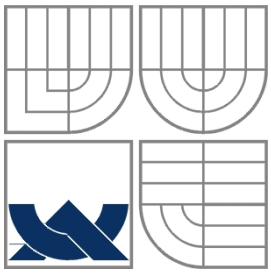
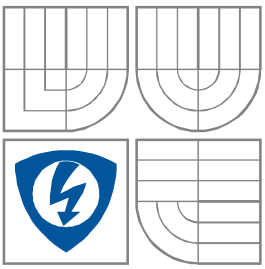


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

DEMONSTRAČNÍ ÚLOHA PRO ROBOTICKÝ MANIPULÁTOR EPSON

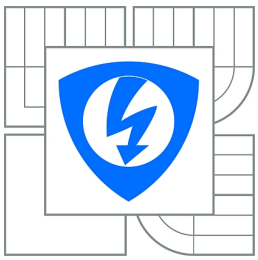
DEMONSTRATION APPLICATION FOR EPSON INDUSTRIAL MANIPULATOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. MARTIN FIREŠ
AUTHOR

VEDOUCÍ PRÁCE doc. Ing. LUDĚK ŽALUD, Ph.D.
SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Martin Fireš	***ID:***	106427
Ročník:	2	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Demonstrační úloha pro robotický manipulátor EPSON

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou průmyslových robotických manipulátorů. Seznamte se s manipulátorem EPSON C3 a jeho programovacím jazykem SPEL+. Navrhněte a realizujte komunikaci mezi řídicí jednotkou manipulátoru a PC v prostředí .NET v jazyce C#. Navrhněte, realizujte a otestujte jednotku pro výběr programu k řídicí jednotce manipulátoru. Po dohodě s vedoucím navrhněte a realizujte jednoduchou demonstrační úlohu pro prezentaci vlastností manipulátoru, přičemž prioritou je bezpečnost.

DOPORUČENÁ LITERATURA:

http://www.robots.epson.com/downloads/brochurefiles/EPSON_C3_6Axis_Robots.pdf
Aleksandar Lazinica and Hiroyuki Kawai, Robot Manipulators New Achievements, InTech, April 2010, ISBN 978-953-307-090-2

Termín zadání:	6.2.2012	***Termín odevzdání:***	21.5.2012

Vedoucí práce: doc. Ing. Luděk Žalud, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Práce se zabývá návrhem a konstrukcí robotického hráče dámy s použitím počítačového vidění, Minimax algoritmu pro výpočet tahu a robotického manipulátoru EPSON C3. Pro komunikaci s robotem byl navržen komunikační protokol a realizována knihovna v jazyce C#.

Klíčová slova

robotický manipulátor EPSON C3, hra dáma, počítačové vidění, Minimax, alfa-beta ořezávání

Abstract

This work deals with design and construction of a robotics checkers player based on computer vision, a move solving algorithm Minimax and a industrial manipulator EPSON C3. Work also deals with design of robot communication protocol and C# communication library execution.

Keywords

Robotics manipulator, EPSON C3, checkers, computer vision, Minimax, alfa-beta pruning

Bibliografická citace:

FIREŠ, M. *Demonstrační úloha pro robotický manipulátor EPSON.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 73 s. Vedoucí diplomové práce doc. Ing. Luděk Žalud, Ph.D..

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma *Demonstrační úloha pro robotický manipulátor EPSON* jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce doc. Ing. Luďkovi Žaludovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce. Dále bych chtěl poděkovat Ing. Karlu Horákovi za účelnou pomoc v oblasti počítačového vidění.

V Brně dne ....

(podpis autora)

OBSAH

1. ÚVOD

Motivací pro vznik diplomové práce bylo ukázat na demonstrační úloze přednosti stacionárního 6-osého manipulátoru EPSON C3. Jako demonstrační úlohu jsem zvolil deskovou hru dáma (anglicky Checkers nebo Draughts). Při strojovém hraní se projevuje přesnost, rychlost a účelné uspořádání mechaniky manipulátoru. Díky možnostem vnějšího řízení lze implementovat strategii hry a snímání scény do nadřazeného PC.

Popisované uspořádání demonstrační úlohy je následující. Člověku, jakožto protihráči robotu, je přiřazena barva kamenů, se kterými hraje. Kameny se dělí na pěšce a dámy, přičemž všechny jsou po celou hru k dispozici na hrací desce. Robot, který má odlišnou barvu kamenů, se snaží nad člověkem vyhrát pomocí vypočítané strategie, která dodržuje platná pravidla dámy. Tahy, které PC vypočítá, se realizují na šachovnici pomocí robotického manipulátoru. Fáze tahu robotu jsou: zvednutí kamene z původní pozice, přesun na novou pozici, položení kamene, odstranění přeskočených kamenů protihráče a přesun manipulátoru do klidové polohy. Celou scénu snímá kamera, aby měl robot přesnou představu o tazích odehraných člověkem a o aktuálních pozicích kamenů na hrací desce. Součástí řídícího programu v PC v prostředí .NET je i vizualizace, která ukazuje aktuální rozložení hry, varovná hlášení a umožňuje potvrzení vykonaného tahu člověkem.

Při konstrukci robotu je nutné vyřešit 3 základní úlohy. Snímání a porozumění scéně právě probíhající hry, strategie hry spojená s výpočtem optimálního tahu robotu a fyzické vykonávání tahu pomocí manipulátoru. Tato práce v následujících kapitolách teoreticky pojednává o možných řešeních všech zmíněných úloh a také o autorem realizovaném řešení.

1.1 SNÍMÁNÁ SCÉNA

Snímací kamera má za úkol zaznamenávat obrazové informace o probíhající hře. Výběr kamery je ovlivněn zvolenou pozicí a uchycením. Jako výhodnější se jeví pevné umístění vůči hrací ploše, tedy nikoliv na pohyblivé části manipulátoru. Při uvažování známé pozice hrací plochy v prostoru manipulátoru, lze na základě porozumění scény přesně určit pozici jednotlivých kamenů.

Na základě zvolené pozice kamery se vybírá konkrétní typ s vhodnou ohniskovou vzdáleností tak, aby kamera snímala ostře celou scénu. Velký důraz se musí klást i na kvalitní nasvícení scény. Světlo by mělo být měkké, aby se nevytvářely odlesky a stíny, mít dostatečnou intenzitu při různých okolních světelných podmínkách a vhodné umístění.

1.2 STRATEGIE HRY

Počítačový program, který hru obsluhuje, má za úkol si udržovat model právě probíhající hry. Na základě platných pravidel usuzuje o správnosti zahraných tahů. Součástí programu je i algoritmus, který hledá vhodný protitah robotu. Hra je modulárnější, navrhne-li se strategie na relativní tahy. Tím se rozumí, že nový tah je odvozen pouze z aktuálního rozložení kamenů na šachovnici a není ovlivněn tahy předchozími. Díky tomu lze hru začít z libovolného rozložení a trénovat tak různé taktiky hry.

1.3 ŘÍZENÍ MANIPULÁTORU

K řízení robotu slouží řídící jednotka EPSON RC180. Elektronická část jednotky v sobě slučuje napájecí a řídící část pro všechny motory os, snímání absolutní polohy všech os, hlídání havarijních stavů či modul digitálních vstupů a výstupů. Jednotka RC180 je vlastně PLC, které obstarává funkce řízení os, výpočtu přímé a inverzní kinematiky a dynamiky a veškeré bezpečností funkce pro robustní řízení. Zároveň však poskytuje pomocí programovacího jazyka SPEL+ volné programování a tím i ovládání všech výše zmíněných funkcí. Díky komunikačnímu rozhraní USB a Ethernet je umožněno jednotku programovat, monitorovat funkci a případně vytvořit komunikační kanál pro vnější řízení.

Řídit jednotku a potažmo celý robot pomocí nadřazeného PC má mnohé výhody. Na PC se totiž lépe implementuje složitá strategie řízení robotu, různé modely prostředí, nasazení umělé inteligence, počítačové vidění či vizualizovaná komunikace s člověkem podpořena zvukovým výstupem. Pro tyto účely je vhodné vytvořit komunikační rozhraní a protokol jak na straně řídící jednotky, tak na straně PC. Na straně PC se jeví jako velmi výhodné implementovat rozhraní ve formě knihovny v programovacím jazyce C#. Jazyk je postaven na platformě .NET Framework, což díky objektově orientovaném přístupu a velkému množství již hotových funkcí přináší značné zjednodušení a zrychlení dalšího nasazení robotu.

2. TEORETICKÁ ČÁST

2.1 PRAVIDLA HRY

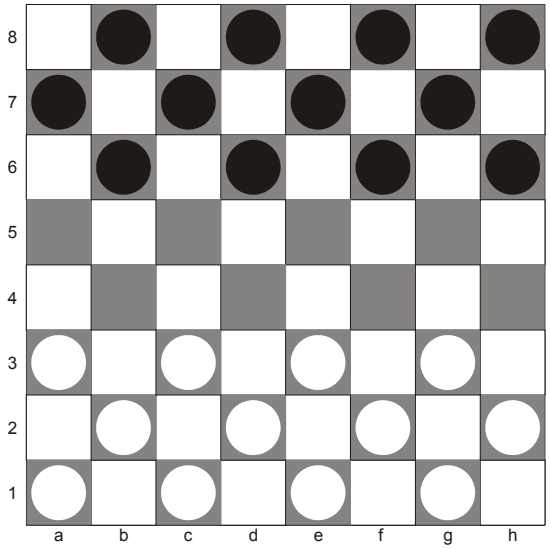
Dáma je mezinárodní desková tahová hra, která má několik národních úprav. Anglická dáma se nazývá English draughts, americký název pro shodnou hru je American checkers. Pravidla české dámy se od té anglické nepatrně liší. Mezinárodní varianta hry se nazývá international draughts nebo international checkers. Charakteristické rysy této úpravy jsou jiná velikost hrací desky a počet kamenů ve hře.

2.1.1 Pravidla české dámy

Podle České Unie Dámy platí pro hru následující pravidla (citováno dle [6]).

2.1.1.1 Hrací plocha

- Česká dáma se hraje na čtvercové desce rozdělené na 64 stejných polí, přičemž se pravidelně střídají světlá a tmavá pole.
- Česká dáma se hraje jen na 32 aktivních tmavých polích
- Šikmé linie vytvořené tmavými poli vytvářejí 13 diagonál. Nejdelší úhlopříčka spojující dva rohy desky a obsahující 8 tmavých polí se nazývá hlavní diagonála.
- Desku je třeba umístit mezi oba hráče tak, aby hlavní diagonála začínala po levé ruce každého hráče, první pole vlevo je u každého hráče tmavé.
- Česká dáma se hraje s dvanácti bílými kameny (nebo kameny světlých odstínů) a dvanácti černými kameny (nebo kameny tmavých odstínů)



Obr. 1 - Hrací deska pro českou i anglickou dámu s výchozím rozestavěním kamenů

2.1.1.2 Tahy figur

- První tah provádí hráč, který má bílé kameny. Oba hráči pak provádí tahy střídavě, vždy jen jeden tah svými vlastními figurami.
- Kamen se pohybuje vždy jen dopředu po diagonále z jednoho pole na druhé volné pole další řady.
- Dámou se stane kámen, který dokončil tah na některém z tmavých polí protilehlé základní řady desky.
- Dáma může uplatnit zvláštní způsoby pohybu a braní až poté, co po její přeměně z kamene zahraje soupeř jeden tah.
- Dáma se přesunuje dozadu a dopředu na libovolná volná pole diagonály, na které se nachází.
- Přemístění figury se považuje za dokončené v okamžiku, jakmile hráč pustil figuru z ruky.
- Jestliže hráč figuru, kterou provádí tah, ještě nepustil, může ji umístit na jiné pole, pokud je to možné podle pravidel.

2.1.1.3 Braní figur

- Braní soupeřovy figury je povinné, kamenem se provádí jen dopředu, dámou lze brát i dozadu. Úplné braní se považuje za jeden tah. Braní vlastních figur je zakázané.
- *Braní kamenem*: Jestliže se kámen nachází na diagonále v sousedství soupeřovy figury, za kterou je volné pole, je povinen ji přeskočit, obsadit toto volné pole a odstranit přeskočenou soupeřovu figuru z desky.
- *Braní dámou*: Jestliže se dáma nachází na stejné diagonále se soupeřovou figurou, za kterou je jedno nebo více volných polí, je povinna tuto figuru přeskočit, zaujmout libovolné volné pole na diagonále za ní a přeskočenou figuru odstranit z desky.
- Braní musí být jasně znázorněno a jednotlivé operace musí být provedeny v pořadí uvedeném v předchozích odstavcích. Nedostatečně znázorněné braní musí být na žádost soupeře znázorněno znovu. Braní se považuje za dokončené až okamžikem odstranění soupeřových figur.
- *Vícenásobné braní kamenem*: Jestliže se kámen po provedeném přeskočení soupeřovy figury opět nachází na diagonále v sousedství soupeřovy figury, za kterou je volné pole, je povinen opat skočit i tuto druhou, případně třetí a další figuru a obsadit volné pole, které se nachází za poslední přeskočenou figurou a poté odstranit z desky všechny přeskočené soupeřovy figury v pořadí, v jakém byly přeskočeny, nebo v pořadí opačném.

- *Vícenásobné braní dámou*: Jestliže se dáma po provedeném přeskočení soupeřovy figury opět nachází na stejné diagonále se soupeřovou figurou, za kterou je jedno nebo více volných polí nebo takovou diagonálu kříží, je povinna opět skočit i tuto druhou, případně třetí a další figuru a obsadit libovolné volné pole, které se nachází za poslední přeskočenou figurou a poté odstranit z desky všechny přeskočené soupeřovy figury v pořadí, v jakém byly přeskočeny, nebo v pořadí opačném.
- Během braní je zakázáno přeskakovat vlastní figury.
- Během braní je dovoleno přesouvat figuru několikrát přes stejné pole, je však zakázáno přeskakovat vícekrát stejnou soupeřovu figuru.
- Vícenásobné braní musí být jasně znázorněno tak, že hráč umisťuje postupně svoji skákající figuru na jednotlivá volná pole za přeskočenými kameny soupeře a nakonec umístí tuto figuru na konečné pole skoku. Nedostatečně znázorněné vícenásobné braní musí být na žádost soupeře znázorněno znovu.
- Vícenásobné přeskočení je ukončeno v okamžiku, kdy hráč pustí přeskakující figuru.
- Odstranění přeskočených figur se považuje za ukončené v okamžiku, kdy hráč odstraní poslední přeskočenou figuru nebo v průběhu odstraňování soupeřových figur ukončí tah.
- Jestliže má hráč více možností braní, může si vybrat libovolnou z nich bez ohledu na počet přeskočených figur. Vícenásobné braní je však třeba dokončit tak, aby po přeskočení soupeřových figur před jejich odstraněním z desky nebyla již pro přeskakující figuru žádná další možnost braní..
- Je-li však možnost braní jak dámou tak i kamenem, má dáma vždy přednost.

2.1.1.4 Vyhodnocení partie

Výhra partie je dosažena hráčem, jehož soupeř je na tahu, ale jehož figury jsou zablokovány a nemůže s nimi provést tah podle pravidel nebo nemá figury.

Partie je *nerozhodná*:

- Když se stejná pozice vyskytne potřetí, přičemž na tahu je vždy stejný hráč.
- Jestliže během 15ti po sobě jdoucích tahů nedošlo k tahu kamenem nebo k žádnému braní, je partie nerozhodná.
- Jestliže nezbývají na desce více než tři dámy proti jedné dámě, která se nenachází na hlavní diagonále, partie je nerozhodná, když oba hráči odehrají každý ještě patnáct tahů.
- Jestliže nezbývají na desku více než dvě dámy nebo jedna dáma s jedním kamenem nebo jedna dáma proti jedné dámě, nebo jestliže nezbývají na desce více než tři dámy proti jedné dámě, která se nachází na hlavní diagonále, partie je nerozhodná, když oba hráči odehrají každý ještě pět tahů

2.1.2 Pravidla anglické/americké dámy

Anglická dáma má téměř totožná pravidla jako ta česká. Jediný rozdíl je v pohybu dámy (označena jako king). Dáma se může obdobně jako pěšec pohybovat pouze o jedno pole vpřed nebo vzad. Aby bylo možné uskutečnit skok a braní spoluhráčovy figury, musí být přímo za braným kamenem volné pole. Na tomto poli skončí hraný kámen dámy. Jestliže má hráč možnost vícenásobného skoku, musí tento tah provést. Remíza se vyhlašuje po 20 tazích bez skoku.

2.1.3 Pravidla mezinárodní dámy

Herní deska mezinárodní dámy se dělí na 10x10 polí, se stejným systémem barevného rozmístění jako u dámy české. Každý hráč má přidanou jednu řadu svých kamenu, tedy začíná s 20 figurami pěšců.

Princip postupu pěšce i dámy se shoduje s českými pravidly. Konec hry není ovlivněn 15ti po sobě jdoucími tahy. Jako remíza se vyhlašuje partie, kdy proti sobě skončí poslední 2 dámy rozdílných barev.

2.2 SNÍMÁNÍ A REPREZENTACE HERNÍ SCÉNY

Celá demonstrační úloha by se dala řešit bez zpětné vazby z hrací desky. Počítač by si neustále udržoval aktuální model rozehrané partie a na základě tahů člověka a počítače by manipulátor hýbal s hracími kameny. Hráč by však nemohl aktivně vstupovat do hry a sám posouvat kameny.

V případě nepřesné informace o poloze kamenů, by docházelo ke kumulované odchylce umisťování. Uchopovací mechanismus manipulátoru by kámen nabral mimo jeho střed a následné položil posunutě vůči požadované pozici. To by mohlo v průběhu hry způsobovat značné obtíže.

Nejefektivnější metodou, jak do úlohy zavést zpětnou vazbu, je snímat scénu pomocí kamery a na základě počítačového vidění reprezentovat získané informace. Alternativní způsob získávání informace o obsazenosti herního pole by spočíval ve využití šachovnice se snímači obsazenosti.

2.2.1 Šachovnice se snímači

Šachovnice s binárními snímači vyniká jednoduchým principem a snadnou reprezentací dat. Použitelná je v případech, kdy není nutná přesná informace o pozici kamene, ale stačí data o obsazenosti pole.

Každé pole obsahuje jeden limitní snímač, který reprezentuje obsazenost. Snímač může být založen na fotoelektrickém jevu, kdy hrací kámen položený na poli odráží světelný paprsek zpět ke snímači. Jiným způsobem může být matice taktilních mechanických spínačů nebo spínačů využívajících kapacitního principu.

Významnou obtíž představuje rozeznávání, zda-li se jedná o kámen typu pěšec nebo dáma. Východiskem by mohlo být z kontextu hry rekonstruovat typ kamene.

Z důvodu výše zmíněných komplikací je vhodnější zvolit metodu s počítačovým viděním.

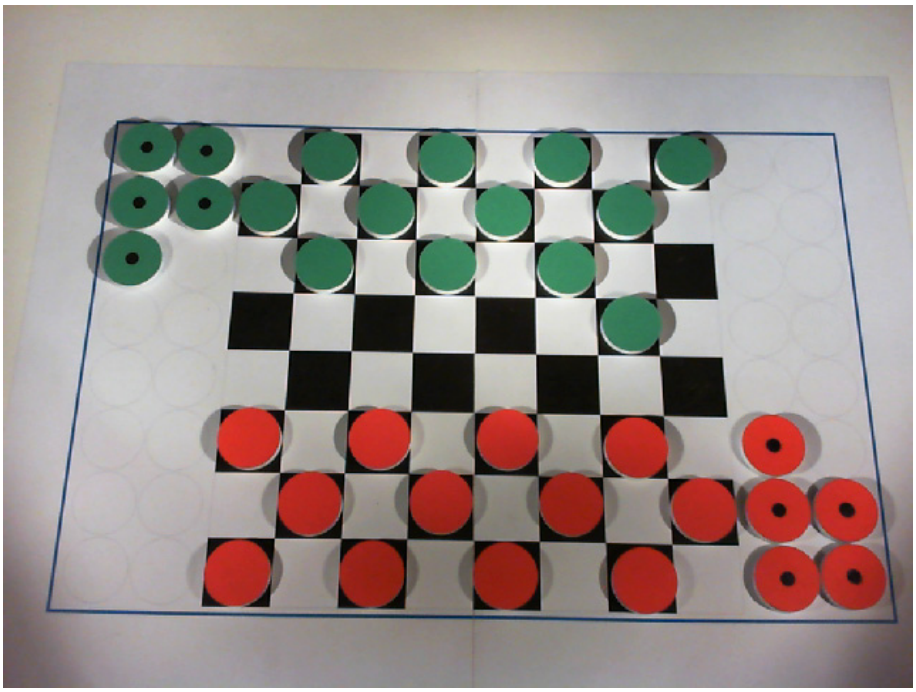
2.2.2 Snímání pomocí kamery

2.2.2.1 Uspořádání herní scény

Umístění kamery musí být takové, že v klidové poloze nesmí rameno manipulátoru clonit ve scéně. Z tohoto požadavku vyplývá, že kamera nemůže být přímo nad šachovnicí a do rozpoznávání obrazu musí být nutně zařazena i kompenzace perspektivy.

Přesná podoba šachovnice a provedení kamenů není dle pravidel pevně stanoveno, což umožňuje přizpůsobit herní materiál tak, aby byl snadno detekovatelný.

Velikost a materiál kamenů souvisí s použitým uchopovacím mechanismem na manipulátoru. Velikosti kamenů je vhodné přizpůsobit i velikost celé šachovnice. Na základě rozměrů výsledné herní desky lze určit vhodnou pozici pro kameru s požadovanou ohniskovou vzdáleností. Na Obr. 2 je vidět jedno z možných provedení herního materiálu.



Obr. 2 - Příklad herní scény

2.2.2.2 Podoba herních kamenů

Standardní bílou a černou barvu kamenů je vhodné nahradit takovým provedením, které nebude splývat s šachovnicí. Podle dále popisované detekce figur se volily barvy co nejvíce vzdálené v barevném spektru.

Výška figur by měla být v kompromisu se snadnou manipulovatelností pro hráče a co nejmenší výškou pro detekci. Příliš vysoká figura, kterou kamera sleduje pod úhlem, se zobrazuje jako opticky zkreslená viz. kapitola 3.2.5.5.

2.2.2.3 Osvětlení scény

Dobré osvětlení scény tvoří základ pro kvalitní snímání. Jsou sice různé metody, jak snímek dodatečně výpočtem rozjasnit, ale za cenu zkreslené barevnosti snímku.

