50,fine,Servo\WObj:=wobj0;
    g_GripOut;
    TP := TP +1;
    WaitTime 1;
    MoveL Tnahore10,v100,fine,Servo\WObj:=wobj0;
ENDPROC
```

Poslední procedurou je *RTrojhran*. Ta se podobně jako procedura *RKostka* stará o přemístění součástky. Prvním krokem je otevření čelistí, ostatní procedury jako první krok tento příkaz nepoužívají. Je to z důvodu, že tato procedura je volána z procedury *Main,* kde je vložena do cyklu *WHILE*. Tento cyklus může proceduru volat opakovaně a je tedy třeba vždy opětovně čelisti otevřít. Dále je pravá paže přesunuta nad součástku, tu poté odebere. Při tom odečte z proměnné *T* jedničku. Poté součástku přemístí a stohuje na určeném místě. Do proměnné *TP* přičte jedna a oddálí se, tím je procedura ukončena.

4.5 GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ

Grafické uživatelské rozhraní poskytuje uživateli interaktivní informační nástroj k pohodlnější a efektivnější práci. Může mít mnoho podob a vykonávat různé funkce. Aplikace může běžet na různých platformách od počítačů a tabletů až po mobilní telefony. Může se jednat o webovou nebo počítačovou aplikaci napsanou v libovolném jazyku. Pak záleží, jak daná aplikace dokáže komunikovat se zdrojem dat.

Způsobů, jak napsat GUI je mnoho, ale možností pro komunikaci s robotem už tolik být nemusí. Je tedy zapotřebí zjistit, jaké jsou možnosti na obou stranách, porovnat je a vybrat tu nejlepší variantu. Ukázkové GUI je vyobrazeno na obrázku 4.15.

Rozdíl mezi webovou a desktopovou aplikací spočívá v tom, že webová aplikace je uživatelům poskytována z webového serveru nebo je uložena na zařízení. K jejich vývoji je využíváno HTML, CSS, PHP, JavaScriptu a nejrůznějších API. Webová aplikace využívá webový prohlížeč v uživatelském zařízení jako klient. Výhodou takové aplikace je možnost spustit ji na libovolném zařízení s libovolným operačním systémem, a při aktualizaci zařízení je jisté, že aplikace bude fungovat stejně a beze změn.

Obrázek 4.15 – Ukázka GUI na FlexPendantu (IRC5 with FlexPendant – Návod k použití, 2018)

Počítačová aplikace spolupracuje přímo se zařízením, nepoužívá klienta. Nevýhodou je, že ne všechny aplikace lze spouštět v různých operačních systémech. Je tedy třeba vědět, na jakých zařízení tato aplikace bude spuštěna.

4.6 NÁVRH UŽIVATELSKÉHO ROZHRANÍ

V této kapitole bude popsán program, který byl vytvořen pro realizaci uživatelského rozhraní aplikace pomocí PC. Aplikace bude obsahovat důležité prvky pro řízení robota, jako je spuštění programu, reset a resetování program pointeru v programu RAPID. Zajímavé části kódu aplikace budou popsány v kapitole 4.6.3 Program. První část kódu obsahuje vysvětlení vizualizace uživatelského rozhraní. Grafická aplikace je vytvořena ve formátu formulářových aplikací WPF. Poté je popsána druhá část obsahující vysvětlení stěžejních částí kódu programu. Program je psaný v jazyku C# s použitím nástrojů PC SDK pro komunikaci s kontroléry robotů ABB.