Cílem návrhu je za jakékoliv situace zajistit vyšší intenzitu umělého osvětlení než přirozeného pozadí. Naplněním tohoto cíle se stává scéna nezávislá na vnějším osvětlení, čímž jsou dosaženy rovnoměrné podmínky pro detekci obrazu.

Dalším důležitým parametrem kromě intenzity je i měkkost světla. V případě příliš ostrého, bodového osvětlení, dochází ke vzniku ostrých stínů a přesvětlených

míst. V takových místech se zcela deformuje barevnost obrazu, což může mít značný vliv na funkci rozpoznávání. Mezi metody jak světlo změkčit patří rozptylování přes difuzní filtr, nasvícení pomocné odrazné plochy nebo bodové osvětlení se středem mimo snímanou scénu. Jako podobnou aplikaci ke scéně deskové hry lze uvést nasvětlení uměleckých obrazů, kde se právě s výhodou využívá nepřímého intenzivního osvětlení ze stran obrazu.

2.2.3 Segmentace a detekce objektů v obraze

Na šachovnici se detekují pomocí počítačového vidění 2 hlavní prvky. Kalibrační značky, které slouží pro zaměření a transformaci šachovnice, a dále rozeznávání samotných herních kamenů.

Jedinou možností, jak kameny obou hráčů od sebe rozdělit, je využit jejich různé barevnosti. Následující odstavec 2.2.3.1 popisuje, jakou metodou to lze efektivně provést.

2.2.3.1 Detekce různě barevných objektů v obraze

Protože barvy v obraze, které potřebujeme od sebe rozdělit, jsou prakticky jen 2 (světlé a tmavé kameny), lze využít metodu vzdálených barev ve spektru.

Obraz v počítači se nejčastěji reprezentuje RGB modelem. Pomocí něho se každý bod obrazu popisuje trojicí úrovní červené, zelené a modré složky. Budeme-li v obraze hledat červený objekt, pak bude u bodů, z kterých se objekt skládá, významná úroveň červené složky a značné nižší nebo nulové zastoupení složek zbývajících. Zjištění lze využít a popsat bod s významnou červenou složkou novou úrovní podle vzorce (1).

$$H_{RED} = \frac{L_{RED}}{L_{RED} + L_{GREEN} + L_{BLUE}} \tag{1}$$

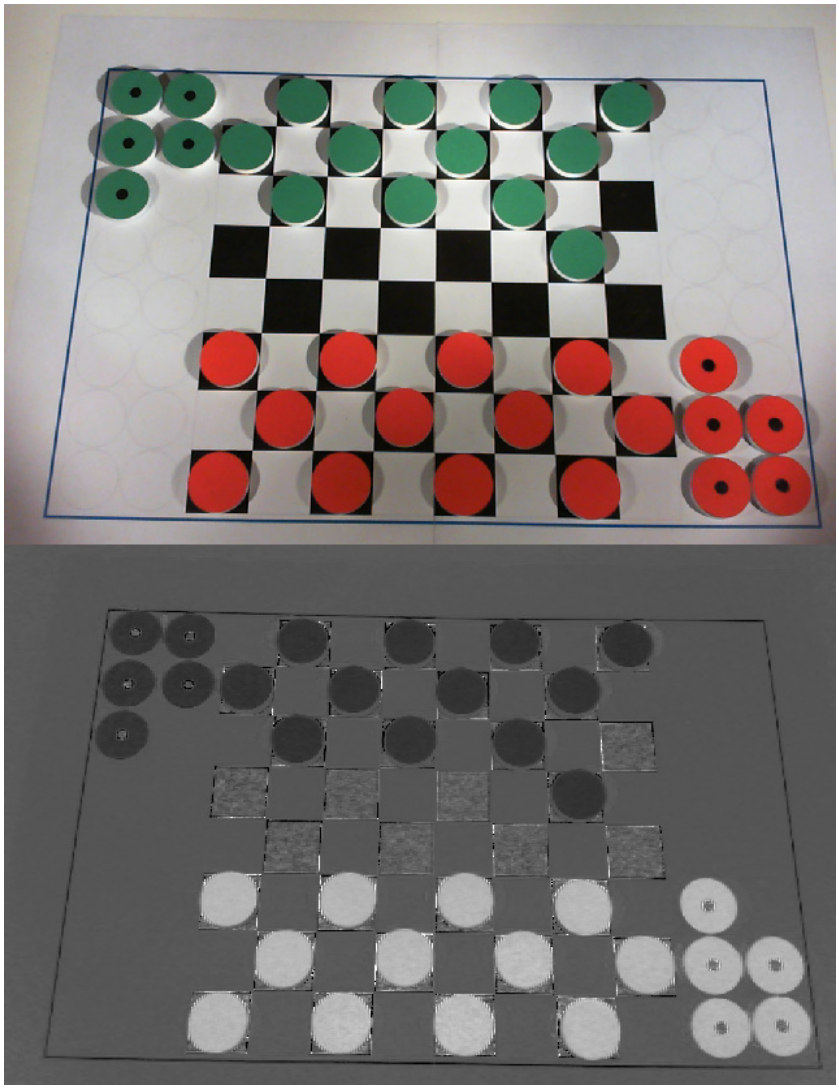
S obměnou pro ostatní barvy lze psát vzorce (2) a (3), kde $L_{RED}, L_{GREEN}, L_{BLUE}$ reprezentuje úrovně jednotlivých složek z původního popisu bodu.

$$H_{GREEN} = \frac{L_{GREEN}}{L_{RED} + L_{GREEN} + L_{BLUE}} \tag{2}$$

$$H_{BLUE} = \frac{L_{BLUE}}{L_{RED} + L_{GREEN} + L_{BLUE}} \tag{3}$$

Jestliže budeme vyšetřovat bod, který se barevně blíží k červené, bude se hodnota H blížit k 1. V případě nízké úrovně červené složky se hodnota blíží k 0. Pomocí metody prahování s vhodně zvolenou hodnotou prahu tak lze od zbytku scény oddělit červené objekty. Stejnou metodou lze vyčlenit i objekty zelené a modré.

Metoda zajišťuje dostatečnou stínovou robustnost. Stín se v RGB modelu projevuje jako snížení jasových úrovni všech složek zároveň. Ve vzorci (1) se tak snižuje čitatel i jmenovatel současně.



Obr. 3 - Mapa červené složky dle (1); bíla -> 1, černá -> 0

2.2.3.2 Detekce pozice šachovnice v obraze

Budeme-li uvažovat teoretickou možnost dokonalé fixace a přesně změřeného úhlu vychýlení kamery vůči kolmici hrací desky, pak by bylo možné vypočítat pozici šachovnice v prostoru bez kalibračních značek. Do výpočtu by musel být zahrnut úhel a vzdálenost vůči kolmici desky a deformační zkreslení obrazu, které mnohdy nebývá mezi údaji výrobce kamery uvedeno. Konstrukce uchycení kamery by musela být velice pevná a i přesto by bylo nutné počítat s otřesy kamery vyvolané velkými setrvačnými silami manipulátoru. Na základě těchto problémů je výhodnější do hrací desky zakomponovat kalibrační značky a vůči nim obraz měřit.

Příkladem různých kalibračních značek jsou záměrné kříže na Obr. 4.

Obr. 4 - Příklady záměrných křížů

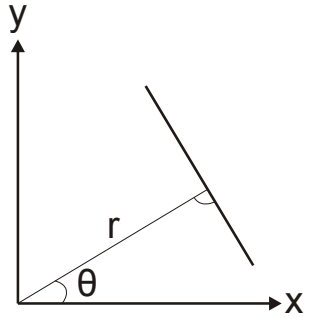
Rozeznávání takových značek v obraze se provádí pomocí rohových detektorů, jako například Moravec nebo Harris/Plessey operátor. Scéna s šachovnicí ale bohužel obsahuje podobných rohů velké množství a tak by se komplikovaně třídily.

Vhodnějším způsobem jak zaměřovat šachovnici je její **orámování** další snadno detekovatelnou barvou. Orámováním šachovnice modrou barvou a následnou filtrací viz (3) získáme čtyřúhelník. Rohy čtyřúhelníku přesně reprezentují pozici šachovnice v prostoru a lze pomocí ní popsat perspektivní zkreslení snímané scény. Rohy čtyřúhelníku hledáme jako průsečíky analyticky popsaných přímek získaných Houghovou transformaci. Přímky je nutné podle směrnicového popisu rozdělit na horní, spodní, levou a pravou hranu čtyřúhelníku a následně mezi nimi vypočítat průsečíky. Použitá metoda vyhledávání přímek má vysokou odolnost vůči šumu a stínům v obraze a je oproti vyhledávání malých kalibračních značek značně robustní. Důvodem robustnosti je skutečnost, že se dlouhé přímky v obraze snadno oddělují od ostatních objektů.

2.2.3.3 Houghova transformace

Jedná se o metodu pro nalezení parametricky popsaných objektů v černobílém obraze. Pomocí transformace lze rozeznávat analyticky popsané přímky, kružnice, elipsy, atd.

V případě scény s šachovnicí hledáme postupně 4 přímky, které procházejí hranami vyfiltrovaného čtyřúhelníku.



Obr. 5 - Parametrický popis přímky polárními souřadnicemi

Metoda funguje na principu hledání co největší shody mezi body v obraze tvořící objekt a analytickým modelem objektu. Hledání přímky v obraze patří mezi nejčastější aplikace transformace. Analytický popis přímky využívá polárního souřadnicového systému. Přímku tak definuje θ úhel natočení normály a vzdálenost r od středu obrazu. V algoritmu se nejprve sestaví nulová matice všech možných úhlů natočení θ a vzdáleností r. Následně se pro každý bod vstupního obrazu zjišťuje, zda-li leží na přímce popsané dvojící (θ_n, r_n). Jestliže leží, prvek matice se souřadnicemi (θ_n, r_n) se inkrementuje o definovaný přírůstek. Po vyhodnocení všech vstupních bodů označují prvky matice s nejvyššími hodnotami přímky, které se v obraze vyskytují s největší pravděpodobností. Jako parametr se metodě předává relativní práh četnosti v rozsahu 0 - 1, který určuje, zda dvojice (θ_n, r_n) může popisovat přímku v obraze.

Pro další zpracování je vhodnější převést přímku v polární reprezentaci na směrnicový popis.

Jako převodní vztah mezi θ ve stupních a velikostí směrnice k lze využít (4).

$$k = \tan\left(\frac{\pi}{180°} \cdot (\theta + 90°)\right) \tag{4}$$

Koeficient q, který pro popis přímky potřebujeme, získáme ze známé dvojice bodů (x, y), kterým přímka prochází (5), a směrnice k.

$$\begin{aligned} x &= \cos\left(\frac{\pi}{180°}\theta\right) \cdot r \\ y &= \sin\left(\frac{\pi}{180°}\theta\right) \cdot r \end{aligned} \tag{5}$$

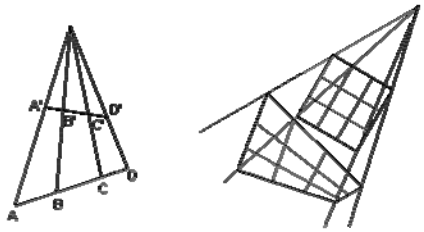
$$q = y - k \cdot x \tag{6}$$

2.2.3.4 Geometrická transformace perspektivního zkreslení

Protože kamera snímá herní desku s šachovnicí pod nenulovým úhlem vůči kolmici desky, dochází při snímání k perspektivnímu zkreslení obrazu. Kalibrační rámeček, který je ve skutečnosti na hrací ploše tvořen obdélníkem, se ve snímaném obraze deformuje na obecný čtyřúhelník. Deformaci lze odstranit vhodnou matematickou transformací za pomoci známého poměru výšky a šířky skutečného obdélníku a pozice 4 rohů zdeformovaného obrazu

2.2.3.5 Kolineární (projektivní) transformace

Jde o transformaci [9], kdy je obraz bodu zobrazen pomocí středového promítání z jedné roviny na druhou. Tato transformace nemá konstantní měřítko, zachovává dvojpoměr (7) v rámci čtveřice bodů ležících na přímce. Přímky se zobrazují opět jako přímky. Přímky rovnoběžné v originále po transformaci směřují do společného bodu.



Obr. 6 - Projektivní transformace

Mezi délkami platí vztahy

$$\frac{AC}{BC} : \frac{AD}{BD} = \frac{A'C'}{B'C'} : \frac{A'D'}{B'D'} = konst \tag{7}$$

Transformační vztah má tvar:

$$X = \frac{ax + by + c}{gx + hy + 1}$$
$$Y = \frac{dx + ey + f}{gx + hy + 1} \tag{8}$$

Pro určení osmi neznámých koeficientů (a,b,c,d,e,f,g,h) (8) je potřeba nejméně 4 identických bodů v obou soustavách, což lze v případě šachovnice splnit.

$$\begin{pmatrix} X_1 \\ Y_1 \\ X_2 \\ Y_2 \\ X_3 \\ Y_3 \\ X_4 \\ Y_4 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -y_1X_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1Y_1 & -y_1Y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -y_2X_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2Y_2 & -y_2Y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -y_3X_3 \\ 0 & 0 & 0 & x_3 & y_2 & 1 & -x_3Y_3 & -y_3Y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -y_4X_4 \\ 0 & 0 & 0 & x_4 & x_4 & 1 & -x_4Y_4 & -y_4Y_4 \end{pmatrix} \bullet \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{pmatrix} \tag{9}$$

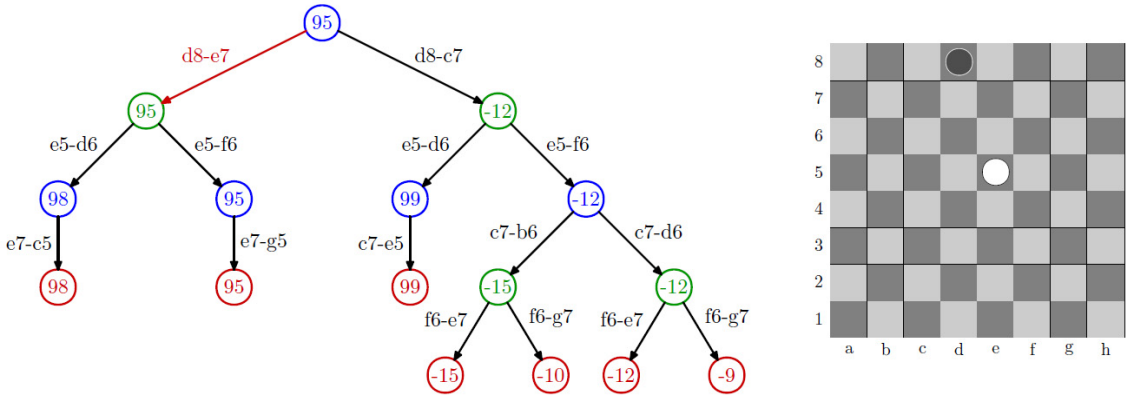
2.3 HERNÍ ALGORITMUS PROTIHRÁČE

Pro účely robotického soupeře v dámě se nelépe hodí algoritmus založený na Minimax strategii nebo strategiích z Minimaxu odvozených. Jedná se o algoritmus používaný pro hraní strategických her mezi dvěma a více hráči. Principem algoritmu je procházení stromu hry a minimalizace maximálních možných ztrát. Algoritmus bývá základem většiny počítačových programů pro hraní her jako je dáma, šachy nebo Go.

2.3.1 Algoritmus Minimax

Typickým úkolem je nalezení nejlepšího tahu v dané pozici. Nutnou součástí algoritmu je funkce, která je schopná ohodnotit libovolnou pozici na hrací desce.

Postup ohodnocování a výběru nejvhodnějšího tahu charakterizuje herní strom, což je druh orientovaného grafu. Hrany herního stromu charakterizují tah a uzly stromu charakterizují pozici hry.



Obr. 7 - Herní strom dle [10]

Algoritmus dle [10][11] projde všechny posloupnosti povolených tahů, které je možné z dané pozice vykonat, a vybere ten tah, který přinese nejvýhodnější pozici. Pro výpočet se používá rekurzivního volání téhož algoritmu. Rekurze a tím i herní strom má předem omezenou hloubku zanoření, aby algoritmus skončil v reálném čase. Podle hloubky se rozděluje obtížnost hráče na různé úrovně, protože s hloubkou zanoření roste i předvídavost hráče. Listy stromu však nemusí být ve stejné hloubce, protože v některé větvi může nastat konec hry.

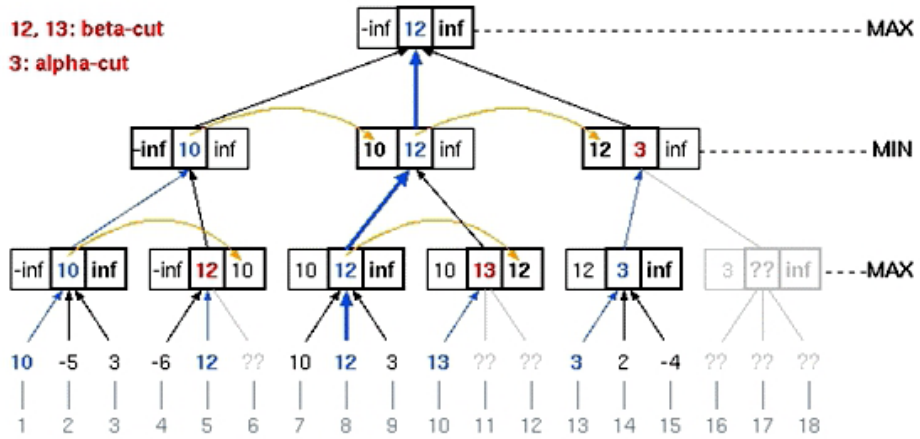
Funkci algoritmu lze demonstrovat na Obr. 7, který ukazuje možnou pozici hry dámy. Úkolem je nalezení nejvhodnějšího tahu pro černý kámen. Kořen stromu obsahuje hodnotu (získanou zpětným šířením), kterou by byla ohodnocena pozice ve hře, do které by se hráči dostali po nevýhodnější posloupnosti tahů pro černý kámen. Červeně je označen přesun černého kamene na pozici e7. Mezi možnými pozicemi e7 a c7 se vybírá ta, která má maximální ohodnocení. Na druhé úrovni jsou možné pozice, z nichž vede další tah protihráč. Ten naopak vyhledává takový tah, který povede k minimálnímu ohodnocení. Kladné ohodnocení značí tah výhodný pro hráče s černým kamenem, záporné hodnocení pro hráče s bílým kamenem. Stejným postupem se pokračuje až do maximální hloubky zanoření. Ze stromu je zřejmé, že nejvýhodnější tah pro černý kámen bude přesun z d8 na e7.

2.3.2 Alfa-beta ořezávání

Problémem algoritmu Minimax je exponenciální časová složitost. V případě stromu s konstantním větvícím faktorem v a hloubkou h je časová složitost v^h. Jako časová optimalizace výpočtu existuje vylepšení pomocí alfa-beta ořezávání.

Princip alfa-beta ořezávání dle [10][11] spočívá v tom, že v některých situacích nemusí Minimax zkoumat další herní pozice, protože je již zřejmé, že nebudou mít na volbu tahu vliv. Algoritmus se rozšiřuje o parametry alfa a beta, které tvoří pro každý uzel minimální a maximální hodnotu.

Účinnost ořezání nepotřebných větví lze zvýšit vhodnou volbou pořadí, v němž jsou procházeny. Nejvýhodnější je projít nejprve ty větve, které by měly dát podle rychlého (ve srovnání s procházení kompletním Minimaxem) odhadu nejlepší výsledky.



Obr. 8 - Minimax s alfa-beta ořezáváním dle [10]

Popsat algoritmus alfa-beta ořezávání lze z Obr. 8. Parametr alfa je u každého ohodnocení uzlu levá hodnota, parametr beta pravá hodnota. Parametry určují meze ohodnocení, při jejichž překročení dochází k oříznutí celé zbývající větve. Budeme uvažovat postupné vyhodnocování zleva doprava. Do parametru beta se ukládá dosud nejvyšší nalezené ohodnocení, do parametru alfa ohodnocení nejnižší. V rámci každé vrstvy a větví patřící k jednomu předchůdci probíhá vyhledávání extrémních ohodnocení. Žluté šipky vyznačují kopírování extrémních ohodnocení do alfa a beta parametrů. Parametry se přenášejí na své následovníky. Alfa ořez lze prakticky demonstrovat na uzlu s červeným ohodnocením 3. Protože následovník předává hodnotu 3 a ta je mimo mez alfa, dochází k ořezu. V reálné situaci to vypadá tak, že hráč vyhodnocuje postupně výhodnost svých tahů. Neustále si udržuje informaci o své dosud nejvýhodnější posloupnosti. Přitom ví, že když se objeví jakákoliv posloupnost s nižším ohodnocením než ta dosud nejvýhodnější, může jí protihráč potencionálně při svém dalším tahu využít. Z toho důvodu celou posloupnost tahů předem vyloučí.

2.3.3 Hodnotící funkce

Vstupem pro hodnotící funkce je ohodnocovaná pozice, výstupem je číslo v rozsahu <-MANY, MANY>. Jedním z možných provedení funkce je sledovat některé významné prvky, které mají vliv na výhru či prohru hráče. Prvky se sledují pro oba hráče odděleně a nakonec se jejich numerická reprezentace od sebe odečte. Bude-li číslo kladné, je tato pozice pro hodnoceného hráče výhodná. V případě nulového

ohodnocení je pozice vyrovnaná pro oba hráče. Hodnotící funkce by měla být rychlá a jednoduchá.

Jako prvky hodnocení lze pro hru dáma uvažovat:

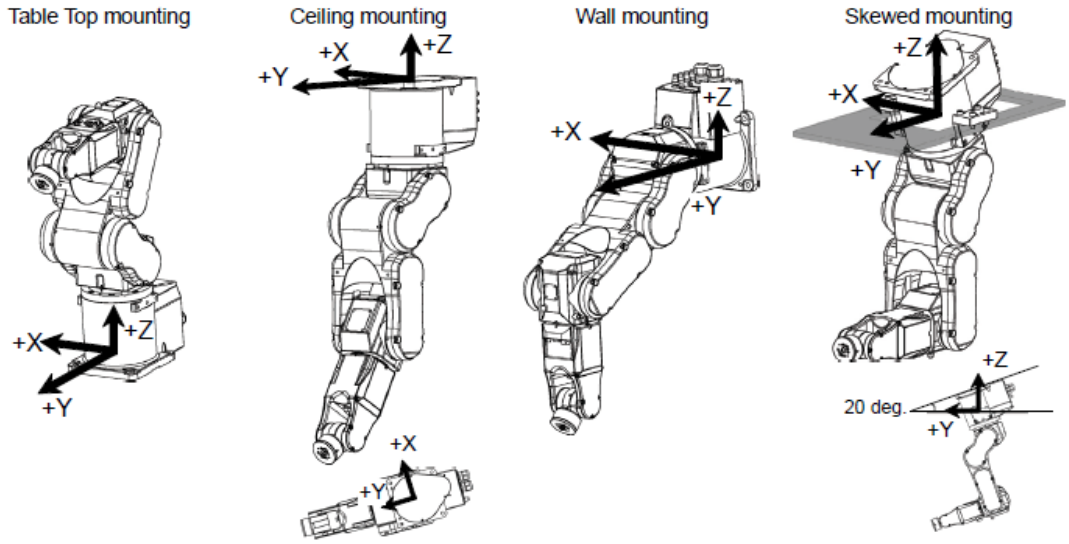
- vážený počet kamenů téže barvy,
- pozici pěšců ve směru k protihráči,
- pozici dam na šachovnici - u krajů a v rozích je dáma méně použitelná,
- bloky figur, osamocené figury atd.

Pomocí změn ohodnocení lze přinutit hráče k větší agresivitě, aktivitě, ochotě dělat výměny a podobně. Lze také vytvořit více hodnotících funkcí pro různé fáze hry.

2.4 ROBOT EPSON C3

2.4.1 Konstrukce robotu

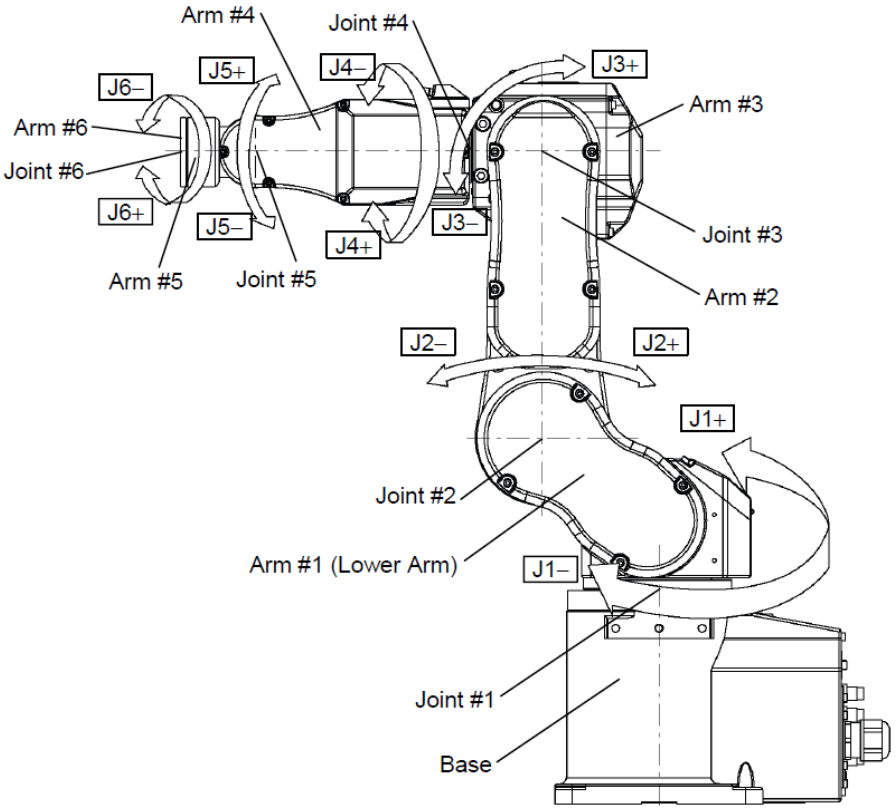
Šestiosý EPSON C3 patří ve své třídě k nejdynamičtěji použitelným robotům, který dosahuje relativní opakovatelnosti ±0.02mm s průměrnou dobou cyklu 0.37s. Universální základ robotu s optimalizovanou konstrukcí z hlediska rozmístění a délek ramen zajišťuje vysokou pohyblivost. Mechanicky vychází koncept robotu ze známého a výhodného uspořádání. Předpokládá se, že hlavní pracovní prostor se nachází kolmo před bází robotu (na ose Y). V tomto směru má robot největší volnost pohybu a největší dosah. Většinou se totiž statickým robotem obsluhují výrobky, které jsou k manipulátoru dopravovány na pásu. Epson C3 však je možné upevnit i ke svislé stěně či případně zavěsit kolmo dolů. V takovém případě se lehkou změnou v programovém nastavení změní mód a báze robotu a pak již lze provozovat robot ve stejném souřadnicovém systému.



Obr. 9 - Možnosti uchycení základny robotu

2.4.1.1 *Jednotlivé osy manipulátoru*

Každá osa je realizována pomocí AC servomotoru, z něhož je hnací síla na osu většinou přenášena pomocí řemenů, případně převodovky s ozubenými koly. Osy jsou vybaveny inkrementálními čidly polohy, díky nimž je měřena absolutní výchylka každé z os. Hodnota inkrementačních pulsů je v programu přístupná jak v čisté podobě, tak v přepočtených úhlech natočení osy.



Obr. 10 - Znázornění os manipulátoru dle [1]

Název a číslo osy	Maximální rozsah ve stupních	Maximální rozsah v inkrementačních pulsech
Joint #1	±170°	±4951609
Joint #2	-160° - +65°	−4660338 - +1893263
Joint #3	-51° - +225°	−1299798 - +5734400
Joint #4	±200°	±4700057
Joint #5	±135°	±3217222
Joint #6	±360°	±6553600

Tab. 1 - Rozsahy natočení jednotlivých os dle [1]

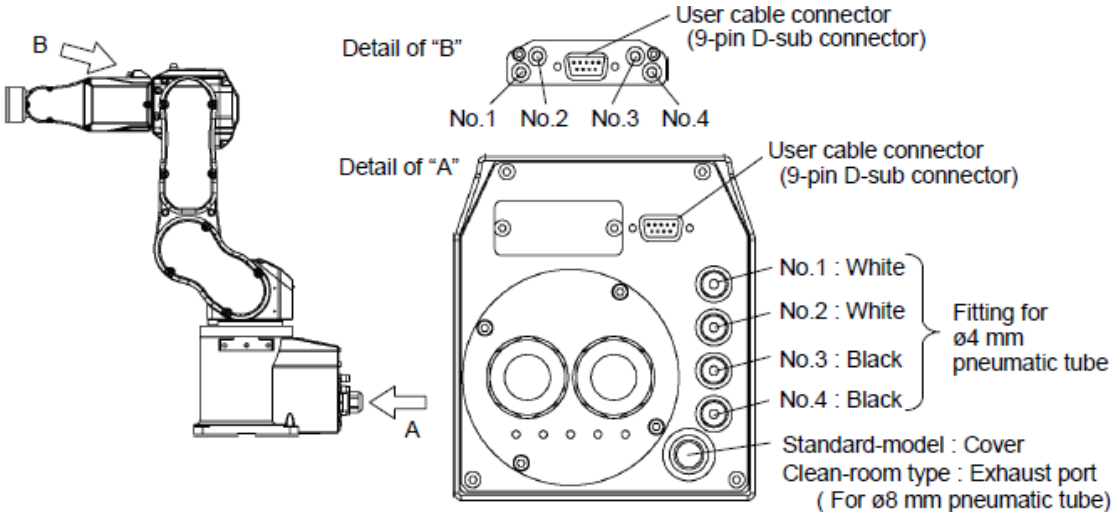
2.4.1.2 *Základní mechanické vlastnosti manipulátoru*

Nominální zatížení manipulátoru je 1kg. V případě, že uvažujeme pohyb s břemenem ve všech osách, je možné maximálně zatížit robot hmotností 3kg. Pouze v případě zavěšení uchopovaného materiálu svisle dolů lze zatížit manipulátor maximální hmotností 5kg. Limity zatížení přímo souvisí s úhlem natočení břemene

vůči kolmici země a pákou, na které je břemeno vyloženo. Při správném popisu zátěže a uložení parametrů do řídící jednotky se automaticky dopočítává maximální zrychlení a zpomalování pohybu, čímž se chrání mechanická část robotu od vysokých dostředivých, odstředivých a setrvačných sil.

V nastavení řídící jednotky lze veškeré parametry týkající se momentu setrvačnosti os (hmotnosti, vzdáleností, tření atd.) upravit. Všechny parametry jsou však od výrobce přednastaveny, včetně absolutního pozicování v prostoru.

Na rameni číslo 4 (mezi kloubem J4 a J5) je umístěna signální LED, která informuje o stavu napájení motorů. V okamžicích, kdy je napájení pro motory vypnuto, nebo při stavu Emergency Stop, je pohyb os zablokován pomocí elektromagnetických brzd, čímž je dosaženo maximální bezpečnosti provozu.

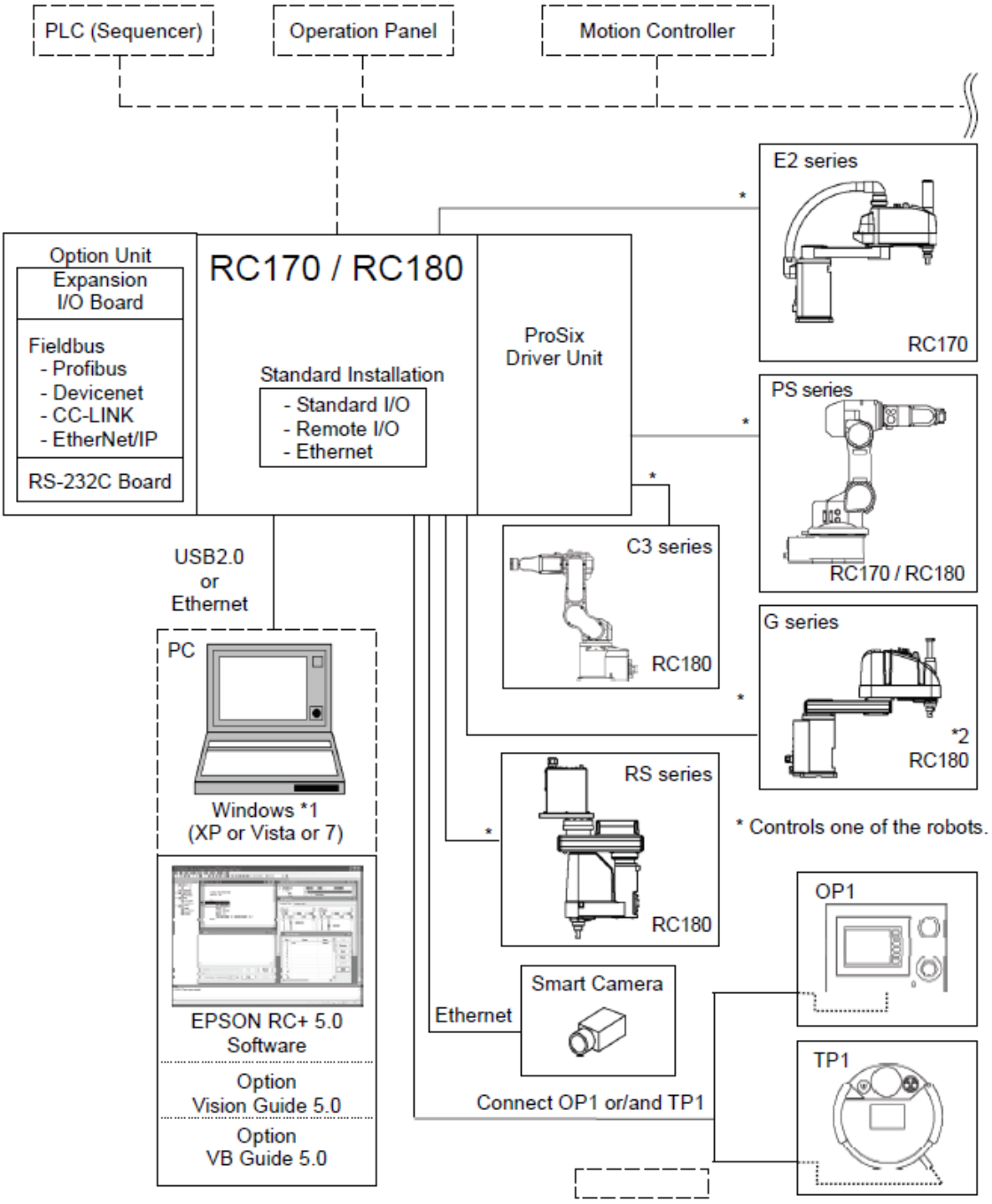


Obr. 11 - Vedení skrz robot využitelné pro uživatelskou aplikaci dle [1]

Skrz celý robot prochází elektrické a pneumatické vedení, které umožňuje uživateli připojení koncových zařízení. Pro účely pneumatického uchopování na principu podtlaku nebo uchopovacích čelistí na principu pneumatických pístů jsou k dispozici 4 hadičky o vnitřním průměru 2.5mm a povoleném přetlaku 0.59 MPa.

Elektrické vedení je tvořeno 9 vodiči zakončenými na obou koncích standardním konektorem Cannon 9Pin. Maximální proudová zatížitelnost každého vodiče je 1A s povoleným AC/DC napětím 30V.

2.5 ŘÍDÍCÍ JEDNOTKA RC180



Obr. 12 - RC180 - možnosti připojení přístrojů dle [1]

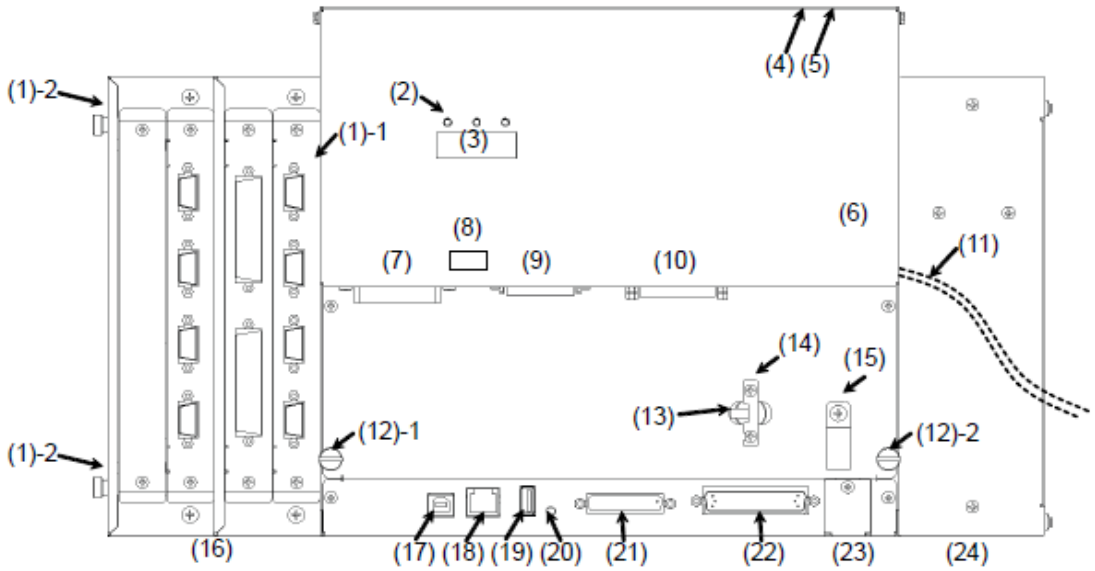
Jedná se o univerzální jednotku, kterou je možné nasadit na většinu manipulátorů od firmy EPSON. Do jednotky byly integrovány funkce, které by mohly být potřebné při běžném průmyslovém nasazení.

Automatizovaný proces lze jednotkou řídit zcela autonomně, kdy není nutné připojení k nadřazenému kontroléru. V takovém případě poskytuje jednotka vykonávání volně programovatelných paralelních programů. Receptury či postupy k samotné činnosti mohou být zavedeny do robotu pomocí USB paměťových médií. Činnost lze monitorovat pomocí připojených operátorských panelů. Pomocí standardních binárních I/O lze jednotkou řídit i okolní technologii spřaženou s robotem, jako například dopravníky, uchopování, bezpečnostní limitní snímače a další. Standardní I/O lze rekonfigurací nastavit k přímému ovládání robotu, tedy k funkci jako Start, Stop, Next, volba programů atd. Pro zajištění maximální bezpečnosti obsahuje jednotka i Emergency Port, jehož funkce není programově upravitelná. Obsahuje signály pro obvody Emergency Stop, koncové bezpečnostní spínače a Reset.

K jednotce lze přidávat i přídavné karty, které obsahují komunikační sběrnice Profibus, DeviceNet, EtherNet/IP, RS-232, další I/O moduly nebo kartu obsahující řízení uchopovacího mechanismu.

Pro vnější řízení se nejčastěji využívá nadřazený PC. Ten může pomocí výše zmíněných komunikačních sběrnic buď posílat pouze souřadnice, nebo robot plně řídit. Firma EPSON má vyvinutý i SW nástroj Vision, který pomocí kamery připojené přes Ethernet, umožňuje řídit průmyslové procesy spojené s počítačovým viděním. Pomocí komunikačních sběrnic USB nebo Ethernet se jednotka programuje a konfiguruje.

2.5.1 Elektronická část jednotky



Obr. 13 - Pohled na jednotku RC180

- Napájení jednotky (11) - konektor pro síťové napětí AC 200-240V.
- Připojení motorů a polohových snímačů - zajištěno pomocí napájecího konektoru (7) a signálního konektoru (9).
- Konektor pro obsluhu Emergency funkcí (10).
- Konektor se standardními vstupy a výstupy (22).
- Rozhraní USB (17) – komunikační rozhraní, programování jednotky.
- Rozhraní USB (19) - slouží k připojení paměťového média s programem nebo recepturou.
- Rozhraní Ethernet – programování jednotky, připojení kamery, nadřazené řízení.
- Status LED bar – informuje o aktuálním módu jednotky– Teach, Auto, Program mode.
- 4x7 segmentový display – zobrazuje číslo vykonávaného řádku uživatelského programu, číslo poruchy robotu nebo status robotu.

Pro zajištění bezpečného provozu zařízení jsou do jednotky implementovány tyto ochranné funkce:

- Signál Emergency Stop – v případě, že se rozpojí kterýkoliv z Emergency kontaktů, přejde jednotka do Emergency módu, kdy se odpojí napájení od všech motorů a sepne elektromagnetická brzda. Pro opuštění Emergency módu je nutné jednotku resetovat.
- Ochranné koncové spínače – po celou dobu rozpojení některého z těchto paralelních signálů robot setrvává na poslední pozici.
- Low Power mód – poskytuje pomalejší mód chodu manipulátoru. Jednotka v tomto módu ignoruje nastavení rychlosti a zrychlení pro jednotlivé osy pohybu a dosazuje jejich snížené hodnoty a také snižuje momentové limity motorů. V případě kolize s vnějším prostředí hrozí menší nebezpečí a fyzické újmy.
- Dynamická brzda – zablokování všech os v případě nebezpečí či odpojeného napětí.
- Detekce poruchy polohových enkodérů.
- Přetížení motorů – měřením momentu motoru (proudu) se vyhodnocuje najetí manipulátoru do překážky, či vysoké zatížení břemenem.
- Neočekávaná změna momentu motoru – například při působení vnější síly na manipulátor.
- Změna rychlosti motorů – při rozdílných hodnotách mezi skutečnou a požadovanou či očekávanou rychlostí vyhlásí poruchu.
- Odchylka skutečné pozice od požadované – v okamžiku dojezdu na pozici a její skutečné odchylky nad povolené pásmo.
- Detekce vysoké teploty jednotky, poruchy ventilátorů, vysokého napětí na jednotce či poklesu napájecího napětí vyhlašuje poruchu jednotky.
- Hlídání chodu CPU (watchdog) a chyby paměti.

Všechny zmíněné poruchy vedou k zastavení jednotky a případnému odpojení napájení od motorů a zablokování elektromagnetické brzdy.

2.5.2 Programové vybavení jednotky

Vysoké nároky na jednoduchost implementace manipulátoru přímo na ovládaný proces vedou k potřebě vysoké programové připravenosti. Uvedení robotu do provozu lze dosáhnout i s minimem instalačních kroků. Obvyklý průběh nasazení obsahuje tyto fáze:

- Instalace manipulátoru v pracovním prostoru, propojení jednotky RC180 s mechanikou robotu, zajištění napájení, kontrola překážek v pracovním prostoru manipulátoru.
- Oživení jednotky, nastavení komunikace, nastavení parametrů jednotky, prostorová a úhlová kalibrace.
- Programování konkrétní aplikace. Uživatelský program se sestavuje v programovacím jazyce SPEL+. Programátor zde může využít bohaté škály hotových funkcí, u kterých se jednotka sama skrytě stará o výpočty matematických modelů inverzní a přímé kinematiky a dynamiky.

2.5.3 Jazyk SPEL+

Pro sestavování aplikačních programů v jednotce Epson RC180 je určen vyšší programovací jazyk SPEL+. Syntaxe jazyka je podobná jako u jazyků z rodiny Basic. Podobně jako v Basicu nejsou příkazy na konci řádku ukončovány, ale na rozdíl od něj je nutné brát v potaz Case-Sensitive. Jako charakteristiku jazyka lze považovat velká první písmena u všech klíčových slov i názvů předem definovaných funkcí.

2.5.3.1 Základní konstrukce a klíčová slova jazyka

- ***Function*** *_name[[{**ByRef | ByVal**}] _varName [( )]* ***As*** *_type] [**As** _type] _code* ***Fend***

Tato konstrukce definuje funkci. Jako volitelné vstupní parametry lze použít parametry předávané odkazem *ByRef*, kdy změna hodnoty parametru ve funkci je zpětně šířena i do vnějšího programu. Druhou možností je použít výchozí typ *ByVal*, kdy je parametr předávaný hodnotou. Jako parametr nelze předávat celou proměnnou typu pole, nýbrž je nutné definovat konkrétní prvek. Ani návratová hodnota nemůže být typu pole. Předávání návratové hodnoty se provádí přiřazením do proměnné stejného jména jako je název funkce. Funkci lze opustit i uprostřed programu pomocí *Exit Function.*

- *_type _varName[()]* - definice proměnné

Jako v ostatních jazycích jsou klíčová slova pro základní datové typy Boolean, Byte, Double, Integer, Long, Real, String. Každá proměnná může být i typu pole o maximálně 3 dimenzích a maximálním počtu prvků 1000 v každé dimenzi. Indexace prvků pole začíná od indexu 1.