4.6.1 PC SDK

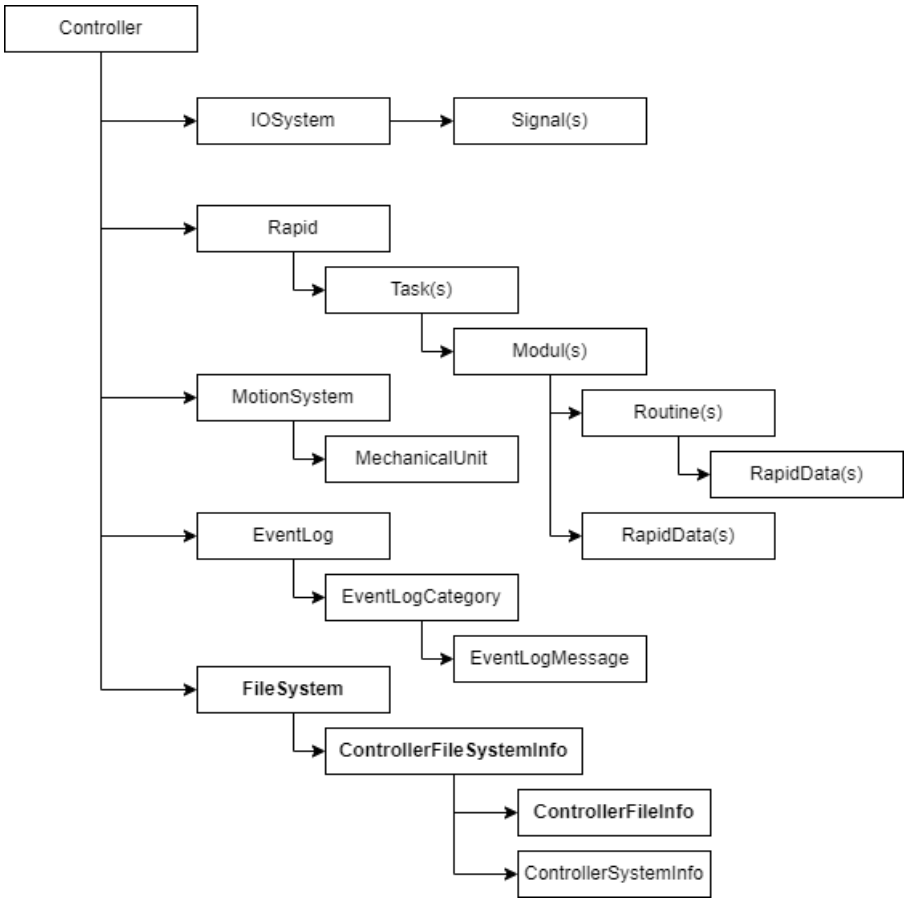
Knihovny aplikačního rozhraní PC SDK pro programovací jazyky C# a Visual Basic jsou rozděleny do tříd následovně:

- Controllers,
- ConfigurationDomain,
- Discovery,
- EventlogDomain,
- FilesystemDomain,
- Hosting,
- IOSystemDomain,
- Messaging,
- MotionDomain,
- RapidDomain,
- UserAuthorizationManagement.

Z tohoto výčtu byly pro aplikaci důležité knihovny Controllers, Discovery, RapidDomain a EventLogDomain.

- Knihovna Controllers disponuje sadou metod a konstruktorů pro obsluhu kontrolérů. Dále umožnuje přístup k jednotlivým instancím připojeného kontroléru. Posloupnost přístupů je na obrázku 4.16.
- Knihovna Discovery obsahuje sadu metod a konstruktorů pro lokalizaci dostupného kontroléru.

- Knihovna RapidDomain umožňuje přístup k datům RAPIDu. Používá se pro čtení a zápis. Na začátku se musí vytvořit nový objekt RapidDomain s definovanou cestou k proměnné. Vytvořený objekt se po ukončení požadavku musí ukončit.
- Knihovna EventLogDomain dává přístup ke třídě EventLogMessages, ta zprostředkovává příjímání informací o stavu kontroléru, RAPIDu a dalších událostí. Přijatá zpráva obsahuje typ události, čas události, název události a případně další sdělení.

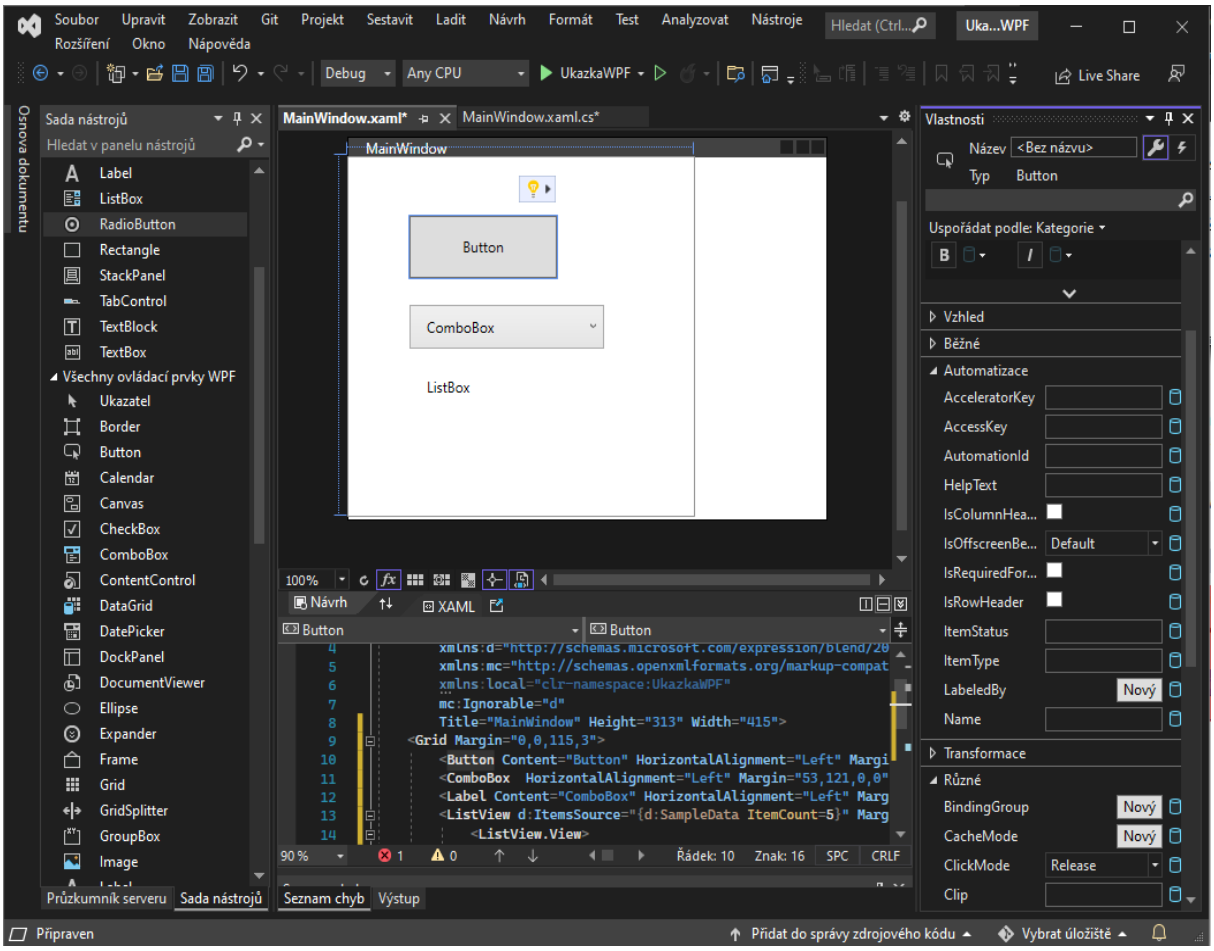


Obrázek 4.16 – CAPI, přístup k instancím

4.6.2 Formulářová aplikace WPF

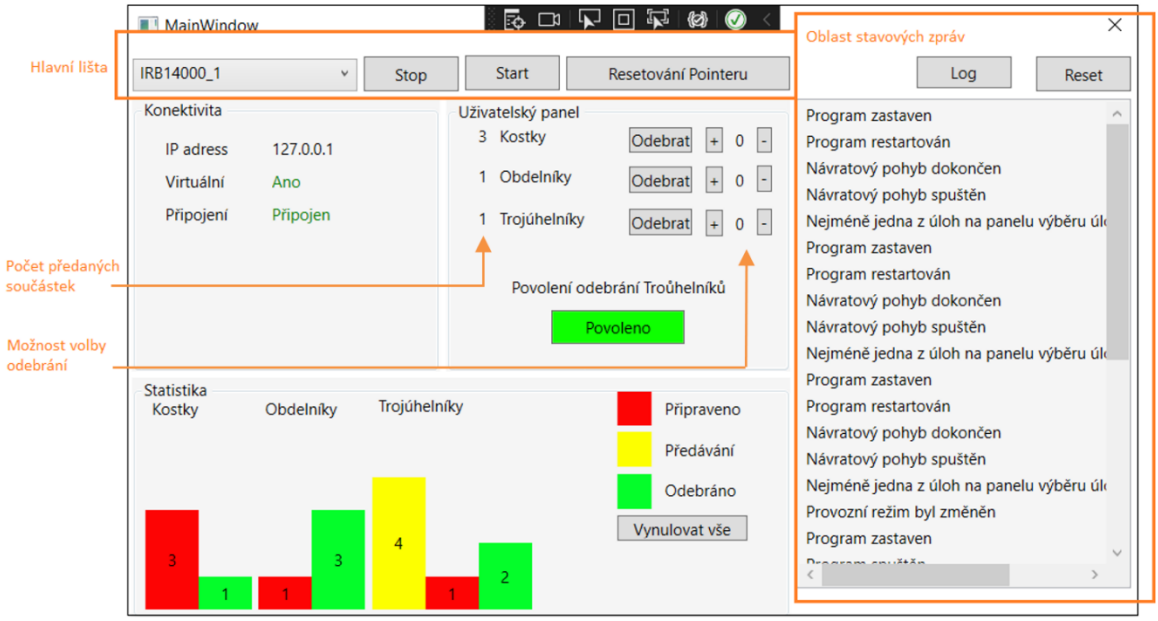
WPF je framework pro komplexní tvorbu formulářových aplikací pro PC, skládá se z graficky navržené aplikace popsané kódem XAML a programem v jazyce C#, nebo Visual Basic. WPF je založená na platformě komponentové architektury. Nabízí tedy řadu již vytvořených komponent, ze kterých lze vytvořit formulářovou aplikaci. Pokud tyto komponenty nestačí, je možno vytvořit nové. Aplikace WPF a uživatelské prostředí Visual

Studio s grafickým návrhářem s ukázkou aplikace je zobrazen na obrázku 4.17. Uprostřed je vyobrazen náhled vytvářené aplikace. Pod tímto náhledem je zobrazen kód XAML, který aplikaci popisuje. Na levé straně je okno se sadou dostupných komponent. Po vložení komponenty do okna WPF je možno upravovat parametry buď v kódu XAML, nebo ve vlastnostech objektu zobrazené v pravé části.



Obrázek 4.17 – Aplikace WPF v grafickém návrháři

4.6.3 Prvky uživatelského rozhraní aplikace



Obrázek 4.18 – Vytvořené uživatelské rozhraní