- Operátor přiřazení **=**

Shodný jako u ostatních jazyků. Při inicializaci proměnné nelze přiřazení uskutečnit na stejném řádku jako je definice.

- Binární operace ***Not***, *_vyrazA* ***And*** *_vyrazB*, ***Or***, ***Xor***.
- Relační operátory **=**, **<>**, **<**, **>**, **<=**, **>=**.
- ***If*** *_podminka* ***Then***_*code [**Else**][**ElseIf** _podminka* ***Then*** *_code] **EndIf**.*
- ***For*** *_var* **=** *_initVal* ***To*** *_finalVal [**Step** incr] _code **Next** [_var].*

Chování shodné jako u jiných jazyků, absence *DownTo,* nahrazuje záporná hodnota inkrementu. Za klíčovým slovem *Next*, které ukončuje cyklus, může být proměnná, do které se přiřadí počet průchodů cyklem.

- ***Do*** *[{ **While** | **Until** } _podminka]_code [**Exit Do**] [_code]* ***Loop***

Příkaz *Exit Do* ukončuje vykonávání celého cyklu a pokračuje se programem za klíčovým slovem *Loop.*

- ***Do*** *_code [**Exit Do**] _code* ***Loop*** *[{ **While** | **Until** } _podminka]*
- ***Select*** *selectExpr* ***Case*** *caseExpr _code [**Default** _code]* ***Send***
- Příkazy skoků ***GoTo*** *_nazevNavesti|_cisloRadku,* a návěští _nazevNavesti**:**

Příkazy skoku jsou v jazyce SPEL+ a potažmo v řídící jednotce řešeny bezpečně. S rozumným používáním konstrukce skoku se na rozdíl od jiných jazyků počítá. Výhodná může být například při obsluze Errorů.

- ***OnErr GoTo*** *_errHandle, errNum =* ***Err***, ***ErrMsg$****(errNum),* ***EResume*** *newCycle*

Konstrukce *OnErr* nastavuje číslo řádku nebo návěští místa v programu, kam se provede odskok, v případě, že se v jednotce potažmo na robotu vyskytne porucha. Příkaz *Err* vrací číslo poslední chyby. *ErrMsg$* vrací popis chyby ve formátu String. *EResume* navrátí běh programu na číslo řádku nebo na návěští po vykonání Error handling.

- ***Call*** *_nameFunction [(argList)]* **–** volání funkce s předáváním parametrů.
- ***Print*** *["_text"][, _valVar] [,_ strVar]* – textový výpis na konzolu monitorovacího okna nebo na operátorský panel. Při výpisu se nemusí převádět číselné proměnné na String.

2.5.3.2 Příkazy a funkce pro pohyb manipulátoru nebo pro nastavování parametrů

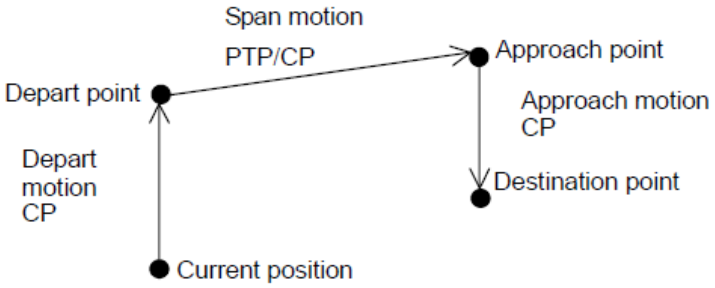
- ***Go*** *{_point|**XY()**}* - PTP posun manipulátoru.

Přesune koncový bod manipulátoru do nové pozice v prostoru definované uživatelským bodem nebo XYZ popisem. Rozdíl oproti příkazu *Move* je takový, že u *Go* není přesně definována trajektorie, jak se do nového bodu robot dostane. Pro výpočet bodu v prostoru a potřebného natočení os se využívá inverzní kinematiky, přičemž výsledná trajektorie je optimalizována s ohledem na maximální rychlost přesunu.

- ***Move*** *{_point|**XY()**}* -posun manipulátoru do nového bodu po přímkové trajektorii.

Příkaz umožňuje přesun manipulátoru do koncového bodu po přímkové trajektorii. Není-li možné na základě výpočtu tento pohyb po přímce provést, zůstane manipulátor na místě a jednotka vygeneruje poruchu.

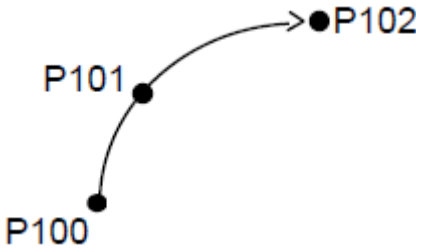
- ***Jump3*** *_departPnt, _apprPnt, _destPnt* – posun manipulátoru po jeřábové trajektorii.



Obr. 14 - Trajektorie pro příkaz Jump3 [3]

Pro manipulátor Epson C3 tvoří náhradu za příkaz *Jump*. Jedná se o spojení dvou přímočarých pohybů (zvedání a spouštění) a jednoho pohybu typu PTP (XY přesun).

- ***Arc*** *_middlePnt, _destPnt (Arc P101, P102)* – posun manipulátoru po křivce přes bod.



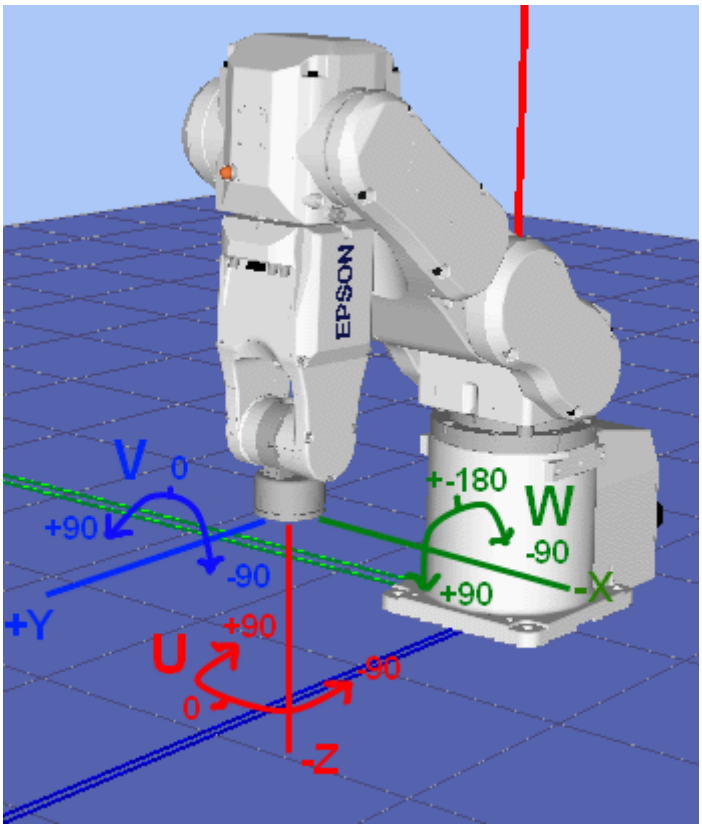
Obr. 15 - Trajektorie pro příkaz Arc [3]

Jednotka vypočítává pohyb do koncového bodu tak, aby byla trajektorie hladká (po křivce) a procházela skrze definovaný střední bod. Křivka se vypočítává pouze v rámci XY os.

- ***Arc3*** *_middlePnt, _destPnt* – posun manipulátoru po křivce přes bod v rámci XYZ

Principiálně stejné jako *Arc* s rozdílem, že křivka je vypočtena v rámci všech 3 prostorových dimenzí.

- ***XY(****x, y, z, u, [v, w])* – výpočet nového bodu na základě 3 dimenzí a 3 náklonů.



Obr. 16 - Souřadnice bodu v 6 osách

První 3 souřadnice vypovídají jednoznačně o poloze koncového bodu manipulátoru v prostoru. Parametry U, V a W udávají náklon nebo rotaci s osou rotace dle [Obr. 16]. Jestliže se u příkazu poslední 2 parametry neuvedou, zůstává náklon stejný jako při posledním pohybu.

- ***JTran*** *_jntNum, _distDeg* – relativní otočení zvolené osy o zvolený úhel.

  Zafixuje pohyb všech os kromě osy zvolené a u ní provede relativní otočení o definovaný úhel ve stupních.

- ***PTran*** *_jntNum, _distPulse* – relativní otočení zvolené osy o zvolený počet pulsů.

  Stejné jako *JTran,* ale místo úhlu se otáčí o definovaný počet inkrementovaných pulsů.

- ***Tool*** *_toolNum* – změna aktuálního nástroje.

  Pozicování koncového bodu pro příkazy pohybu manipulátoru je vhodné upravit podle aktuálního nástroje, uchopovadla či jiného břemene. Jednotka umožňuje uložit až 16 nástrojů do paměti. Příkaz *Tool* změní aktuálně používaný nástroj. Sada 0 je výchozí neměnitelná, u níž není nastaven žádný offset.

- ***TLSet****_toolNum, _toolPntOffset* – změna offsetu zvoleného nástroje.

  Při dosazení *XY(x, y, z, u, v, w)* za *_toolPntOffset* se koncový bod manipulátoru posune po všech osách a náklonech jako u [Obr. 16]. Pouze offset u souřadnice *x* a náklonu *v* má opačný směr než jaký je běžný směr os v pozici manipulátoru [Obr. 16].

- ***InPos*** – dotaz, zda- se manipulátor právě pohybuje nebo stojí.

Vrací binární hodnotu 1, v případě, že manipulátor dosáhnul posledního požadovaného koncového bodu a již se nehýbe.

- ***CX(CurPos), CY(), CZ(), CU(), CV(), CW()*** – aktuální souřadnice manipulátoru.

Funkce *CurPos* vrací aktuální pozici manipulátoru v prostoru ve formě bodu. Funkce *CX()*, *CY()*....získává z předaného bodu jednotlivé souřadnice X, Y, ...

- ***Agl(_numJnt)*** – vrací aktuální úhel natočení ve stupních zvolené osy.
- ***AglToPls(_J1, _J2, _J3, _J4, _J5, _J6,)*** – převádí absolutní úhly natočení jednotlivých os na hodnoty pulsních inkrementů. Výstup je ve formě bodu popsaného pulsy.
- ***PPls(_srcPnt,_ numJnt)*** – vrací pulsy dané osy z předloženého bodu.
- ***Home*** – přesune manipulátor do výchozí pozice definované jako Home.
- ***HomeSet _PJ1, _PJ2, _PJ3, _PJ4, _PJ5, _PJ6*** – nastavení výchozí pozice.

Nastavení výchozí pozice v inkrementačních pulsech pro každou osu. Posloupnost natáčení os při návratu do výchozí polohy lze v jednotce definovat.

- ***On(_outNum), Off(_outNum)*** – set a reset funkce pro výstupy z tabulky I/O bodů.
- ***Sw(_inNum)*** – přečtení stavu vybraného vstupu z tabulky I/O bodů.
- ***Speed(_perc, [_depart], [_appro]), SpeedS*** – rychlost pohybu manipulátoru v prostoru.

Definuje rychlost pohybu celého manipulátoru. Hodnota *_perc* je v procentech 1-100 a určuje rychlost pro PTP pohyb (*Go*), *_depart* a *_appro* určuje rychlost pro zvedací a spouštěcí sekvenci u *Jump* pohybu. SpeedS nastavuje rychlost pro přímočarý pohyb (*Move*) z rozsahu 1 – 20000 v mm/s.

- ***Accel(_accel, _decel, [_deptAcce, _deptDecel, _apprAcce, _apprDecel]), AccelS*** – zrychlení

Definuje zrychlení a zpomalení pohybu celého manipulátoru, hodnota je v procentech 1-100 a udává zrychlení pro PTP pohyb (*Go*), *AccelS* nastavuje zrychlení a zpomalení pro přímočarý pohyb (*Move*) z rozsahu 0.1 – 25000 v mm/s^2.

- ***Power {Low|High}*** – mění globálně výkon motorů

V případě režimu *Low*, jsou nastavené parametry Accel a Speed ignorovány.

- ***OpenNet #_portNum As {Server|Client}*** – inicializace komunikačního rozhraní.
- ***WaitNet #_portNum*** – čekání na spojení z druhé strany komunikačního kanálu.
- ***Input #_portNum, [$_strName, _numVal]*** – ASCII čtení z komunikačního portu.

Po otevření komunikačního rozhraní (portu), lze přicházející ASCII data oddělená mezerou a ukončená definovaným znakem číst. Data mohou být podle formátu automaticky konvertována na číselné typy. V případě, že nepřichází správný formát dat dle posloupnosti Input...., je vyhlášena porucha.

- ***Print #_portNum, [$_strName, _numVal]*** – ASCII zápis na komunikační port

Příkaz funguje obdobně jako *Input,* pouze s opačným směrem.

2.6 UCHOPOVACÍ AKTOR MANIPULÁTORU

Pro účely přesunu kamenů na šachovnici je potřeba zvolit vhodný uchopovací mechanismus. V průmyslové praxi se nejčastěji používají 2 mechanismy:

- *Podtlaková úchopná hlavice* - funguje na principu odsátí vzduchu v prostoru mezi pryžovým zvonem a uchopovaným materiálem. Značnou nevýhodu představuje potřeba vyvíječe podtlaku. V robotice se často používá kompresor s Venturiho trubicí jako ejektor.
- *Uchopovací čelisti* - čelisti mohou být ovládány elektrickými servomotory nebo pomocí pneumatického mechanismu. Elektricky ovládané čelisti jsou vhodné v aplikacích, kdy je nutné přesně řídit pozici čelistí. Pro správnou funkci je však nutné přidat elektroniku pro měření momentu motoru - uchopovací sílu. Pneumatické čelisti mají uchopovanou sílu řešenou buď řízením ovládacího tlaku, nebo autonomně mechanicky. Konstrukce se prodražuje potřebou vyvíječe tlaku.

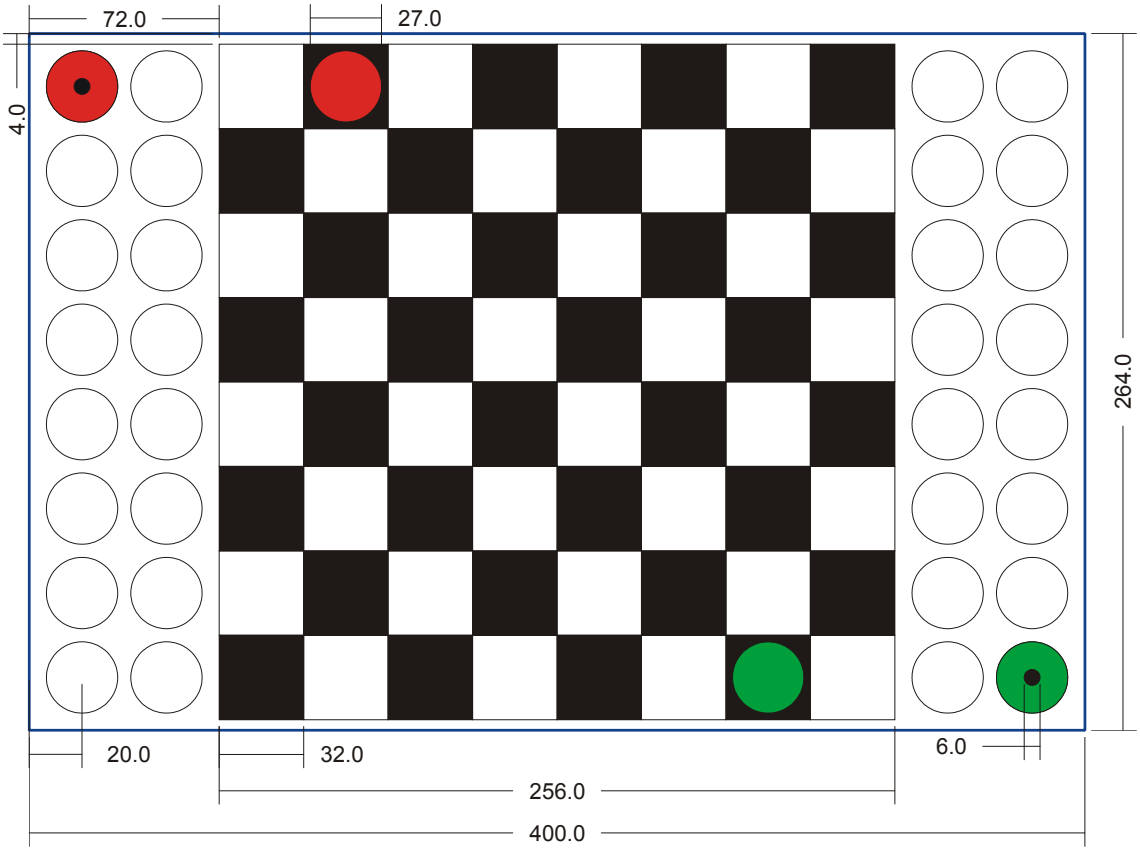
U robotického hraní dámy lze jako materiál hracích kamenů zvolit feromagnetický kov a figury uchopovat pomocí přídržného elektromagnetu. Uchopovaná figura uzavře jako kotva jinak otevřený magnetický obvod, který je vyvozován stejnosměrným elektrickým proudem. Výhodou je velice jednoduché řízení (spínání DC proudu) a velká přídržná síla.

3. PRAKTICKÁ ČÁST

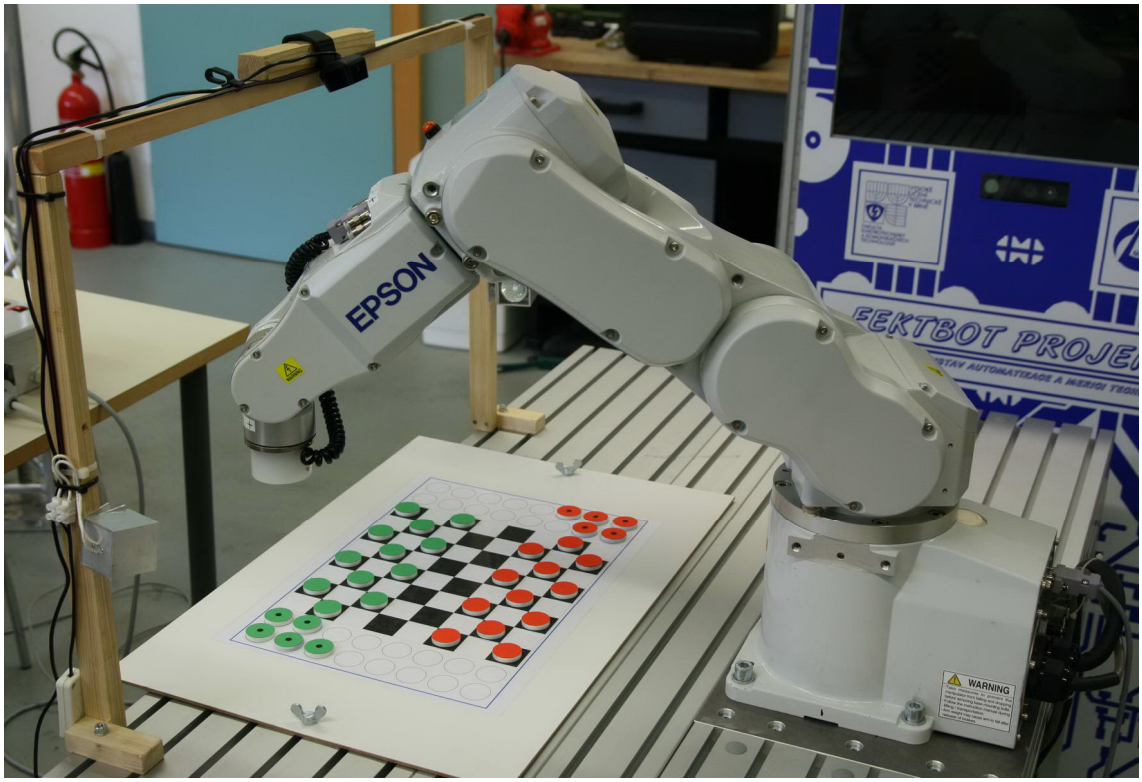
3.1 HERNÍ SCÉNA

3.1.1 Kompozice herní scény a herní materiál

Skutečné provedení šachovnice a hracích kamenů je zobrazeno na Obr. 2 a Obr. 17. Umístění šachovnice v manipulačním prostoru robotu je vidět na Obr. 18. Zalaminovaný výtisk šachovnice je nalepen na sololitové desce bílé barvy a tloušťky 4 mm. Výška ocelových kamenů je 4 mm, na horní straně s nalepovacím označením figury. Pozice levého horního rohu šachovnice v prostorových souřadnicích manipulátoru je [+200,+267] [X, Y].



Obr. 17 - Šachovnice s kótami, míry v mm



Obr. 18 - Prostorové uspořádání demonstrační úlohy

3.1.2 Osvětlení scény a uchycovací portál

Uchycení kamery a osvětlení zajišťuje portál nad šachovnicí. Pro testovací účely byl vyroben dřevěný vzorek o šířce 750 mm a výšce 500 mm. Vzdálenost portálu od robotu byla zvolena jako kompromis mezi přímou pozicí nad šachovnicí, místem, které neomezuje pracovní prostor robotu a umístění, které garantuje viditelnost celé šachovnice bez těla robotu.

Jako osvětlovací člen byl vyroben prototyp hliníkového svítidla s halogenovou žárovkou. Konstrukce vyniká vysokou účelností a velkou intenzitou světla. Reflektor svítidla je vůči svému uchycení naklápěcí, což umožňuje lépe směrovat paprsek světla. Omezujícími parametry žárovky jsou patice MR16, 12 VDC, 20 W, průměr reflektoru 30 mm. Napájení 12V DC bylo zvoleno s ohledem na bezpečnost, protože nelze vyloučit dotek živé části.

Obr. 19 - Použité svítidlo

3.2 SNÍMÁNÍ ŠACHOVNICE A POČÍTAČOVÉ VIDĚNÍ

3.2.1 Snímací kamera

U detekce objektů na šachovnici nejsou kladeny vysoké požadavky na rychlost snímání. Okamžiky vzorkování jsou pouze v časech, kdy robot nebo hráč ukončí svůj tah. Jako dostatečné rozlišení pořízených snímků lze považovat 640 x 480 pixelů. Toto rozlišení je kompromisem mezi vysokou přesností detekce a nízkou dobou zpracování obrazových dat. Pro přesné nastavení ohniskové vzdálenosti je výhodné použít kameru s automatickým ostřením.

Na základě výše zmíněných požadavků byla vybrána webová kamera Microsoft LifeCam HD-5000 s kvalitní optickou soustavou, příznivou cenou a udávanými parametry:

- Automatické ostření: 15 cm - nekonečno; diagonální snímací úhel 66°.
- Rozlišení snímače: 1.3 Mpix; max. rozlišení videa:1280 x 720 bodů; 30fps.
- Rozhraní USB2.0; 24-bitová barevná hloubka.

Obr. 20 - Snímací kamera Microsoft LifeCam HD-5000

Při použití v praxi se ale ukázalo automatické ostření jako pomalé. Automatické ostření lze v dodávaném SW ovladači vypnout a nastavit jej na pevnou hodnotu, avšak platnost tohoto parametru je pouze po dobu připojení kamery k PC. Funkce pro dosažení optimální expozice TrueView a automatické vyvážení bílé se ukázaly jako velice praktické, protože spolehlivě poskytují, spolu s vhodným osvětlením, podobné obrazové výsledky při různých vnějších podmínkách.

3.2.2 Knihovna Aforge.NET

Na knihovnu v prostředí .NET, která se zabývá počítačovým zpracováním obrazu, jsou kladeny 2 základní požadavky. Musí zprostředkovat rozhraní mezi standardní kamerou a programovacím jazykem C# a dále by mělo obsahovat paletu základaních metod počítačového vidění. Všeobecně známá otevřená knihovna OpenCV výše zmíněné požadavky splňuje. Ve spojení s programovacím jazykem C# a platformou .NET ke své funkci ale potřebuje tzv. wrapper, který převádí objekty z C++ na C#. Vhodnější je tedy využít knihovnu, která je přímo určena pro .NET. Aforge.NET je otevřená knihovna pod licencí LGPL, která obsahuje všechny zmíněné a mnohé další funkce.

Struktura celé knihovny je vhodná pro přímou implementaci ve vývojovém prostředí Microsoft Visual Studio a je přímo sestavovaná v jazyce C#. Dělí se podle své funkce do hlavních jmenných prostorů. Pro zprostředkování rozhraní s kamerou jsou nutné součásti AForge.Video a AForge.Video.DirectShow. Pro práci se získaným obrazem jsou potřebné části AForge.Imaging a AForge.Imaging.Filtres.

3.2.3 Komunikační rozhraní s kamerou

Třída AForge.Video.DirectShow.VideoCaptureDevice obstarává všechny nástroje potřebné k nastavení a samotnému spuštění rozhraní. Umožňuje vybírat konkrétní kameru, upravit rozlišení snímku, vzorkovací frekvenci získávaných snímku nebo otevřít dialogové okno ovladače s parametry kamery.

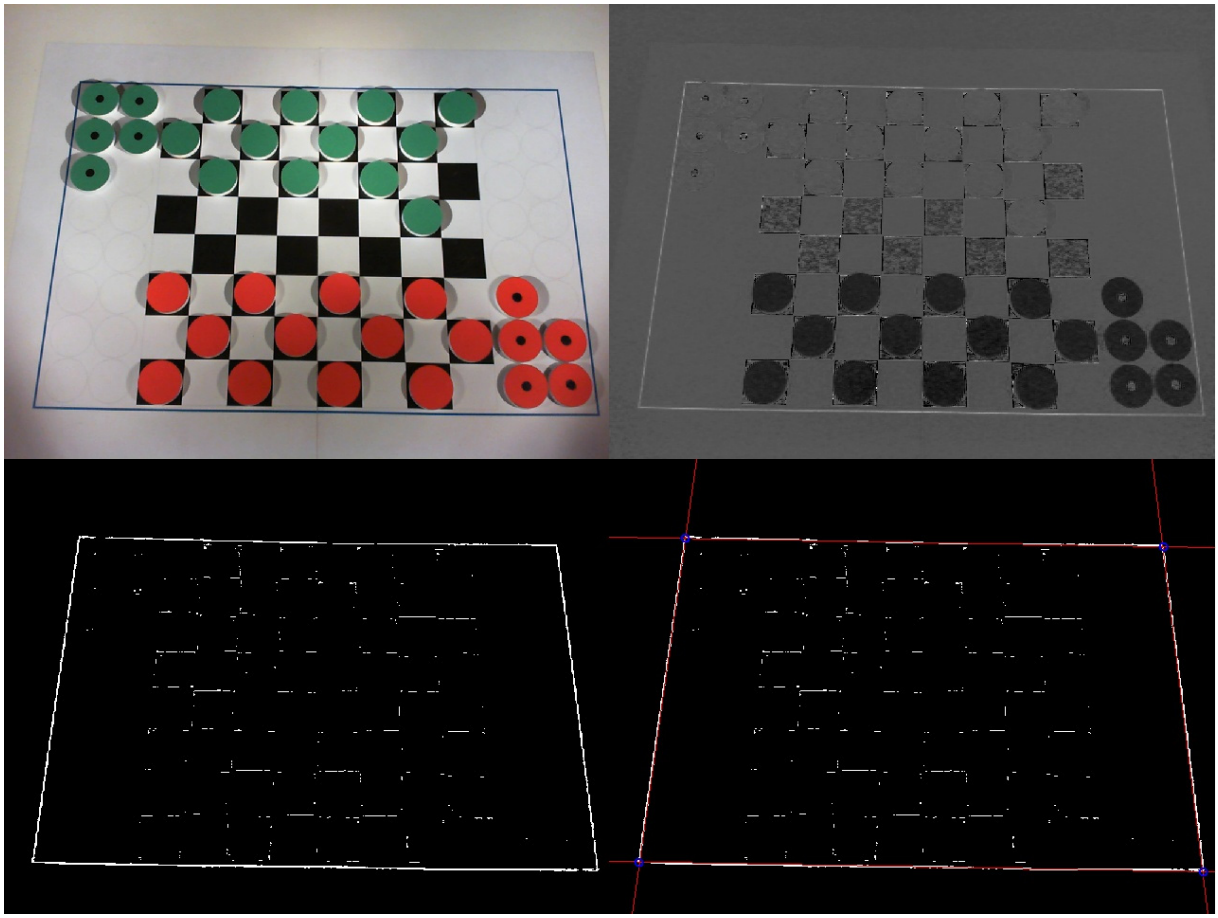
U řídícího programu demonstrační úlohy se okno ovladače otevírá kvůli vypnutí automatického ostření nebo změně parametrů expozičních funkcí jako je TrueColor a vyvážení bílé. Výstupní data z rozhraní jsou pravidelně se obnovující snímky ve formátu Bitmap. Obnovovací frekvence byla nastavena na 3 snímky za sekundu. Další funkce pro zpracování obrazu si snímek kopírují pouze v okamžiku vyhodnocení tahu, a proto nemusí být frekvence příliš vysoká. Startovat a zastavovat rozhraní pouze v okamžicích potřeby není možné, protože start rozhraní trvá přibližně 2 - 3 sekundy.

3.2.4 Detekce šachovnice v obraze, geometrická transformace

3.2.4.1 Detekce šachovnice v obraze

Postup pro nalezení šachovnice v obraze vychází z kapitoly 2.2.3. Získaný snímek scény se předloží jako vstup pro mapovací funkci (3), která vyhledává modré

objekty v obraze. Výstup je zobrazen jako šedotónový obraz (pravý horní obraz na Obr. 21), kde bílá barva symbolizuje hodnotu mapované úrovně 1. Pomocí empiricky stanovené hodnoty prahu (100 v rozsahu 0-255) se snímek převede do černobílého modelu (levý spodní obraz na Obr. 21).



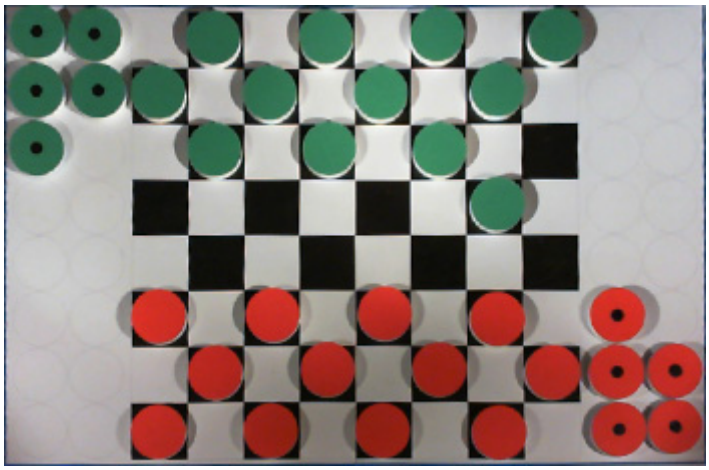
Obr. 21 - Sekvence pro detekci šachovnice v obraze

K vyhledání úseček v obraze slouží Houghova transformace, která je součástí knihovny AForge.NET. Výchozí hodnota relativní intenzity rozhodovacího prahu v rozsahu 0-1 byla empiricky zvolena jako 0.4. V případě, že se však nepodaří při této úrovni nalézt v obraze 4 kompatibilní přímky, hodnota prahu je cyklicky upravována. V případě, že metoda nalezne přímek příliš mnoho, hodnota prahu se inkrementuje až o 0.05. V případě malého množství nalezených přímek se práh dekrementuje až o 0.05. Překročí-li úroveň rozsah <0.1, 0.9> vyhlásí se chybové hlášení „Can't choose suitable relative intensity for Hough transf“. Na pravém spodním obraze v Obr. 21 jsou nalezené přímky vyznačeny červenou barvou. Po nalezení se přímky z polárních souřadnic převedou do směrnicového vyjádření, které zajišťuje přehlednější další výpočty. Podle analytického popisu se přímky roztřídí na horní, spodní, pravou a levou s dělící pozicí ve středu obrazu. Z analytického popisu se určí i průsečíky mezi

přímkami, které jsou dále brány jako kalibrační body (na pravém spodním obraze v Obr. 21 jsou vyznačeny modrými kruhy).

3.2.4.2 Geometrická transformace deformovaného obrazu

Vstupem pro metodu, která odstraňuje perspektivní zkreslení pomocí kolineární transformace, jsou 2 sady po čtyřech bodech. Jako výchozí použijeme kalibrační body obrazu. Koncové body odpovídají skutečným vzdálenostem mezi rohy modrého orámování. Aby se zajistilo další snadné poměřování, bylo zvoleno měřítko výsledného obrazu jako 1 mm : 1 pixel. Obslužná třída v Aforge.NET se jmenuje QuadrilateralTransformation.



Obr. 22 - Transformovaný obraz s kompenzací perspektivního zkreslení

3.2.5 Detekce kamenů v obraze

Pro detekci pozic červených a zelených kamenů se odděleně provede následující sekvence úprav, které vycházejí z kapitoly 2.2.3.

3.2.5.1 Převod do červené a zelené mapy

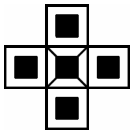
Stejně jako při detekci modrého rámu se nejprve předloží již transformovaný obraz do mapovací funkce, která zvýrazní vyhledávané barvy.

3.2.5.2 Filtrování roztřepených hran, vyhlazení obrazu

Po přemapování obrazu do konkrétních barevných úrovní může dojít k těmto degradačním změnám:

- Hrany figur se mohou roztřepit,
- mezi blízkými kameny může docházet ke splývání figur,
- černý střed kamene pro odlišení dámy může splývat s okolním povrchem kamene,
- různé stíny, odlesky a nedokonalosti nasvícení se mohou na povrchu figur zvýraznit a struktura barevné mapy kamene tak nemusí být jednolitá.

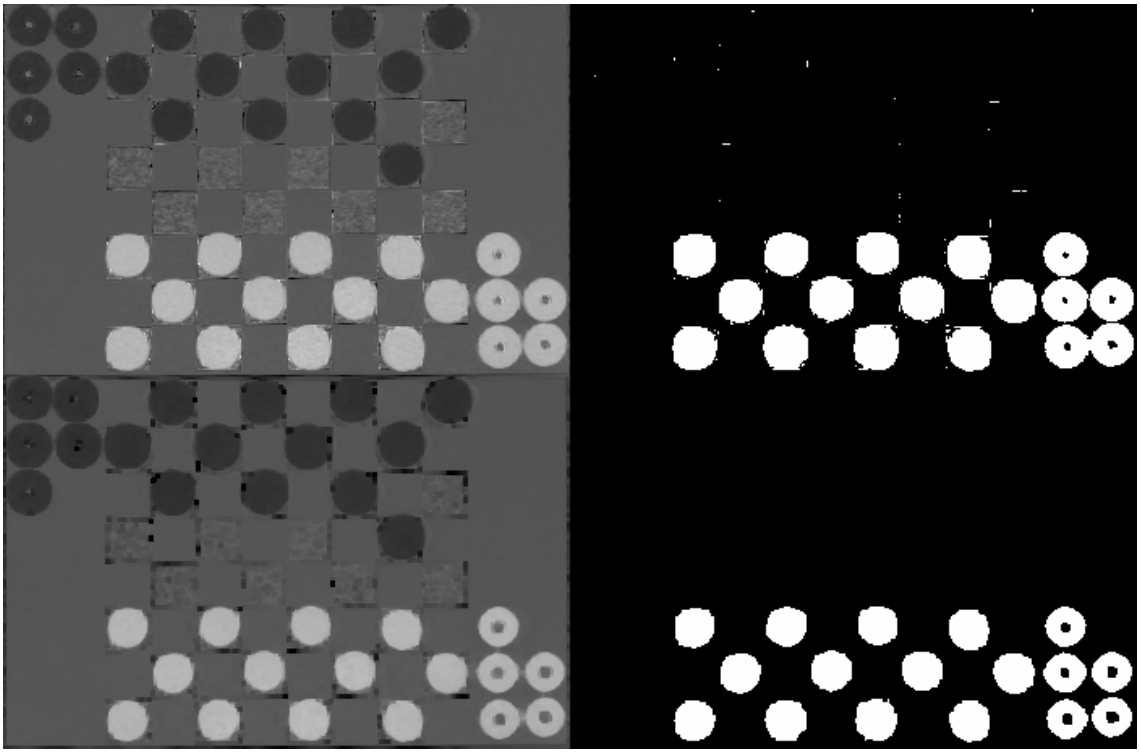
Všechny zmíněné nedostatky (pravý horní obraz na Obr. 24 a Obr. 25) lze omezit použitím některé z filtračních metod. Nejlepších výsledků bylo dosaženo při aplikaci morfologické operace eroze. V tomto případě se jedná o šedotónovou verzi operace se strukturním elementem dle Obr. 23 místo častěji používané binární varianty. Eroze se používá pro zjednodušení struktury objektů, objekty tloušťky 1 se ztratí, a tak se složitější objekt rozdělí na několik jednodušších. Šířka objektu hracího kamene je sice pomocí eroze snížena o jeden pixel z každé strany, což ale nemá na přesnost detekce vliv, protože pozice kamene je odvozena od středu nalezeného kamene. Výsledky operace eroze lze dokladovat na levém spodním obraze na Obr. 24 a Obr. 25.



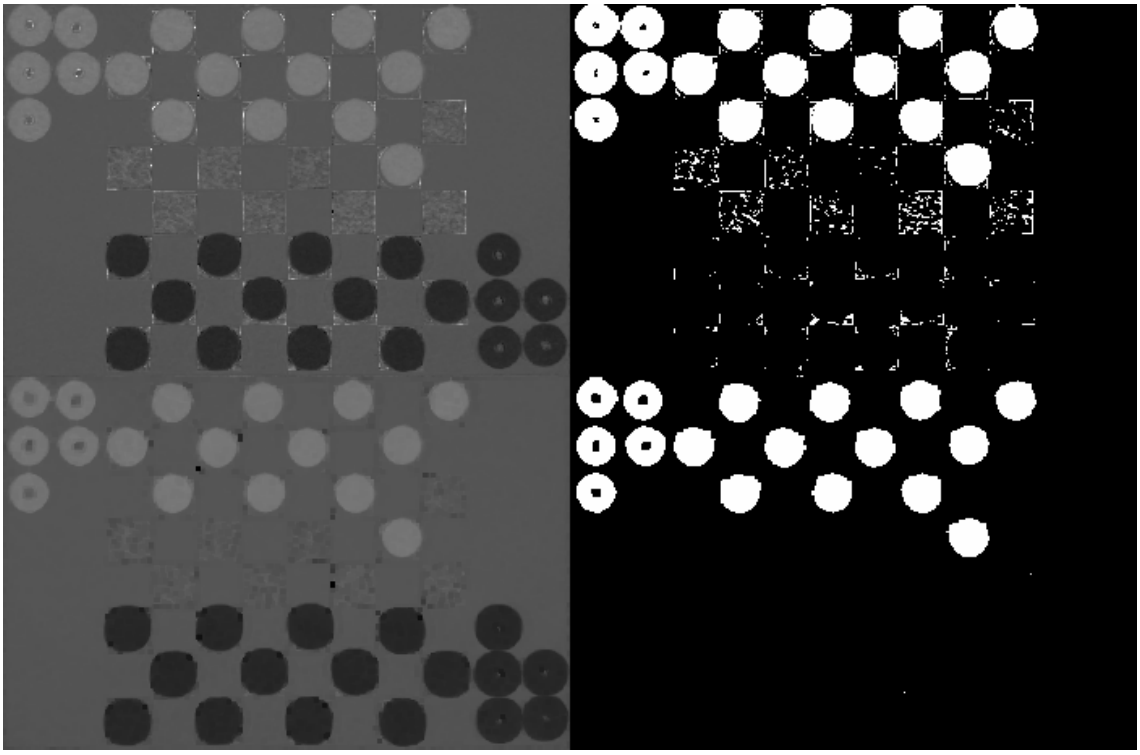
Obr. 23 - Strukturní element 3x3 pro šedotónovou erozi

3.2.5.3 Prahování obrazu

Po provedení filtrace se obraz převede do binárního modelu pomocí prahování s vhodně zvolenou velikostí prahu. Vzhledem k vyšší výstupní mapované úrovni figur v červené barvě se volí hodnota prahu pro červený obraz vyšší (hodnota 135 z rozsahu 0-255). U zelených figur byla zvolena hodnota prahu jako 100.



Obr. 24 - Sekvence detekce červených figur, výchozí Obr. 22



Obr. 25 - Sekvence detekce zelených figur, výchozí Obr. 22

3.2.5.4 Nalezení kamenů v segmentovaném obraze

Knihovna Aforge.NET implementuje třídu BlobCounter, která hledá v černobílém obraze shluky objektů s bílou barvou. Pro každý shluk lze definovat jeho minimální a maximální šířku a výšku, při které je shluk hodnocen jako nalezený objekt. Tuto třídu lze pro hledání kamenů s výhodou použít, protože velikosti kamenů jsou předem známé. Výstupem z BlobCounter jsou pozice obdélníků, které vyznačují meze nalezených shluků. Jako dostatečně spolehlivý způsob hledání těžiště kamenů se ukázal prostý střed vyznačujících obdélníků. Pozice těžiště kruhových figur by se měla limitně blížit ke skutečnému středu figury. Jestliže by obraz získaný ze segmentace obsahoval kolem obvodu kamenů zbytky hran šachovnice, bylo by nutné zvolit pro hledání těžiště jinou metodu, například založenou na geometrických momentech.

3.2.5.5 Kompenzace perspektivního zkreslení výšky kamenů

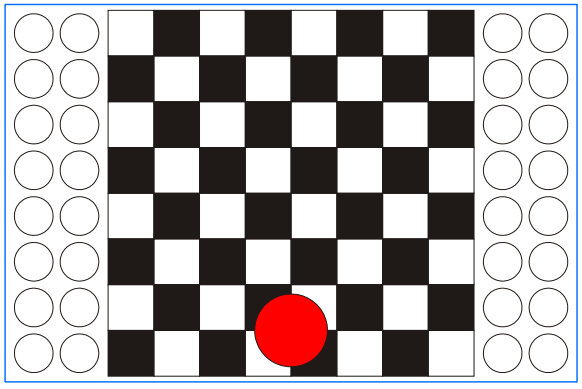
Při kompenzaci perspektivního zkreslení pomocí kolineární transformace se transformuje pouze plocha šachovnice jako celek. Nevykompenzovaným zůstává perspektivní zkreslení jednotlivých kamenů, které je zapříčiněno jejich nenulovou výškou a pozorováním pod nenulovým úhlem vůči kolmici desky. Při určování pozic kamenů pak dochází k posunu mezi detekovaným a skutečným středem, protože horní plocha kamene není přímo nad jeho základnou.

Obr. 26 - Detail zkreslení obrazu kamenů

Pro kompenzaci zkreslení výšky lze posun odečíst na základě empirického měření posunu. Z přibližné pozice kamery (červený kruh) na Obr. 27 bylo odvozeno, že stačí definovat posun na horním, spodním, levém a pravém okraji šachovnice a pomocí lineární interpolace dopočítat posun na libovolném místě. Konkrétní hodnoty posunů byly stanoveny dle Tab. 2, při uvažování počátku souřadnicové soustavy šachovnice v levém horním rohu.