Uživatelské rozhraní je tvořeno jedním okenním formulářem, zobrazeným na obrázku 4.18. Tento formulář lze rozdělit do následujících sekcí: hlavní lišta, konektivita, uživatelský panel, statistika a sekce pro stavové zprávy.

Sekce hlavní lišty se skládá z jednoho kombinovaného pole pro vyhledání kontrolérů. Následují tři tlačítka s funkcí okamžitého pozastavení běhu programu (tlačítko Stop), spuštění programu (tlačítko Start) a resetování ukazatele programu (tlačítko Resetování Pointeru).

Pod hlavní lištou je rezervován prostor pro informace o připojení. Veškerou uživatelskou kontrolu za běhu a předpokladu správné funkčnosti obstarává sekce Uživatelský panel a částečně i sekce Statistika. V této sekci je v první řadě zobrazen aktuální počet předaných součástek a jejich typ. Dále je zde k jednotlivým typům možnost volby odebrání požadovaného množství již předaných součástek robotem. V poslední řadě je zde speciální povolení pro typ trojúhelníku ve formě tlačítka. Stiskem tohoto tlačítka lze automaticky povolit/zakázat odebírání součástky tvaru trojúhelníku.

V sekci Statistika je poté zobrazen počet připravených součástek k odebrání po přemístění robotem (červený sloupec a popis Připraveno) a počet již odebraných součástek po přemístění robotem (zelený sloupec a popis Odebráno). V případě typu trojúhelník je zde také i žlutý sloupec s popisem Předávání. Tento ukazatel se uplatní v případě, kdy je zakázáno automatické odebírání součástek typu trojúhelník a samotné součástky tím pádem zůstávají v mezikroku přebírání od jednoho robotického ramene k druhému. Ukazatel tedy indikuje, kolik součástek čeká ve frontě k převzetí od druhého robotického ramene. Posledním prvkem

v této oblasti je tlačítko: Vynulovat vše. Stiskem tohoto tlačítka uživatel docílí nastavení evidovaných statistických hodnot a hodnot součástek, které jsou připraveny k odebrání na nulu. Stisknutí tak odpovídá ději, kdy uživatel chce provést odebrání součástek všech typů a začít novou evidenci v sekci Statistika.

Poslední sekce okenního formuláře slouží pro zobrazení aktuálních stavových zpráv. Pro export stavových zpráv nebo jejich reset jsou zde tlačítka: Log a Reset.

4.6.4 Struktura programu

V této kapitole jsou popsány stěžejní části kódu v jazyku C#, vytvořená aplikace je součástí přílohy A. Kód lze rozdělit na dvě části, hlavní blok programu *MainWindow* a třídu *ControllerData*. Nejprve proběhne hlavní metoda MainWindow, poté jsou metody volány událostmi vyvolanými prvky v aplikaci popsané v kapitole 4.6.3. Program obsahuje devět následujících globálních proměnných, stěžejných pro chod program:

```
//Proměnné pro data z jazyku RAPID
public int K;     //Předaných kostek
public int O;     //předaných kvádrů
public int T;     //Trojhranů k odebrání
public int Tm;    //Předaných trojhranů
//Pomocné proměnné pro odebírání
public int Ko;    //Odebrat kostek
public int Oo;    //Odebrat kvádrů
public int To;    //Odebrat trojhranů
//Odebraných celkem
public int Kp;    //Odebraných kostek
public int Op;    //Odebraných kvádrů
public int Tp;    //Odebraných trojhranů
```

Program začíná voláním metody *MainWindou* ukázané níže, ve které dochází k inicializaci aplikace WPF, prvotního hledání dostupných kontrolérů, a nastavení nových časovačů. Postup hledání dostupných kontrolérů spočívá v založení nového objektu třídy *ControllerData* s identifikátorem *this*. Následuje přiřazení zdroje dat pro kombinované pole *DataContext*, do kterého se budou dostupné kontroléry zapisovat, a volání metody *NajdiKontroler*. Metoda *NajdiKontroler* je popsána u třídy *ControllerData*. Další částí je inicializace časovače skládající se z přiřazení události vyvolávající metodu a definování časového intervalu jednoho vykonání.

```
public MainWindow()
    {
        //inicializace WPF
        InitializeComponent();
        //inicializace kotroléru
        ControllerData = new ControllerData(this);
        DataContext = this;                          //Zdroj dat pro ComboBox
        ControllerData.NajdiKontroler();
        //inicializace časovače
        TimerHodnota.Tick += new EventHandler(TimerHodnota_Tick);
        TimerHodnota.Interval = new TimeSpan(0, 0 ,0, 0, 500);     //500 ms
        TimerStav.Tick += new EventHandler(TimerStav_Tick);
        TimerStav.Interval = new TimeSpan(0, 0, 0, 2);             //2 s
    }
```

Třída *ControllerData*, která je použita již v metodě *MainWindow*, obsahuje objekty pro správu dostupných kontrolérů a metodu *NajdiKontroler*. Ve třidě je již nastaven identifikátor *Guid*. Dále se zde nastavují objekty *Kontrolery* (Vyhledané kontroléry) a *VybranyKontroler* (Připojený kontrolér). Při volání metody *NajdiKontroler* je nejprve založen nový skener pro vyhledání dostupných kontrolérů. Skener je následně spuštěn metodou *Scan*, nalezené kontroléry jsou nastaveny a propsány do kombinovaného pole pro zobrazení nalezených kontrolerů.