	+3	
+1		-1
	0	

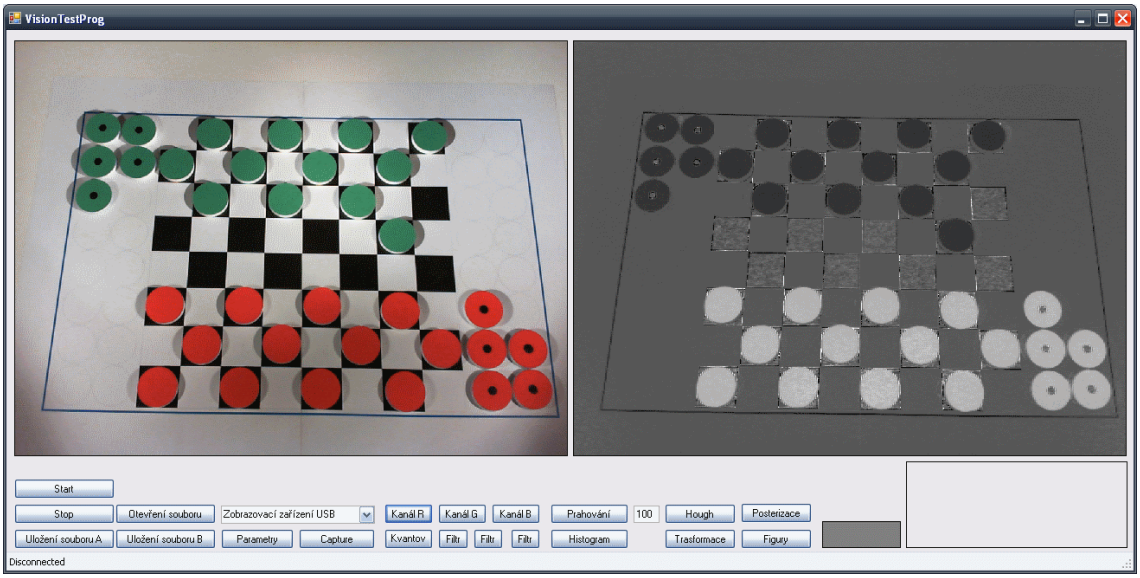
Tab. 2 - Posuny pozic středu figur na šachovnici



Obr. 27 - Přibližná pozice kamery nad šachovnicí

3.2.5.6 Testovací program počítačového vidění

Všechny funkce snímání a počítačového vidění byly nejprve otestovány na vlastním pomocném programu, který v sobě implementuje různé Aforge.NET nástroje.

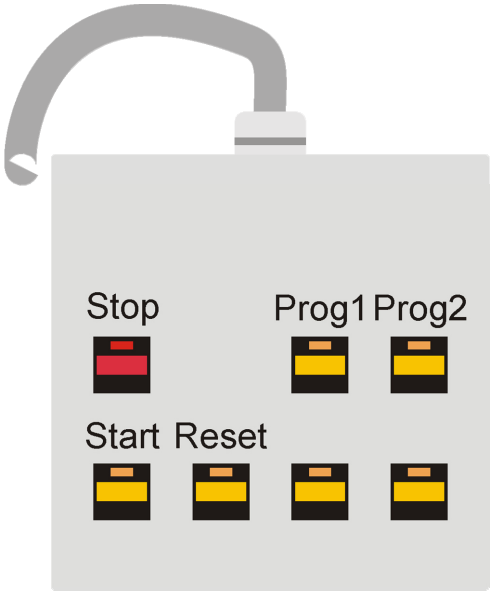


Obr. 28 - Testovací program počítačového vidění

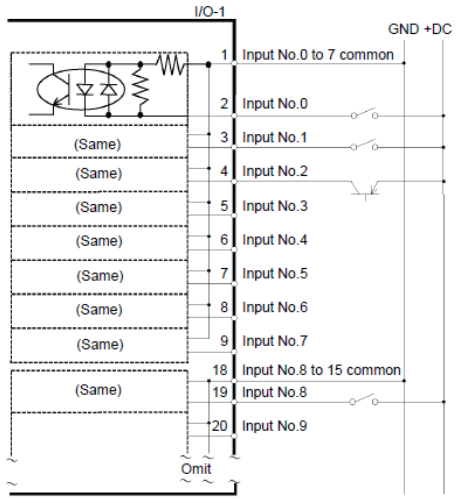
3.3 OVLÁDÁNÍ MANIPULÁTORU

3.3.1 Ovládací panel základních funkcí

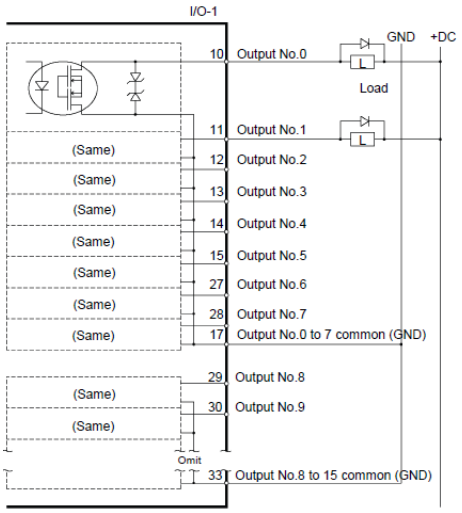
Jak bylo řečeno o řídící jednotce RC180 v kapitole výše, umožňuje plnohodnotné nasazení na konkrétní aplikaci bez dalšího řídícího členu. Pro tyto účely je nutné zajistit prostředek pro spouštění, zastavování a přepínání uživatelských programů. Jedním s možných řešení je využít standardních binárních vstupů/výstupů a k nim připojit tlačítka a LED pro monitorování funkce.



Obr. 29 - Pohled na rozmístění tlačítek a LED na ovládacím panelu



Obr. 30 - Možné zapojení binárních vstupů u RC180



Obr. 31 - Možné zapojení binárních výstupů u RC180

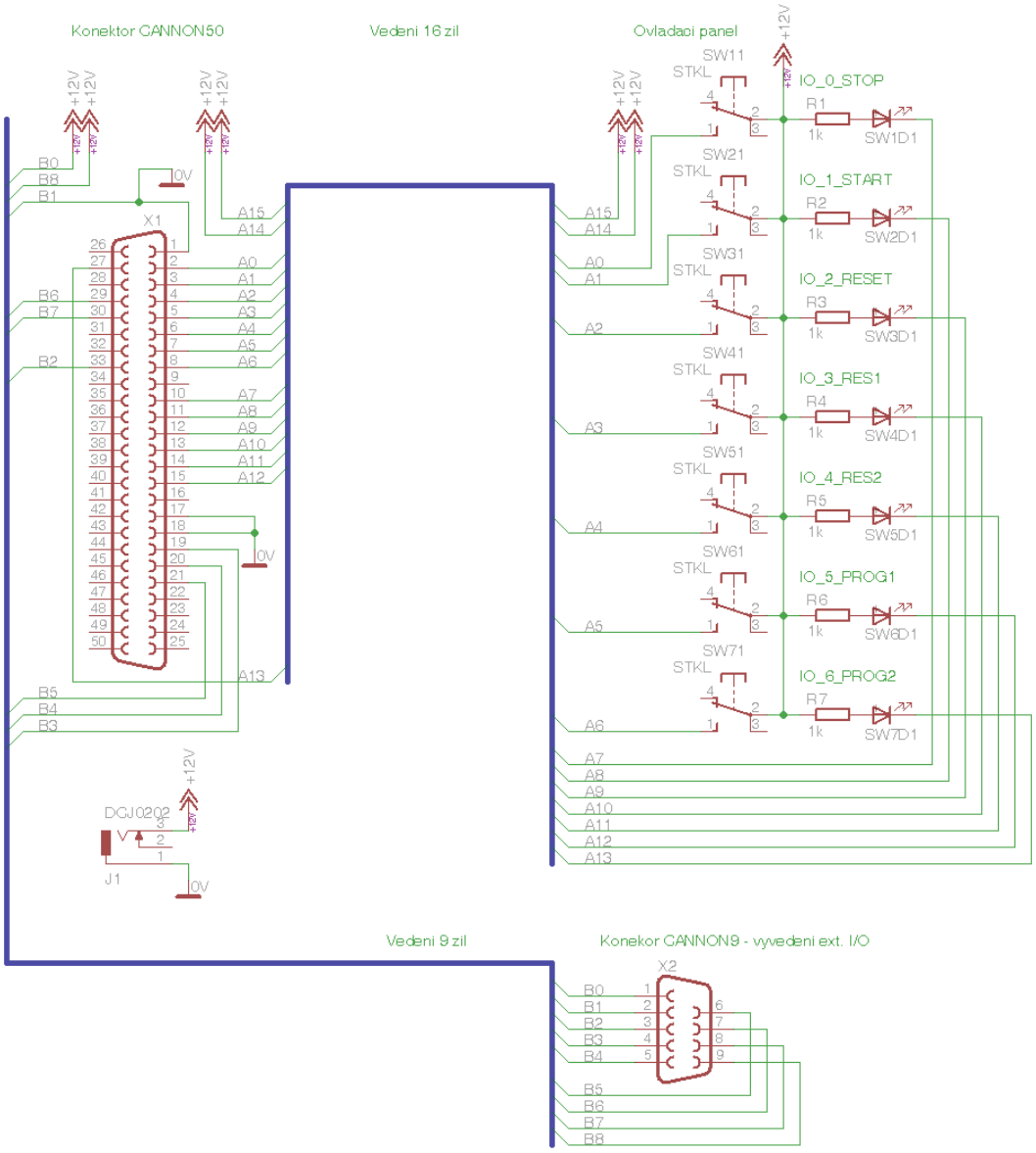
3.3.1.1 Návrh ovládacího panelu

K základnímu ovládání jednotky dostačuje 7 tlačítek:

- Zastavení vykonávání všech programů – z bezpečnostních důvodů červené.
- Zavedení spouštěcího programu – v případě, že je jednotka v pořádku, je umožněno spustit zaváděcí program.
- Reset chyby nebo varovného hlášení.
- 2 tlačítka pro spouštění jednoho ze dvou programů uložených v jednotce.
- 2 rezervní tlačítka s libovolně konfigurovatelnou funkcí.

Pro signalizaci stavu manipulátoru bylo zvoleno 7 LED, které jsou součástí tlačítek a mohou tak logicky souviset se spouštěnou funkcí.

Ovládací panel musí obsahovat konektor pro externí napájení +12 V, protože v konektoru I/O ani jinde na jednotce není žádné napájení vyvedeno.



Obr. 32 - Schéma připojení ovládacího panelu + konektor IO na RC180

DB50	Barva	In/Out	No. I/O	Tlačítko
1	GND			
2	modrá	Input	0	Červené
3	růžová	Input	1	Žluté Tl.1
4	šedá	Input	2	Žluté Tl.2
5	žlutá	Input	3	Žluté Tl.3
6	zelená	Input	4	Žluté Tl.4
7	hnědá	Input	5	Žluté Tl.5
8	bílá	Input	6	Žluté Tl.6
10	šedo/růžová	Output	0	Červené
11	žluto/hnědá	Output	1	Žluté Tl.1
12	fialová	Output	2	Žluté Tl.2
13	bílo/žlutá	Output	3	Žluté Tl.3
14	hnědo/zelená	Output	4	Žluté Tl.4
15	červeno/modrá	Output	5	Žluté Tl.5
17	GND			
18	GND			
19	bílo/zelená	Output	6	Žluté Tl.6
	černá+červená			+V12

Tab. 3 - Značení žil v kabelu k ovládacímu panelu

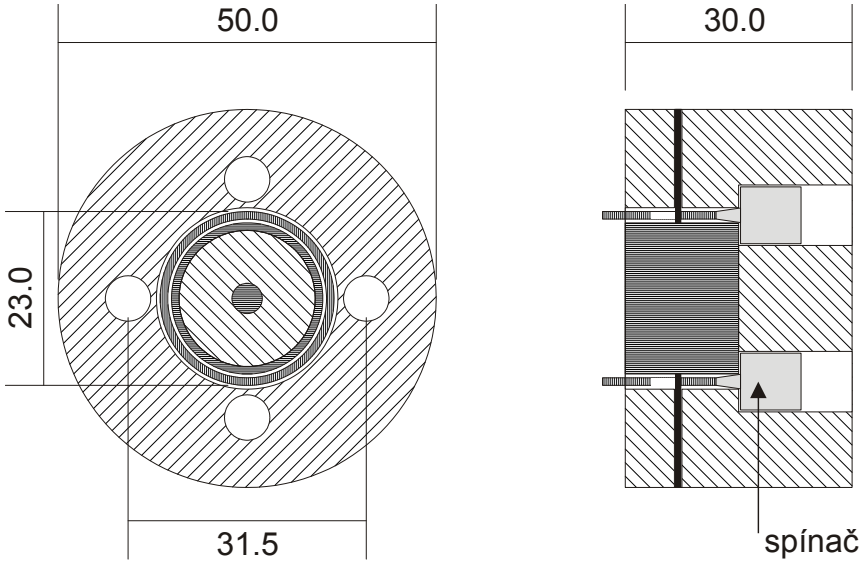
Aktuální hodnotu stavu tlačítek může jednotka získávat programově pomocí SPEL+ funkce *Sw(numInput)* a stavy LED nastavovat pomocí funkcí *On(numOutput)/ Off(numOutput)*. Pro tlačítka Stop/Start/Reset se však hodí pevné spojení standardního vstupu s Remote Control funkcí. Přiřazení se definuje v Epson RC+ *Setup/Controller/Remote Control/Inputs.* RC180 má ve výchozím stavu přepnuty standardní IO do virtuálního režimu, kdy nereaguje na fyzické změny na IO portu. Zrušit virtuální režim lze v *Setup/Controller/Preferences/Virtual I/O*. Pro přehlednost lze každému I/O přiřadit i Label v *Tools/I/O Label Editor.*

3.4 PŘÍDRŽNÝ ELEKTROMAGNET

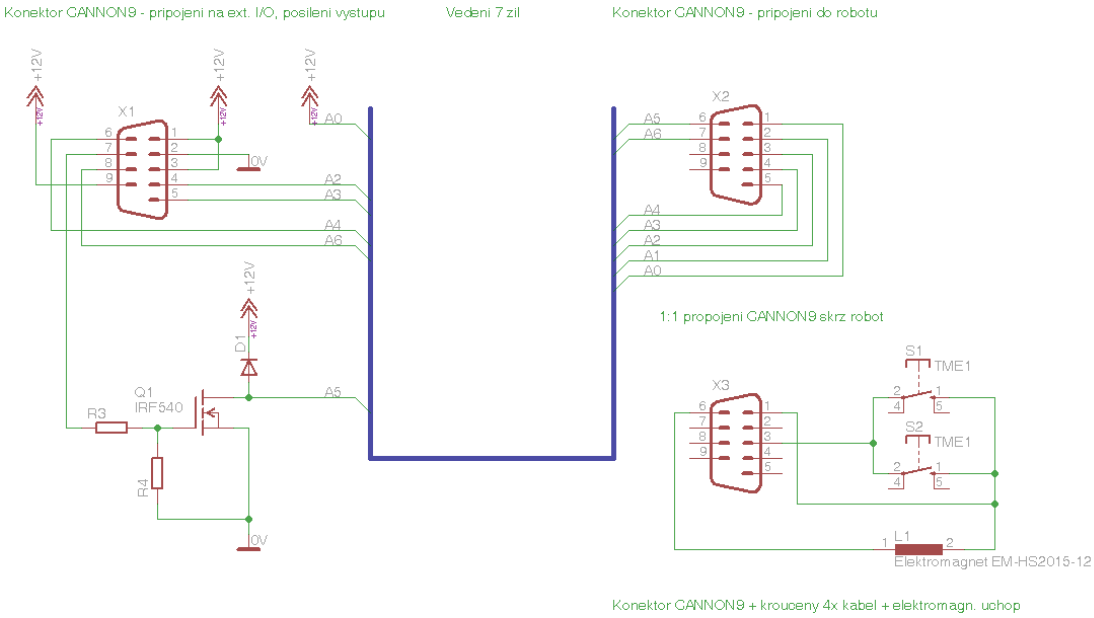
Pro účely hry dámy byl zvolen malý elektromagnet od firmy INTERTEC COMPONENTS - ITS-MS-2015-12VDC, který udrží předmět o hmotnosti 2 kg. Napájen je napětím 12V s proudovým odběrem cca 200 mA [12].

Elektromagnet bohužel nelze spínat přímo pomocí binárních výstupů na jednotce RC180, protože každý výstup lze proudově zatížit maximálně 100 mA. Proto je nutné posílit výstup pomocí zesilovače s tranzistorem, nejlépe s technologií MOSFET pro malý spínací odpor. Elektromagnet je vhodné doplnit i o koncový spínač, který kontroluje, jestli byl kámen skutečně uchycen nebo ne. Provedení spínače bylo zvoleno jako posuvný prstenec okolo elektromagnetu

Obr. 33 - Přídržný elektromagnet ITS-MS-2015-12VDC



Obr. 34 - Mechanická část uchopovacího mechanismu



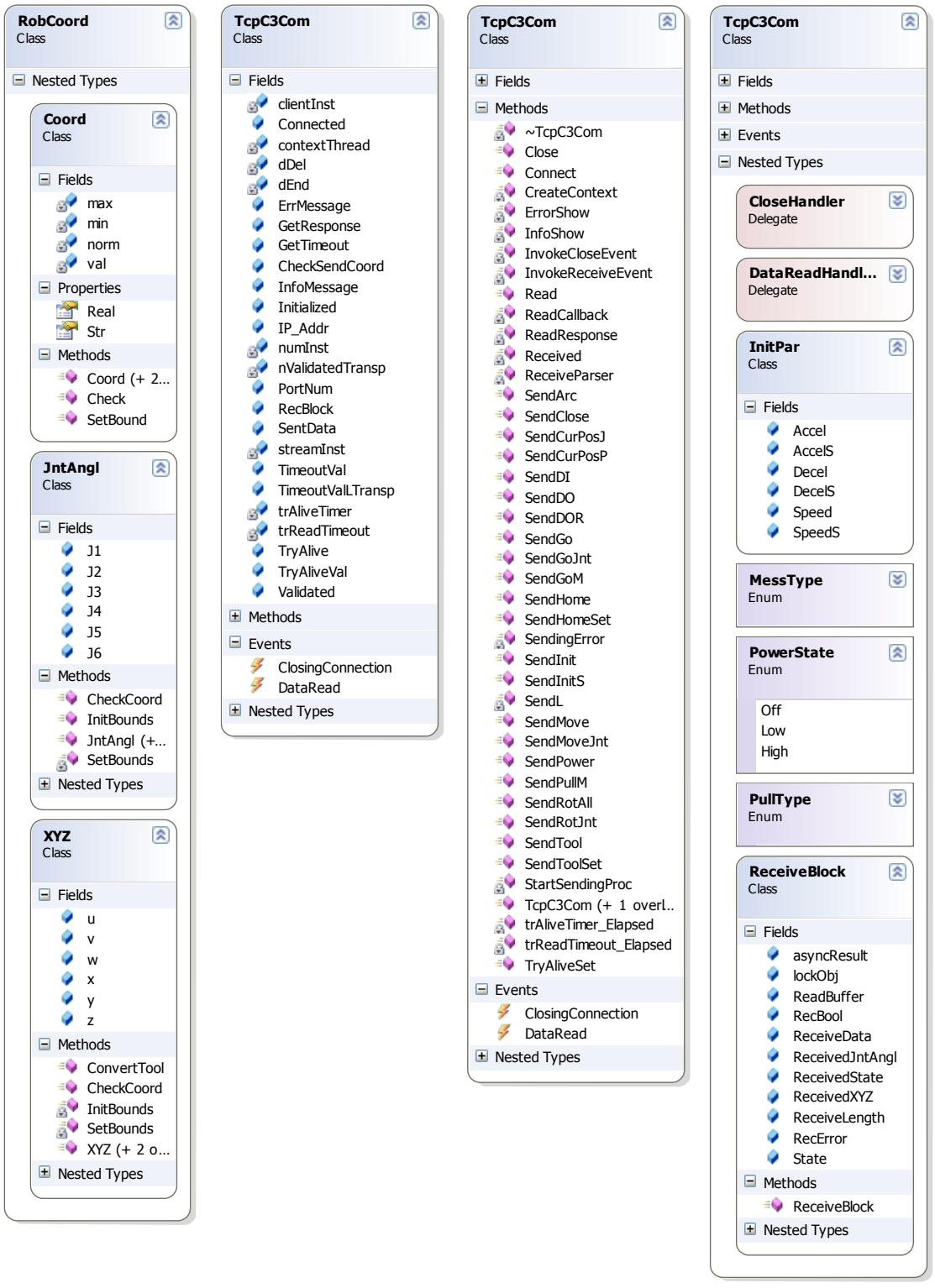
Obr. 35 - Elektronická část uchopovacího mechanismu

Vyvedeni Ext. I/O z konektoru IO			Prodloužení a zesílení			Konektor back robot	Konektor el. magnet
DB50		kabel	DB9		kabel	DB9	kabel
	+12V	červená	1	spojeno			
	GND	fialová	2		modrá	2	
33	Common output 8 to 15	modrá	3	spojeno			
19	Input 8	černá	4		bílá	3	bílá
20	Input 9	šedá	5		růžová	4	
21	Input 10	bílá	6		žlutá	5	
29	Output 8	oranžová	7	báze gate			
30	Output 9	hnědá	8		zelená	7	
	+12V	žlutá	9		hnědá	1	hnědá + žlutá
				Output 8 zesílený	šedá	6	zelená

Tab. 4 - Značení žil v kabelech uchop. mechanismu

3.5 KNIHOVNA V C# PRO ŘÍZENÍ MANIPULÁTORU

Pro komunikaci s jednotkou RC180 byla vytvořena knihovna v jazyce C#. Knihovna komunikuje pomocí standardní třídy TcpClient ze jmenného prostoru System.Net.Sockets založená na soketovém přístupu. Na jednotce se pomocí ovládacího panelu spouští zaváděcí program. V okamžiku, kdy uživatel stiskne tlačítko Prog1, se spustí komunikační funkce. Uvnitř funkce se spustí v jednotce TCP server, který pak naslouchá a čeká na připojení TCP klienta přes rozhraní Ethernet. IP adresa serveru pro testovací účely byla nastavena na 10.0.41.10 a TCP port 2000.



Obr. 36 - Diagram tříd ve jmenném prostoru EpsonC3Com

Třída RobCoord vytváří datovou strukturu pro ukládání a práci s uživatelskými body, souřadnicemi, pozicemi os, nástroji a jiné. Třída TcpC3Com obstarává veškeré metody a parametry potřebné pro komunikaci s jednotkou. TcpC3Com obsahuje i třídu ReceiveBlock, která slouží k ukládání a zpracování přijatých dat po TCP.

3.5.1 Třída RobCoord

Třída RobCoord pouze zapouzdřuje souřadnicové podtřídy. Třída Coord tvoří strukturovaný základní datový prvek pro třídy JntAngl a XYZ. Představuje jednu souřadnici nebo osu.

3.5.1.1 Třída Coord

Ke každé souřadnici je výhodné uložit kromě její hodnoty také povolené meze. V případě, že je vložená hodnota mimo meze a je zapnuta normalizace, jako aktuální se uloží hodnota normovaná. Normovaná hodnota slouží i jako inicializační.

- Parametr Real - pomocí Set a Get odpovídá hodnotě souřadnice ve formátu Double.
- Parametr Str – hodnota v reprezentaci String.
- Metoda Check(bool _norm) – vrací logickou 0, při hodnotě mimo meze, při _norm = 1 zároveň nahradí aktuální hodnotu normovanou.

3.5.1.2 Třída XYZ

Reprezentuje všech 6 souřadnic k popisu koncového bodu v prostoru. Každá ze souřadnic je typu Coord.

- Metoda CheckCoord – kontroluje a případně normalizuje všechny souřadnice
- Metoda ConvertTool – mění polaritu některých souřadnic určených pro offset nástroje, aby lépe odpovídaly bázovým souřadnicím manipulátoru
- Metoda InitBounds – datově nastavuje meze pro všechny souřadnice

3.5.1.3 Třída JntAngl

Každá souřadnice reprezentuje jednu osu manipulátoru ve stupních. Pro každou souřadnici platí stejné charakteristiky jako u třídy *XYZ*.

3.5.2 Třída TcpC3Com

Pro obsluhu jednoho komunikačního kanálu slouží jediná instance třídy TcpC3Com. Hlavním parametrem třídy je jedna instance clientInst typu TcpClient. Přes ni je uskutečňována veškerá komunikace.

Standardní třída TcpClient používá pro přímý styk s rozhraním NetworkStream, jehož instance streamInst je i další důležitý parametr třídy TcpC3Com. Třída TcpClient zaštiťuje celé spojení se serverem pomocí otevření kanálu, vytvoření komunikační NetworkStream a zároveň celé spojení také uzavírá. NetworkStream se stará o samotnou výměnu dat, tedy odesílá data k serveru a naopak přijímá přicházející data ze serveru.

3.5.2.1 Uskutečnění spojení, metoda SendConnect()

Při zavolání metody Connect třídy TcpC3Com se pokusí klient navázat spojení se serverem. Využívá k tomu stejnojmennou metodu Connect() třídy TcpClient. Parametry serveru jako IP adresa a číslo komunikačního TCP portu se získávají z vlastností třídy. Ty lze nastavit buď pomocí konstruktoru nebo přímou úpravou vlastností. Jestliže se spojení podařilo navázat, nastaví se vlastnost Connected na true. Platnost této vlastnosti může být pravidelně kontrolována při aktivním flagu TryAlive viz. odst. 3.5.2.21. V případě, že se spojení nepodaří navázat, neaktivní vlastnost Connected zabrání ve spouštění dalších komunikačních metod a generuje se chybová výjimka. Pro úspěšné navázání musí v jednotce běžet Tcp server (viz. úvod kap. 3.5).

3.5.2.2 Ukončení spojení, metoda Close() a SendClose()

Opět se využívá stejnojmenné metody z třídy TcpClient. Bezpečné ukončení spojení inicializuje i destruktor třídy a také se volá v situaci, kdy zpětná potvrzující zpráva překročila dobu TimeoutVal. Po bezpečném uzavření komunikačního kanálu jednotka RC180 přejde do smyčky zaváděcího programu a pro nové spojení je nutné znovu spustit Prog1. Metoda SendClose je připravena pro bezpečné ukončení TCP Serveru. Program jednotky skočí do zaváděcí smyčky.

Klient – příkaz	!	Exit	LF
Server - odpověď	<	Exit	LF

Tab. 5 - Komunikační datagram - uzavření spojení

3.5.2.3 Základní principy komunikačního protokolu

Komunikační protokol je postaven na předávání jednoduchých nešifrovaných ASCII zpráv ukončených předem nastaveným oddělovačem.

- Komunikaci vždy inicializuje klient.
- Zpráva od klienta může začínat buď znakem '!', který uvozuje příkaz, na který server neodpovídá žádnými daty pouze vrátí ten stejný příkaz s počátečním

znakem ‘<’. Nebo může začít znakem ‘?’, což značí, že jednotka kromě vráceného příkazu musí dodat i vyžádaná data.

- Oddělovač mezi samotnými daty je znak mezera ‘ ’.
- Po přijmutí zprávy jednotkou dochází k parsování a v případě, že formát příchozího požadavku je známý, provede se požadovaná akce. V případě, že při vykonávání dojde k poruše manipulátoru nebo příkaz nemůže být díky chybě vykonán, je namísto echa zpětně odeslána zpráva ve formátu ‚E_cisloChyby’. Číslo chyby se ukládá v ReceiveBlock do proměnné RecError. Po chybách souvisejících s komunikací jednotka zpětně nic neposílá. U některých nezávažných chyb (jako například nekompatibilní formát příchozích dat) jednotka příchozí zprávu ignoruje a čeká na nový příkaz. U chyb závažných se jednotka bezpečně dostane do programu zavaděče. Seznam chyb jednotky, jejich čísla a popis je přístupný na konci dokumentace [3].
- Po vykonání požadovaného příkazu jednotkou pošle zpětně echo zprávy ve formátu „<prikaz“, nebo v případě dotazovacího příkazu „<prikaz data1 data2...“. Klient okamžitě po odeslání začne asynchronně naslouchat. Jakmile přijde zpětně ucelená zpráva, spustí se parser, který ověří, že přijatá zpráva je kompatibilní s odeslanou. Jestliže přišla zpráva s daty, pokusí se správným formátováním data rozdělit a uložit do vhodné struktury (RobCoord).
- Jestliže je aktivní flag Timeout, spolu s čekáním na zpětnou zprávu běží časovač timeoutu. V případě, že zpráva nedorazí v akceptovatelné době, vyhlásí klient výjimku a bezpečně ukončí spojení.

3.5.2.4 Čtení zpětné zprávy od serveru podrobně

Doručení a provedení každého odeslaného příkazu do jednotky je ověřeno pomocí zpětné zprávy. Ta v sobě nese pouze část s názvem odeslaného příkazu.

Po odeslání každé zprávy se spustí metoda ReadResponse(), která při aktivním flagu GetResponse(povoluje potvrzování zpráv) vytvoří nové vlákno pro asynchronní čtení BeginRead(). Zároveň spustí časovač pro stopování timeoutu. Díky rozdělní do vláken můžou fungovat obě metody paralelně. Jakmile přijdou ke klientovi ucelená data, ukončí se otevřené asynchronní vlákno a spustí se parser, který zkontroluje, zda-li přijatá data jsou kompatibilní odpovědí na vyslaný příkaz. Když data nepřijdou do maximální doby TimeoutVal, celé spojení se serverem se ukončí, protože není garantované, v jakém stavu se manipulátor nachází. Obsahuje-li zpětná zpráva dotazovaná data, pokusí se parser data roztřídit do struktury RobCoord:XYZ (ReceivedXYZ), RobCoord:JntAngl (ReceivedJntAngl), případně do binární proměnné (RecBool). Všechny struktury pro ukládání příchozích dat jsou součástí ReceiveBlocku. Vnější program obsluhující třídu TcpC3Com je o okamžiku zpracování zpětného hlášení informován pomocí eventu DataRead. Je-li odpověď od serveru kompatibilní s odeslaným příkazem nastaví se flag Validated na aktivní.

3.5.2.5 *Inicializace jednotky a výkon motorů, metoda SendInit a SendPower*

Pod pojmem inicializace se rozumí nastavení požadovaných rychlostí a zrychlení pohybu manipulátoru a dále také výkon motorů. Metoda je přetížená a umožňuje provést inicializaci s výchozími parametry v jednotce. U přetížené metody s parametry je umožněno rychlosti a zrychlení předat pomocí struktury InitPar. V případě úspěšného odeslání vrací logickou 1. Hodnota parametru Power – High je přenášena jako 2, Low jako 1 a Off jako 0. Ostatní parametry jsou předávány jako desetinná čísla převedena na řetězec String. Pomocí metody SendPower lze přepínat pouze výkon motoru. Ovládání je shodné jako u příkazu SendInit.

Klient – příkaz	!	Init	LF
Server - odpověď	<	Init	LF

Tab. 6 - Komunikační datagram - inicializace spojení bez parametrů

nebo

Klient	!	Init	LF	_Power		_Speed		_Accel		_Decel		_SpeedS		_AccelS		_DecelS	LF
Server	<	Init	LF														

Tab. 7 - Komunikační datagram - inicializace spojení s parametry

3.5.2.6 *Příkaz Go, metoda SendGo*

Vstupní parametry pro metodu jsou uloženy v instanci třídy RobCoord:XYZ. Posílány jsou všechny 3 osy, tedy X, Y, Z a natočení U, V a W.

Klient	!	Go	LF	_X		_Y		_Z		_U		_V		_W	LF
Server	<	Go	LF												

Tab. 8 - Komunikační datagram - pohyb Go

3.5.2.7 *Příkaz Move, metoda SendMove*

Obdobné jako zpráva metody *SendGo*.

Klient	!	Move	LF	_X		_Y		_Z		_U		_V		_W	LF
Server	<	Move	LF												

Tab. 9 - Komunikační datagram - pohyb Move

3.5.2.8 *Příkaz Arc3, metoda SendArc*

Metoda pro najetí na pozici po křivce proložené středním bodem. Označení *f* značí finální pozici, *m* střední bod. Mezi oběma pozicemi je v datagramu běžná mezera.

Klient	!	Arc	LF	_Xf		_Yf		_Zf		_Uf		_Vf		_Wf	
				_Xm		_Ym		_Zm		_Um		_Vm		_Wm	LF
Server	<	Arc	LF												

Tab. 10 - Komunikační datagram - pohyb Arc3

3.5.2.9 *Příkaz GoJnt/ MoveJnt pro absolutní natočení os, metoda SendGoJnt/SendMoveJnt*

Umožňuje přesouvat manipulátor do přesné pozice udané absolutním natočením každé osy. Přesun může být typu PTP s příkazem Go nebo po přímce s příkazem Move.

Klient	!	GoJnt	LF	_J1		_J2		_J3		_J4		_J5		_J6	LF
Server	<	GoJnt	LF												

Tab. 11 - Komunikační datagram - absolutní osový pohyb Go

nebo

Klient	!	MoveJnt	LF	_J1		_J2		_J3		_J4		_J5		_J6	LF
Server	<	MoveJnt	LF												

Tab. 12 - Komunikační datagram - absolutní osový pohyb Move

3.5.2.10 *Příkaz GoM pro nakládání/skládání materiálu, metoda SendGoM*

Příkaz určený k naložení a složení přitahovaného materiálu pomocí elektromagnetu. Manipulátor se nejprve přesune na požadovanou pozici a zde podle bitu typeCarry buď sepne (hodn. 1) nebo rozepne elektromagnet (hodn. 0). Na místě setrvá na 0.5 sekundy a následně se manipulátor zvedne v ose z o 10 mm, aby zkontroloval naložení nebo vyložení materiálu. Jestliže koncovým spínačem není zjištěn vyžadovaný stav, předá se zpětným příkazem chybový bit infoStat s hodnotou 1. Chyba může nastat v případě, že se nepodařilo manipulátor naložit nebo vyložit.

Klient	!	GoM	LF	_typeCarry		_X		_Y		_Z		_U		_V		_W	LF
Server	<	GoM	:	_infoStat		LF											

Tab. 13 - Komunikační datagram - nabírání materiálu GoM

3.5.2.11 *Příkaz PullM pro přesun materiálu, metoda SendPullM*

Příkaz určený k přesunu materiálu pomocí elektromagnetu z jednoho místa na druhé. Celá sekvence pohybu je podobná jako příkaz Jump3. Manipulátor se nejprve přesune pomocí příkazu Go nad výchozí pozici A do výšky H. Změní pohyb na MoveM (obdoba s GoM) a podle typu příkazu typeCarry buď materiál naloží (hodn. 1) nebo vyloží (hodn. 0). Opět se přesune (Move) do výšky H a dále nad koncovou pozici B (Go). Zde provede inverzní operaci k nakládání/skládáni. Úhly U, V a W jsou stále stejné. Jestliže kdykoliv při nakládání/skládání nebude koncový spínač ve správné poloze zahlásí se porucha pomocí bitu infoStat

Klient	!	PullM	LF	_typeCarry		_Xa		_Ya		_Za		_U		_V	_W		_H		_Xb		_Yb		_Zb	LF
Server	<	PullM	:	_infoStat		LF																		

Tab. 14 - Komunikační datagram - přesun materiálu PullM

3.5.2.12 Příkaz pro relativní pohyb jedné osy, metoda SendRotJnt

Po přijetí příkazu jednotka provádí relativní pohyb pomocí funkce JTran, tedy otočení osy o relativní úhel ve stupních.

Klient	!	RotJnt	LF	_numJnt		_rotVal	LF
Server	<	RotJnt	LF				

Tab. 15 - Komunikační datagram - relativní rotace jedné osy

3.5.2.13 Příkaz pro relativní pohyb všech os, metoda SendRotAll

Na rozdíl od SendRotJnt se postupně s osami J1 až k ose J6 provádí relativní natočení. Vstupem do metody je objekt typu RobCoord:JntAngl.

Klient	!	RotAll	LF	_J1		_J2		_J3		_J4		_J5		_J6	LF
Server	<	RotAll	LF												

Tab. 16 - Komunikační datagram - relativní rotace všech os

3.5.2.14 Příkazy nastavení a pohyb k výchozí pozici, metoda SendHomeSet a SendHome

Pomocí metody SendHomeSet se nastavují úhly natočení pro výchozí pozici. Pořadí, v jakém natáčení probíhá, je určeno nastavením robotu v jednotce. Jednotka si zároveň zajišťuje převod mezi úhly ve stupních na úhly v inkrementačních pulsech, které funkce HomeSet v jazyce SPEL+ vyžaduje. Metoda SendHome přesune manipulátor do výchozí pozice.

Klient	!	Home	LF
Server	<	Home	LF

Tab. 17 - Komunikační datagram - přesun na pozici Home

a

Klient	!	HomeS	LF	_J1		_J2		_J3		_J4		_J5		_J6	LF
Server	<	HomeS	LF												

Tab. 18 - Komunikační datagram - nastavení pozice Home

3.5.2.15 Příkazy nastavení offsetu nástroje, metody SendToolSet a SendTool

Velikost jednotlivých souřadnic offsetu nástroje od koncového bodu manipulátoru se předává pomocí objektu třídy RobCoord:XYZ. Pro snazší představu jsou souřadnice všech os přepočítány na kladné přírůstky v kladném směru od koncového bodu robotu. Směr přírůstků je při pozici robotu jako na [Obr. 16] stejný jako základní souřadnice.

Klient	!	Tool	LF	_numTool	LF
Server	<	Tool	LF		

Tab. 19 - Komunikační datagram - použití zvolené sady nástroje

a

Klient	!	ToolS	LF	_numTool		_X		_Y		_Z		_U		_V		_W	LF
Server	<	ToolS	LF														

Tab. 20 - Komunikační datagram - nastavení konkrétní sady nástroje

3.5.2.16 Dotaz pro čtení aktuální pozice v prost. souřadnicích, metoda SendCurPosP

V jakémkoliv okamžiku lze poslat jednotce dotaz na její aktuální pozici. Ta navrátí svou pozici v prostorových souřadnicích a následně se převedou do struktury RobCoord:XYZ.

Klient	?	CurPosP	LF												
Server	<	CurPosP	LF	_X		_Y		_Z		_U		_V		_W	LF

Tab. 21 - Komunikační datagram - čtení aktuálních prostorových souřadnic

3.5.2.17 Dotaz pro čtení aktuálního natočení os, metoda SendCurPosJ

Obdobně jako u SendCurPosP se navrací natočení každé osy a následně se data převedou do struktury RobCoord:JntAngl.

Klient	?	CurPosJ	LF												
Server	<	CurPosJ	LF	_J1		_J2		_J3		_J4		_J5		_J6	LF

Tab. 22 - Komunikační datagram - čtení aktuálního natočení os

3.5.2.18 Dotaz na probíhající pohyb, metoda SendCurPosJ

Jestliže se robot již nehýbe a dojel na požadovanou pozici, vrátí log. 1, ve všech ostatních situacích log. 0.

Klient	?	InPos	LF		
Server	<	InPos	LF	_notMoving	LF

Tab. 23 - Komunikační datagram - informace o probíhajícím pohybu

3.5.2.19 Dotaz na stav digitálního vstupu, metoda SendDI

Příkaz slouží k přečtení hodnoty jednoho standardního binárního vstupu.

Klient	?	DI	LF	_numDI	LF
Server	<	DI	LF	_valDI	LF

Tab. 24 - Komunikační datagram - přečtení zvoleného digitálního vstupu

3.5.2.20 Příkaz pro nastavení digitálního výstupu, metoda SendDO

Příkaz pro set (log. 1) nebo reset jednoho standardního digitálního výstupu.

Klient	?	DO	LF	_numDI		_valDO	LF
Server	<	DO	LF				

Tab. 25 - Komunikační datagram -nastavení zvoleného digitálního výstupu

3.5.2.21 Kontrola průchodnosti spojení

Vzhledem k tomu, že se statusem flagu Connected podmiňuje další komunikace, je nutné znát skutečnou průchodnost komunikačního kanálu. Pro tyto účely se při aktivním flagu TryAlive pravidelně v časech klidu na sběrnici a v definovaných intervalech testuje odpověď na jednoduchý příkaz. Jako informace o průchodnosti rozhraní samozřejmě slouží i běžný funkční příkaz a jeho odpověď. Jakmile zpětné hlášení nedorazí do snížené doby TimeoutValTransp, změní se flag Connected na neaktivní.

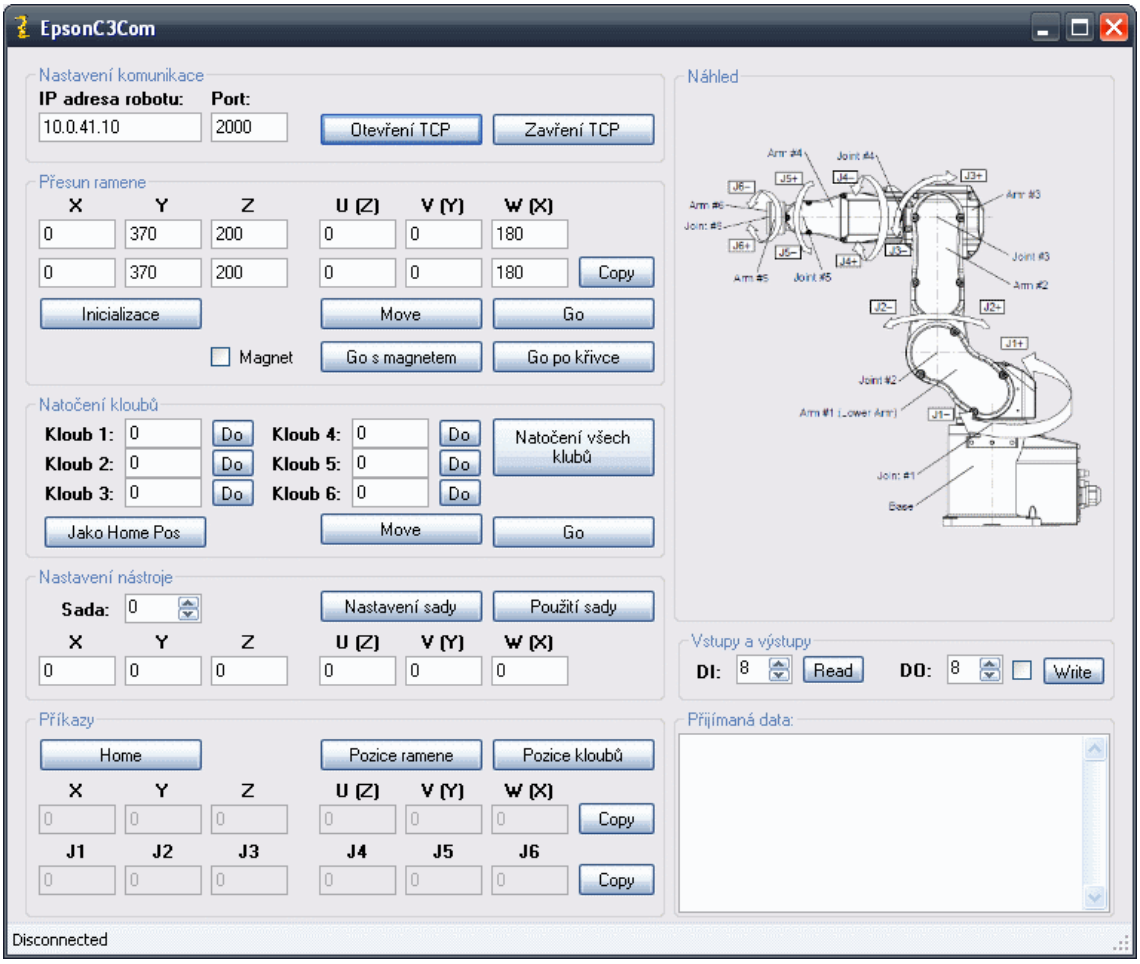
Klient	?	L	LF
Server	<	L	LF

Tab. 26 - Komunikační datagram - kontrola průchodnosti spojení

3.5.3 Testovací uživatelský program EpsonC3Com

Pro otestování funkcí knihovny v C# byl vytvořen uživatelský program, jehož hlavní částí je uživatelské grafické rozhraní. Otestovat lze téměř všechny poskytované metody a funkce.

Po zadání správných parametrů komunikace (IP adresa a číslo portu serveru) a klepnutí na tlačítko Otevření TCP, by se ve spodním status baru měl změnit stav Disconnected na Connected. Hodnota se získává z flagu Connected třídy TcpC3Com. Aby bylo možné hýbat s robotem, je nutné provést Inicializaci - pošle SendInit bez parametrů, použijí se výchozí hodnoty. Dále je už možné plně s robotem manipulovat. V pravé části uživatelského okna se zobrazují data s časovou hlavičkou, které přicházejí směrem od manipulátoru. Při jakýchkoliv poruchách či chybách se zobrazují varovná hlášení formou vyskakovacích oken.



Obr. 37 - Okno uživatelského rozhraní EpsonC3Com

3.6 KOMPLETNÍ ŘÍDÍCÍ PROGRAM

Výsledný řídící program je složen z nezávislých dílčích projektů jako je snímání z kamery a počítačové vidění, ovládacího a komunikačního rozhraní s manipulátorem a logiky samotné hry. Naprogramování a odladění spolehlivé logiky hry, obsahující části jako virtuální model hry a algoritmus pro výpočet tahů protihráče, by bylo časově velice náročné. Z toho důvodu byl vyhledán podobný otevřený projekt, který by se svými vlastnostmi co nejvíce blížil potřebám robotického hráče dámy.