```
public class ControllerData
  {
    public MainWindow Guid { get; set; }
    public ControllerData(MainWindow guid)
    {
      Guid = guid;
    }
    public ControllerInfoCollection Kontrolery { get; set; }
    public Controller VybranyKontroler { get; set; }
    public void NajdiKontroler()
    {
      var scanner = new NetworkScanner();
      scanner.Scan();
      Kontrolery = scanner.Controllers;
    }
  }
```

Metoda *TimerHodnota_Tick* níže je volaná při zapnutém časovači *TimerHodnota* deklarovaná v metodě *MainWindow*. Časovač je zapnut při připojeném kontroléru, vypíná se při přerušení spojení či jiné chybě. Metoda při každém vykonání načte hodnoty proměnných

(*K, O, T, TP*) z programu RAPID. Příchozí hodnoty jsou datového typu *string*, a před přiřazením do proměnné v C# se přetypují na datový typ *int*. Dále jsou aktualizovány zobrazené hodnoty v uživatelském panelu a statistice.

```
private void TimerHodnota_Tick(object sender, EventArgs e)
    {
        //Vyčtení hodnot z RAPIDu
        K = int.Parse(DejHodnotu("K"));
        O = int.Parse(DejHodnotu("O"));
        T = int.Parse(DejHodnotu("T"));
        Tm = int.Parse(DejHodnotu("TP"));
        //Aktualizovaní počtu v rozhraní
        Kostky.Content = K;
        Kvadry.Content = O;
        Trojuhelniky.Content = Tm;
        //aktualizace grafu
        Graf(K1, K);
        Graf(O1, O);
        Graf(T1, T);
        Graf(T2, Tm);
    }
```

Metoda *TimerStav_Tick* níže je volaná při zapnutém časovači *TimerStav* deklarovaná v metodě *MainWindow*. Časovač je zapnut při detekci chyby související s připojením, aby bylo možno zjistit, kdy opět je kontrolér připojen. Metoda tedy při každém vykonání ověřuje, zda je kontrolér připraven a zda jsou spuštěny motory. Pokud jsou předešlé dvě podmínky splněny, je signalizována opětovná možnost přístupu ke kontroléru a vypsán log-soubor.

```
private void TimerStav_Tick(object sender, EventArgs e)
    {

  try
     {
      if (!ControllerData.VybranyKontroler.Connected)
      {
        LabelConnect.Content = "Nepřipojeno";
      }
      else
      {
       if (ControllerData.VybranyKontroler.State == ControllerState.MotorsOn)
          {
             LabelConnect.Content = "Připojeno";
             TimerStav.Stop();
             VypisLog();
```

```
            }
        }
       }
      catch (Exception)
       {
        ControllerData.VybranyKontroler.Dispose();
        ChybovaHlaska("Kontolér odpojen, resetujte aplikaci");
        LabelConnect.Content = "Nutný reset aplikace";
       }
      }
```

Metoda *ComboBoxController_SelectionChanged* níže je událost volaná po vybrání kontroléru z kombinovaného pole. Začíná vytvořením odkazu odesílatele, v tomto případě je odesílatel události *ComboBox*. Poté se přiřadí k objektu *VybranyKontroler* data zvoleného kontroléru a je vytvořen objekt s přihlašovacími údaji. Poté je metodou *Logon()* odeslán příkaz s přihlašovacími údaji k přihlášení zvoleného kontroléru a spuštěn časovač *TimerHodnota*. Následně je vyplněna tabulka konektivity. V závěru se vyprázdní log-soubor se dřívějšími událostmi.

```
        private void ComboBoxController_SelectionChanged
                (object sender, SelectionChangedEventArgs e)
                {
              //Připojení k vybranému kontroléru
                var comboBoxControllers = sender as ComboBox;
                ControllerData.VybranyKontroler =
                   Controller.Connect(comboBoxControllers.SelectedItem
                   as ControllerInfo, ConnectionType.Standalone);
                UserInfo Yumi = new UserInfo("Jméno", "Heslo");
                ControllerData.VybranyKontroler.Logon(Yumi);
              //Spuštění časovače a nastavení dat konektivity
                TimerHodnota.Start();
                LabelIPadress.Content                                                    =
ControllerData.VybranyKontroler.IPAddress.ToString();
                if (ControllerData.VybranyKontroler.IsVirtual)
                {
                   LabelVirtual.Content = "Ano";
                }
                else
                {
                   LabelVirtual.Content = "Ne";
                }
                LabelConnect.Content = "Připojen";
                LabelConnect.Foreground = Brushes.Green;
                LabelVirtual.Foreground = Brushes.Green;
                LabelConnect.Foreground = Brushes.Green;
```

```
        //Vyprázdnění log-souboru
            EventLog log = ControllerData.VybranyKontroler.EventLog;
            EventLogCategory cat;
            cat = log.GetCategory(CategoryType.Common);
            cat.Clear();
            cat.Dispose();
        }
```

Metoda ChybovaHlaska níže je volána v některých metodách v případě chyby se vstupním parametrem textu popisující chybu. Začíná přenastavením barev komponent týkající se konektivity a výpisem z log-souboru. Poté spustí časovač *TimerStav* a vypíše chybovou hlášku ze vstupního parametru.

```
    private void ChybovaHlaska(string chyba)
        {
            LabelConnect.Foreground = Brushes.Red;
            LabelVirtual.Foreground = Brushes.Red;
            LabelConnect.Foreground = Brushes.Red;
            VypisLog();
            TimerStav.Start();
            MessageBox.Show(chyba);
        }
```

Metoda *StartRapid* níže je používána ke spuštění aplikace robota, je volána po stisknutí tlačítka Start na hlavní liště aplikace. Kontrolér může ovládat pouze jedna relace. Pro získání možnosti zápisu musí aplikace požádat o udělení tzv. *Mastership*. Aplikace tedy požádá o povolení zápisu, je-li přidělen, provede vnořený kód, v tomto případě spuštění programu RAPID a v případě neaktivovaného časovače *TimerHodnota* je tento časovač spuštěn. Předtím je ověřeno, zda je robot v automatickém režimu. Programová konstrukce *try-catch* je zde umístěn pro zachycení případných výjimek. Neošetřená výjimka při některém z požadavků by způsobila kolaps aplikace, takto jsou ošetřeny všechny požadavky aplikace. Aplikace obsahuje i metodu *StopRapid*, která se používá pro ukončení provádění programu RAPID a je shodná, kromě dvou výjimek, s metodou *StartRapid*. První výjimkou je, že neobsahuje zapnutí časovače *TimerHodnota,* ale jeho vypnutí, a místo metody *Start* používá metodu *Stop* ze stejné třídy *Rapid*. Metoda *StopRapid* je volána po stisknutí tlačítka Stop na hlavní liště aplikace.

```
    private void StartRapid()
        {
        try
```

```
        {
        if (ControllerData.VybranyKontroler.OperatingMode
              == ControllerOperatingMode.Auto)
           {
           using (Mastership m = Mastership.Request
                 (ControllerData.VybranyKontroler.Rapid))
           {
           ControllerData.VybranyKontroler.Rapid.Start(true);
           }
            if (!TimerHodnota.IsEnabled) TimerHodnota.Start();
        }
     }
     catch (Exception)
        {
         ChybovaHlaska("Chyba při spuštění Rapidu");
        }
     }
```

Metoda *ZapisHodnotu* slouží pro zapsání hodnoty do proměnné v RAPIDu. Je volána tlačítky Odebrat na uživatelském panelu a tlačítkem Vynulovat vše. Zápis do proměnné je ukázán na proměnné s datovým typem *Num*. Postup je stejný jako pro jiné datové typy, se kterými RAPID pracuje. Nejdříve je vytvořena instance *RapidData* s deklarací vybraného kontroléru, programu, modulu a proměnné, do které má být zapsána hodnota. Poté aplikace požádá o povolení k zápisu. Následně je proměnná v RAPIDu přepsána na požadovanou hodnotu. Vytvořená instance *RapidData* se musí po použití ukončit. Část metody je opět zapouzdřená v konstrukci *try-catch* z důvodu ošetření výjimek. Pokud se vyskytne výjimka, je zavolána metoda *ChybovaHlaska*.

```
private void ZapisHodnotu(string promena, int hodnota)
     {
        Num x = new Num();
        x.Value = hodnota;
        try
        {
           RapidData rd =
              ControllerData.VybranyKontroler.Rapid.GetRapidData
              ("T_ROB_L", "Module1", promena);
           using (Mastership master = Mastership.Request
              (ControllerData.VybranyKontroler.Rapid))
           {
              rd.Value = x;
           }
           rd.Dispose();
        }
```

```
        catch (Exception)
        {
            ChybovaHlaska("Chyba při zápisu hodnoty do Rapidu");
        }
    }
```

Metoda *DejHodnotu* níže slouží pro vyčtení hodnot proměnných z RAPIDU, je obdobná metodě *ZapisHodnotu* pro zápis hodnoty do programu RAPID, nemusí ale žádat o přidělení *Mastership*.

```
private string DejHodnotu(string promena)
    {
        string x = "0";
        try
        {
            RapidData rd = ControllerData.VybranyKontroler.Rapid.GetRapidData
                ("T_ROB_L", "Module1", promena);
            x = rd.Value.ToString();
            rd.Dispose();
        }
        catch (Exception)
        {
            ChybovaHlaska("Chyba při čtění hodnoty z Rapidu");
            TimerHodnota.Stop();
            StopRapid();

        }
        return x;
    }
```

Metoda *ButtonReset_Click* slouží k resetování připojeného kontroléru. Metoda je spuštěna událostí vyvolané kliknutím na tlačítko Reset. K vyvolání restartu je zapotřebí držet dvě povolení *Mastership*, a to přístup k RAPIDu a ke kontroléru. Před samotným restartem se vypne časovač *TimerHodnota*, následně se provede restart a zapne se časovač *TimerStav*. Pokud se v průběhu restartu objeví výjimka, vypíše se chybová hláška s textem.

```
private void ButtonReset_Click(object sender, RoutedEventArgs e)
    {
    try
    {
        if (ControllerData.VybranyKontroler.OperatingMode ==
                ControllerOperatingMode.Auto)
        {
```

```
            using (Mastership m = Mastership.Request
                  (ControllerData.VybranyKontroler.Rapid))
            {
               using (Mastership mm = Mastership.Request
                     (ControllerData.VybranyKontroler.Configuration))
               {
               TimerHodnota.Stop();
               ControllerData.VybranyKontroler.Restart
                        (ControllerStartMode.Warm);
               TimerStav.Start();
               }
            }
         }
      }
      catch (Exception)
      {
         ChybovaHlaska("Chyba při restartu");
         }
      }
```

Metoda *ButtonResPoint_Click* slouží k resetování program pointeru v programu RAPID. Metoda je spuštěna událostí vyvolané kliknutím na tlačítko Resetování Pointeru. Program pointer se pro každý program robota resetuje samostatně. Jelikož se při restartu program pointeru provede inicializace v programu RAPID, je případně zakryto tlačítko pro zakázání odebrání trojhranu, a naopak zviditelněno tlačítko pro povolení odebraní trojhranu. Poté se provede zapnutí časovače *TimerHodnota*, pokud byl předtím vypnut. Pokud se při resetování program pointeru vyskytne výjimka, je volána metoda *ChybovaHlaska* s textem o chybě.

```
private void ButtonResPoint_Click(object sender, RoutedEventArgs e)
      {
      StopRapid();
      try
      {
         if (ControllerData.VybranyKontroler.OperatingMode ==
               ControllerOperatingMode.Auto)
         {
            using (Mastership m = Mastership.Request
                  (ControllerData.VybranyKontroler.Rapid))
            {
            ABB.Robotics.Controllers.RapidDomain.Task _task =
                     ControllerData.VybranyKontroler.Rapid.
                     GetTask("T_ROB_L");
            _task.ResetProgramPointer();
            _task = ControllerData.VybranyKontroler.Rapid.
```

```
                GetTask("T_ROB_R");
            _task.ResetProgramPointer();
            }
        ButtonPovTroj.Visibility = Visibility.Visible;
        ButtonZakTroj.Visibility = Visibility.Hidden;
        if (!TimerHodnota.IsEnabled) TimerHodnota.Start();
        }
    }
    catch (Exception)
    {
        ChybovaHlaska("Chyba při spuštění RAPIDu");
    }
    }
```

Metoda *VypisLog* slouží k výpisu událostí z log-souboru. Metoda je volaná z některých metod a také tlačítkem Log v oblasti stavových zpráv. Nejprve se vyprázdní možné staré zprávy v *List1* a zkontroluje se, zda je v robot v automatickém chodu. Poté je vytvořena instance potřebná k vyčtení logu. Instance *EventLog* obsahuje, z jakého logu se mají data vyčíst, obsahuje všechny zprávy od zapnutí kontroléru. Instance *EventLogCategory* obsahuje přístup ke zprávám určené kategorie, v kódu níže je použitá obecná kategorie *Common*, která obsahuje obecné události kontroléru. Tato a další kategorie jsou detailně popsány v (Application manual – PC SDK, 2012). V cyklu *foreach* se poté vyčtou do kolekce *List1* všechny události kontroléru v dané kategorii. Vytvořené instance se nakonec ukončí. Pokud se při resetování program pointeru vyskytne výjimka, je volána metoda *ChybovaHlaska* s textem o chybě.

```
private void VypisLog()
    {
        List1.Items.Clear();
        try
        {
            if (ControllerData.VybranyKontroler.OperatingMode
                == ControllerOperatingMode.Auto)
            {
                EventLog log = ControllerData.VybranyKontroler.EventLog;
                EventLogCategory cat;
                cat = log.GetCategory(CategoryType.Common);
                foreach (EventLogMessage emsg in cat.Messages)
                {
                    List1.Items.Add(emsg.Title);
                }
                log.Dispose();
                cat.Dispose();
            }
        }
```

```
catch (Exception)
{
    MessageBox.Show("Automat není v chodu");
}
```

5 ZÁVĚR

Cílem této diplomové práce bylo vytvoření aplikace strojového vidění s použitím kolaborativního robota ABB YuMi, kde robot odebírá náhodně umístěné předměty z podložky a následně je třídí a přesouvá na určená místa, a vytvoření příslušného uživatelského rozhraní na bázi PC aplikace. V rámci diplomové práce byly realizovány zadané cíle v požadovaném rozsahu. K realizaci strojového vidění k lokalizaci předmětů byly vybrány součástky blíže specifikované v kapitole 4.1. Pro tyto součástky byl nakonfigurován modul strojového vidění robota. Pro tvorbu programu strojového vidění a robota YuMi byl použit software od ABB RobotStudio. Uživatelské rozhraní bylo naprogramováno v prostředí Microsoft Visual Studio a v programovacím jazyce C#. V rámci návrhu strojového vidění bylo využito několik různých způsobů nastavení obrazu, kalibrace a nastavení lokalizačního nástroje PatMax Pattern. Postup vytvoření úlohy pro modul strojového vidění byl rozebrán v kapitole 4.3. Za účelem splnění dílčího úkolu této diplomové práce byly vytvořeny dva programy pro paže robota ABB YuMi popsané v kapitole 4.4. Vytvořené uživatelské prostředí je popsáno v kapitole 4.6.

POUŽITÁ LITERATURA

Application manual – Integrated Vision. 2021. ABB [online]. [cit. 10. 2. 2022]. Dostupné z: https://abb.sluzba.cz/Pages/Public/OmniCoreRoboticsDocumentationRW7/Application%20Equipment%20&%20Accessories/Vision%20Systems/en/3HAC067707-001.pdf

Application manual – PC SDK. 2012. ABB [online]. [cit. 5. 2. 2022]. Dostupné z: https://library.e.abb.com/public/124d6b59313ed85fc125793400410c5b/3HAC036957-en.pdf

Application manual – Robot Web Services. 2020. ABB [online]. [cit. 27. 2. 2022]. Dostupné z: https://developercenter.robotstudio.com/api/rwsApi/index.html

Bezpečná robotika – Bezpečnost ve spolupracujících robotických systémech. 2018. Sick AG [online]. [cit 6.1.2022]. Dostupné z: https://cdn.sick.com/media/docs/9/69/469/whitepaper_safe_robotics_cs_im0080469.pdf

Co je to strojové vidění a jak může pomoci? 2019. Control Engineering [online]. [cit. 12. 3. 2022]. Dostupné z: https://www.vseoprumyslu.cz/automatizace/kontrola-procesu/co-je-to-strojove-videni-a-jak-muze-pomoci.html

CVL 9.0 Vision Tools Guide. 2019. Cognex Corporation [online]. [cit. 10. 2. 2022]. Dostupné z: https://support.cognex.com/docs/cvl_900/EN/cvl_vision_tools_guide.pdf

ČSN EN ISO 10218. 2011. Roboty a robotizované výrobní technologie. Praha: Česká agentura pro standardizaci, 94s.

ČSN EN ISO 12100. 2011. Bezpečnost strojních zařízení – Všeobecné zásady pro konstrukci – Posouzení rizika a snižování rizika. Praha: Česká agentura pro standardizaci, 106s.

ČSN EN ISO 13857. 2021. Bezpečnost strojních zařízení – Bezpečné vzdálenosti k zamezení dosahu do nebezpečných zón horními a dolními končetinami. Praha: Česká agentura pro standardizaci, 32s.

Demystifying Collaborative Industrial Robots. 2020. International Federation of Robotics [online]. [cit. 6.1.2022]. Dostupné z: https://ifr.org/downloads/hidden/IFR_Demystifying_Collaborative_Robots_Update_v03.pdf?utm_source=CleverReach&utm_medium=email&utm_campaign=Paper+Download&utm_content=Mailing_12323895

Havle, O. Strojové vidění I: Principy a charakteristiky. 2008. Automa [online]. [cit. 16. 2. 2022]. Dostupné z: https://www.automa.cz/Aton/FileRepository/pdf_articles/36550.pdf

IRB 14000 YuMi. 2020. ABB [online]. [cit. 12. 1. 2022]. Dostupné z: https://new.abb.com/products/robotics/cs/prumyslove-roboty/yumi.

IRC5 a RobotStudio – Návod k použití. 2018. ABB [online]. [cit. 12. 1. 2022]. Dostupné z: https://abb.sluzba.cz/Pages/Public/IRC5RoboticsDocumentationRW6/Safety%20information/Getting%20started/cz/3HAC027097-014.pdf

KOLÍBAL, Z. 2016. Roboty a robotizované výrobní technologie. Brno: Vysoké učení technické v Brně – nakladatelství VUTIUM, 780s. ISBN 978-80-214-4828-5.

Operating manual – IRB 14000. 2019. ABB [online]. [cit. 15. 1. 2022]. Dostupné z: https://abb.sluzba.cz/Pages/Public/IRC5RoboticsDocumentationRW6/Robots/Collaborative%20Robots/en/3HAC052986-001.pdf

Operating manual – IRC5 with FlexPendant. 2021. ABB [online]. [cit. 17. 2. 2022]. Dostupné z:

https://abb.sluzba.cz/Pages/Public/IRC5RoboticsDocumentationRW6/Robots/Collaborative%20Robots/en/3HAC052986-001.pdf

Product manual – IRB 14000 gripper. 2021. ABB [online]. [cit. 5. 2. 2022]. Dostupné z: https://abb.sluzba.cz/Pages/Public/IRC5RoboticsDocumentationRW6/Robots/Collaborative%20Robots/en/3HAC054949-001.pdf

Technical reference manual, RAPID Instructions, Functions and Data types. 2010. ABB [online]. [cit. 12. 2. 2022]. Dostupné z: https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf

PŘÍLOHY

A – CD

Příloha A

Příloha k diplomové práci

Automatické roztřiďování předmětů robotem ABB YuMi

Bc. Dušan Vašek

CD

Obsah

1 Text diplomové práce ve formátu PDF.

2 Úplný zdrojový kód aplikace v programu RobotStudio, a zdrojový kód pro uživatelské rozhraní.