Projekt „Checkers for Pocket PC using a Recursive Min-Max Tree“ od autora Leonardo Paneque lze s určitými úpravami snadno úloze přizpůsobit. Projekt je sice sestaven jako program pro mobilní platformu Windows Mobile, ale převedení do plnohodnotné Windows aplikace nepředstavuje významný problém. Projekt byl sestaven v prostředí Microsoft Visual Studio a jazyce C# pro softwarovou strukturu .NET Compact Framework, která je součástí .NET Framework, z čehož vyplývá i praktická shodnost kódu na obou platformách.

3.6.1 Struktura původního programu Checkers

Celý projekt Checkers se skládá z podprojektů GamesPackage a BoardEngine. Třídy těchto 2 podprojektů vytváří abstraktní základ pro různé deskové hry. Až podprojekt Checkers upravuje obecný model přímo pro hru dáma. Třídy, které se týkají dámy a které dědí vlastnosti jejich obecných předků, mají ve svém názvu přidáno slovo „Checkers“.

3.6.1.1 Podprojekt GamesPackage

GamesPackage je balík s nástroji, které vytvářejí abstraktní rozhraní pro řízení hry. Hlavní součástí balíku je děděná třída Moderator. Ta slouží jako šablona spouštěných událostí, které proběhnou po té, co jeden z hráčů označí svůj tah za ukončený. Dále balík definuje obecnou třídu hráče a delegáty událostí jako je konec hry nebo změna hráče na tahu.

3.6.1.2 Podprojekt BoardEngine

Jedná se o balík, který ovládá samotnou hrací desku.

Abstraktní třída Piece definuje paměťovou strukturu, která popisuje jeden hrací kámen na šachovnici. Součástí popisu je souřadnice pole na šachovnici, na kterém se kámen nachází. Třída BoardPosition slouží jako struktura pro ukládání dvojice souřadnic na hrací ploše. Typ PieceColor definuje 2 barvy kamenů hráčů.

Třída Board vytváří strukturu pro obsluhu hrací desky. Součástí je paměťový prostor pro každé z 8 x 8 polí šachovnice. Jestliže se na konkrétním poli šachovnice nachází kámen, do příslušného prvku struktury Board se vloží odkaz na paměťovou buňku typu Piece.

3.6.1.3 Podprojekt Checkers

Základem hry jsou dvě instance typu IPlayer, které reprezentují oba hráče partie. Hráči mohou být typu HumanCheckersPlayer nebo SimpleCheckersPlayer, přičemž SimpleCheckersPlayer si vypočítává svůj optimální tah pomocí algoritmu Minimax. Každý hráč si obstarává posloupnost kroků, které v rámci jednoho tahu musí provést. Konkrétní hráč signalizuje vnějšímu řízení pomocí událostí změny jako konec tahu, konec výpočtu tahu nebo hráč nyní na tahu. Jeden objekt třídy CheckersModerator moderuje celou partii. Postupně přiděluje hráči právo být na tahu, vyhodnocuje konec hry, hru pozastavuje nebo se vrací k předchozím uloženým stavům.

Společná pro oba hráče je instance hrací desky CheckersBoard. Na konkrétních polích desky jsou uloženy odkazy hracích figur třídy CheckersPiece. Každá figura jako potomek CheckersPiece je buď typu Pawn (pěšec) nebo Queen. Každý typ definuje svá pravidla pro pohyb na šachovnici. Hráč při vykonávání svého tahu přemisťuje, přidává nebo odstraňuje figury na společné šachovnici. Na začátku tahu sestaví

automaticky každá figura na šachovnici seznam povolených tahů, které lze s figurou vykonat. Do výpočtu tahů je zakomponována i povinnost skoku.

O vykreslení aktuální podoby šachovnice se stará třída BoardDrawer.

3.6.2 Úpravy původního projektu s logikou hry

Díky striktnímu oddělení modelu a řízení hry od uživatelského rozhraní lze snadno převést projekt mezi platformami. Jediné rozdílné části mezi mobilní a desktop platformou jsou grafické prvky formuláře uživatelského rozhraní. Po jejich konverzi lze spustit původní program na desktop platformě.

Česká pravidla dámy definují, že první pole na šachovnici vlevo je u každého hráče tmavé. V původním programu na tomto místě bylo pole světlé. Úpravou v třídě BoardDrawer je nutné změnit vykreslování šachovnice a úpravou v inicializační metodě třídy CheckersBoard změnit počáteční pozice kamenů po zahájení hry.

3.6.3 Sekvence provedení tahu obou hráčů

V programu je člověk označen za hráče s bílou barvou typu HumanCheckersPlayer (dále uváděný jako hráč) a robot za hráče s černou barvou typu SimpleCheckersPlayer (dále uváděný jako protihráč). Změna fyzické barvy kamenů je v uživatelském rozhraní upravitelná.

3.6.3.1 Provedení tahu člověka

Nejprve je hráč informován o konci tahu protihráče pomocí události OnOtherPlayerMovePerformed. V takovém okamžiku se obnoví zobrazení hrací desky. Jestliže byl tah správný, je vzápětí vyvolána i událost OnPlay.

Uvnitř modelu rozehrané partie se automaticky sestaví seznam povolených tahů pro každý kámen hráče na desce. Spustí se také metoda SaveStateAfterBlackPlayed(), která pomocí počítačového vidění rozezná kameny na desce a uloží je do hlavní struktury boardVisOld a podstruktur visBoardOld a visTableOld.

Detekce kamenů na desce s použitím počítačového vidění probíhá v následujících krocích. Nejprve se zkopíruje poslední získaný snímek ze zapnuté kamery. Tento snímek se předá počítačovému vidění pomocí metody ProcessImage, která spouští postupně veškeré procedury detekce. První prováděnou procedurou je nalezení kalibračních rohů modrého orámování dle kap. 2.2.3 a následná geometrická transformace pro kompenzaci perspektivního zkreslení. Dále je spuštěno vyhledávání červených kamenů na šachovnici. Kameny se hledají i mimo šachovnici na odkládacích pozicích. Rozhodnutí, zda je kámen typu pěšec nebo dáma, se provádí podle množství zastoupení černé barvy na pixelech okolo nalezeného středu. Pro každý střed kamene se určí i korekce perspektivního snímání výšky, která se přičítá k nalezeným souřadnicím středu podle její pozice na desce. Totožná detekce se spustí i pro kameny zelené barvy. Výstupem z obou procedur jsou 2 seznamy nalezených

figur s jejich středy na hrací desce, typem a barvou kamene, které se sloučí do jedné výstupní struktury TableData. Podle známé pozice šachovnice na hrací desce se sdružený seznam doplní o položku souřadnic polí na šachovnici u těch figur, které leží uvnitř šachovnice. Seznam TableData slouží jako výstup z počítačového vidění.

Seznam figur na desce TableData se mimo počítačové vidění ukládá do seznamu visTableOld. Seznam visBoardOld vychází z visTableOld, přičemž kopíruje pouze kameny, které jsou umístěny na šachovnici. Seznam s kameny na šachovnici se také převádí do boardVisOld typu CheckersBoard, který v sobě obsahuje automatické generování možných tahů. Objekt šachovnice se doplní i o objekt BoardAbsPos, který v sobě ukládá pro každou figuru na desce její absolutní pozici.

Nyní program čeká na okamžik, než hráč provede svůj tah a potvrdí jej pomocí příslušného tlačítka. V tento okamžik se zkopíruje aktuální snímek a proběhne stejný proces rozpoznávání a ukládání jako na začátku tahu s tím rozdílem, že názvy objektů mají nyní místo přípony Old příponu New. Odehraný tah hráče se zjistí pomocí metody ReconstrMoveFromVision, která porovnává šachovnice na začátku a konci tahu (boardVisOld a boardVisNew). Součástí je i kontrola, zda bylo možné tah zahrát, jestli hráč odstranil přeskočené kameny protihráče, jestli vyměnil na konci šachovnice pěšce za dámu či zda se na šachovnici neodehrálo příliš mnoho změn. Jestliže je tah vyhodnocen jako korektní, zavolá hráč vlastní metodu MakeAMove se svým tahem ve formátu CheckersMove. Metoda nejprve provede změny z tahu na sdílené šachovnici MyBoard a potom pomocí CheckersModeratora předá právo tahu svému protihráči.

3.6.3.2 Provedení tahu robotu

Robot nejprve při události OnOtherPlayerMovePerformed obnoví zobrazovanou šachovnici a také spustí vypočet ideálního tahu pomocí metody Minimax. Až jsou výše uvedené kroky hotovy, zavolá se událost OnPlay. Při obsluze události se spustí sekvence procesů, při nichž robot provede tah na desce. Počítačové vidění se nyní nespouští, ale vychází se ze zaznamenaného stavu šachovnice v objektu boardVisNew z konce tahu hráče. Protože robot pracuje i s odkládacími plochami pro kameny po stranách šachovnice, převede se seznam nalezených kamenů v visTableNew do objektů checkersPlantWhite a checkersPlantBlack typu CheckersPlant. Tato třída třídí nalezené kameny do jednotlivých odkládacích pozic, vyhledává volná pole pro odkládání nebo hledá mezi kameny dámu.

Sekvence procesů, které provádí robot, se řeší v metodě StateAutomatRobotMove. Metoda funguje jako stavový automat. Nejprve je volána při spuštění tahu, kdy se metodě předá nejlepší nalezený tah a následně si jí volá robot po provedení jednotlivých procesů. Se spuštěním stavového automatu se otvírá v uživatelském prostředí okno s informací, že robot právě pracuje. Základním procesem je provedení tahu s kamenem za pomocí příkazu PullM. Pozice kamene

v souřadnicích robotu se vypočítá jako základní offset levého horního rohu orámování hrací desky a k němu přičtená pozice kamene z počítačového vidění v rámci hrací desky. Z toho vyplývá, že je nutné, aby hrací deska nebyla jakkoliv vůči souřadnicovým osám X, Y robotu natočená. V případě potřeby by bylo nutné natočení fyzicky měřit a do programu přidat výpočet homogenní transformace.

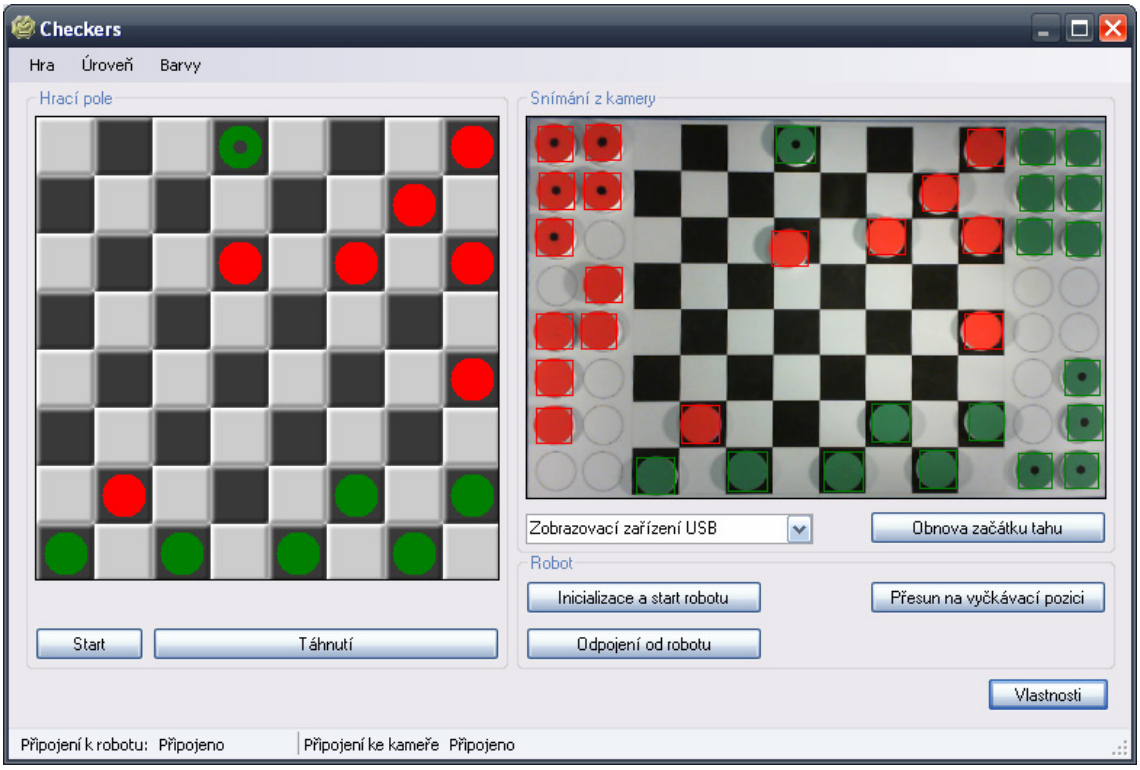
Robot v případě vícenásobného skoku v tahu přesouvá kámen pouze z výchozí pozice do koncové. Pomocí obsluhy události změny z robotu je vyhodnoceno, jestli přesun kamene proběhl v pořádku nebo ne. Jestliže ano, stavový automat přejde na další proces ve stavovém automatu. Jestliže vypočtený tah končí na konci hrací desky (robot získal dámu), nepřesouvá se kámen na koncovou pozici, ale rovnou na první volné místo na odkládací ploše. Dále se vyhledá první dáma a ta se přesune na koncovou pozici. Další proces je odstranění přeskočených kamenů. Robot vždy nalezne pro každý hráčův kámen volnou pozici na jeho odkládací ploše a tam kámen přesune. Závěrečným procesem je pak přesun manipulátoru do výchozí vyčkávací pozice, uzavření informačního okna a provedení samotného tahu na modelu hrací desky.

3.6.4 Uživatelské rozhraní

Rozhraní s vizualizací hry je rozděleno do 3 skupin. Skupina Hrací pole zobrazuje aktuální model hry uložený v MyBoard. Pomocí tlačítek Start se spouští nová partie, kterou lze začít z libovolného rozestavění figur na hrací desce. Pomocí tlačítka Táhnutí se potvrzuje tah hráče. Ve skupině Snímání z kamery se zobrazuje poslední zaznamenaný snímek z kamery, který prošel procesem počítačového vidění. Scéna má kompenzované perspektivní zkreslení a pomocí barvených obdélníků jsou vyznačeny nalezené kameny na hrací desce. Dále skupina obsahuje pole pro výběr kamery a tlačítko, které obnoví začátek tahu člověka na základě nového snímku z kamery (aktualizace objektu boardVisOld). Poslední skupina ovládá robot, tedy jeho spuštění a inicializaci, která uvolní elektromagnetickou brzdu a nastaví rychlosti a zrychlení pohybu. Pomocí tlačítka Přesun na vyčkávací pozici se robot přesune do pozice, kdy necloní snímání hrací desky. Tlačítkem Odpojení od robotu se uzavře spojení mezi řídící jednotkou RC180 a počítačem.

V horním menu je možné přepnout z českých pravidel hry na anglickou dámu nebo změnit úroveň hry protihráče, což u algoritmu Minimax mění hloubku zanoření. Pomocí menu Barvy lze vyměnit barvy kamenů mezi hráčem a protihráčem.

Při stisku tlačítka Vlastnosti se otevře nové okno, ve kterém je možné pro diagnostické účely zobrazit obnovovaný neupravený snímek z kamery nebo zobrazit prahovanou mapu detekovaných černých a červených kamenů nebo okno vlastností ovladače kamery.



Obr. 38 - Uživatelské rozhraní řídícího programu

4. ZÁVĚR

Diplomová práce se zabývá konkrétním použitím 6-osého průmyslového manipulátoru EPSON C3 na demonstrační úloze robotického hráče dámy. Manipulátor se skládá z 6-osého průmyslového robotu Epson C3 a řídící jednotky Epson RC180. Během celého vývoje bylo praxí potvrzeno, že kompletní řešení firmy Epson je značně výkonné, mechanika robotu vyniká vysokou přesností a výhodným uspořádáním. Řídící jednotka je díky integraci veškerých důležitých funkcí připravena pro snadné nasazení přímo na aplikaci, přičemž celý návrh jednotky zajišťuje vysokou bezpečnost a spolehlivost.

Výstupem projektu je robotický hráč dámy, který reaguje na reálnou situaci na hrací desce, a na základě výpočtu vhodného protitahu a pomocí manipulátoru EPSON protitah robot realizuje. Práce postupně popisuje různé možnosti řešení 3 hlavních částí projektu a dále autorovu kompletní úspěšnou realizaci. Mezi hlavní části patří snímání a detekce scény s šachovnicí, sestavení modelu hry a výpočet vhodného protitahu a ovládání manipulátoru, který koná protitahy robotu.

Ve výsledné realizaci je snímání a detekce kamenů na šachovnici řešeno pomocí kamery, jejíž snímky se předávají počítačovému vidění, pomocí něhož jsou všechny hrací kameny rozeznány a určena jejich poloha v prostoru. Pro zjednodušení a zefektivnění detekce je herní materiál upraven barevně, materiálově i uspořádáním. Vyhledávání kamenů a okrajů hrací desky je založeno na jejich významně rozdílných barvách. Principy sestavení scény a vyhledávání hracích kamenů se v praxi ukázaly jako velmi dobře zvolené, protože detekce funguje spolehlivě a značně robustně.

Strategie hry v řídícím programu je částečně převzata z otevřeného projektu Checkers od autora Leonardo Paneque (CPOL licence). Po převedení původního projektu z platformy Windows Mobile na Windows aplikaci se ukázalo jeho uspořádání jako velice výjimečné, modulární a snadno použitelné pro robotickou aplikaci. Projekt je složen z modulárních celků, díky nimž lze pomocí minima úprav uspořádání člověk vs. robot převést na uspořádání člověk vs. člověk nebo robot vs. robot. Taktika hry robotu založená na algoritmu Minimax s alfa-beta ořezáváním staví proti člověku plnohodnotného protihráče, který má nastavitelnou výkonnostní úroveň.

Po ověření všech použitých funkcí jazyka SPEL+ přímo na manipulátoru byl navržen vhodný komunikační protokol mezi jednotkou a PC. Struktura protokolu byla navržena tak, aby poskytovala vývojáři na straně PC univerzální paletu snadno použitelných funkcí pro ovládání s velkým důrazem na bezpečnost provozu. Obsluha protokolu na straně jednotky je definována SPEL+ programem, který po otevření TCP serveru přijímá příkazy a data, provádí je a zpětně informuje komunikační protějšek. To vše s využitím bezpečného chybového a poruchového systému, který protějšek informuje ve většině případů ještě před fyzickou poruchou. Jednotka díky pokročilým

modelům mechaniky detekuje možnou kolizi a zároveň se stará o bezpečné SW spouštění všech funkcí. Aby bylo možné funkčnost knihovny demonstrovat, bylo sestaveno základní grafické uživatelské rozhraní pro testování vzdáleného řízení. Dále byl navržen a zkonstruován ovládací panel, který se připojuje k řídící jednotce, a díky němuž lze ovládat manipulátor bez PC. Do projektu robotického hráče dámy byla jednoduše vložena sestavená komunikační knihovna. Na této praktické úloze se potvrdilo, že knihovna i protokol byly navrženy vysoce účelně.

Mezi náměty k dalšímu vývoji patří důkladné studium již hotových .NET komunikačních knihoven od výrobce EPSON. Knihovny v sobě implementují především monitorovací funkci, ale umožňují také v jednotce manipulátoru spustit uživatelský program, jednotku resetovat a různé další funkce. Spouštění konkrétních příkazů však tímto způsobem realizovat nelze a proto je vhodné knihovnu používat paralelně s knihovnou z této diplomové práce. Jako rozšíření robotického hráče dámy by bylo výhodné přidat funkci pro automatické rozmisťování kamenů pomocí manipulátoru na výchozí pozice na hrací desce.

Do diplomové práce byly zapracovány veškeré zadáním vytyčené cíle a na kompletně funkčním robotickém hráči byl celý návrh úspěšně ověřen.

5. SEZNAM POUŽITÉ LITERATURY

[1] *Epson C3 Compact 6-Axis Robot Manual.* URL:<http://www.robots.epson.com/downloads/manuals/EPSON_C3_Robot_Manual(R4).pdf >[citováno 2012-05-16].

[2] *Micro PowerDrive RC170/RC180 Controller Manual.* URL:<http://www.robots.epson.com/downloads/manuals/EPSON_RC170_RC180_Controller_Manual(R13).pdf>[citováno 2012-05-16].

[3] *Epson RC+5.0 SPEL+Language Reference Rev.2 (for Micro PowerDrive RC170/RC180).* URL:<http://www.robots.epson.com/downloads/manuals/EPSON_SPEL+_Lite_Language_Ref-RC170_RC180(V54R2).pdf>[citováno 2012-05-16].

[4] *Epson RC+ User Guide (for Micro PowerDrive RC170/RC180).* URL:<http://www.robots.epson.com/downloads/manuals/EPSON_RC+50_Users_Guide-RC170_RC180(V54R1).pdf>[citováno 2012-05-16].

[5] SHARP, J. *Microsoft Visual C# 2010* Computer Press. ISBN: 9788025131473

[6] Česká Unie Dámy. *Pravidla české dámy dle České Unie Dámy .* URL:<http://ceskauniedamy.com/dwnld/doc/pravidla/CZ_PRAVIDLA_CD_2006_10_01.pdf>[citováno 2012-05-16].

[7] PANEQUE, L.. *Checkers for Pocket PC using a Recursive Min-Max Tree. 2008.* CPOL licence. URL:< http://www.codeproject.com/Articles/27129/Checkers-for-Pocket-PC-using-a-Recursive-Min-Max-T >[citováno 2012-05-16].

[8] HORÁK, K., KALOVÁ, I., PETIOVSKÝ, P., RICHTER, M. *Počítačové vidění.* URL:< http://www.uamt.feec.vutbr.cz/vision/TEACHING/MPOV/Pocitacove_videni_S.pdf >[citováno 2012-05-16].

[9] FAJT, J. *Geometrické transformace v GIS.* URL:< http://gis.zcu.cz/studium/ugi/referaty/05/GeometrickeTransformace/index.html>[citováno 2012-05-16].

[10] KÜHR, T. *Algoritmus Minimax.* URL:< http://dostal.inf.upol.cz/data/1011/PS1/minimax.pdf>[citováno 2012-05-16].

[11] NĚMEC, J. *Šachové myšlení. 13.dílný seriál. 2006.* URL:< http://dostal.inf.upol.cz/data/1011/PS1/minimax.pdf>[citováno 2012-05-16].

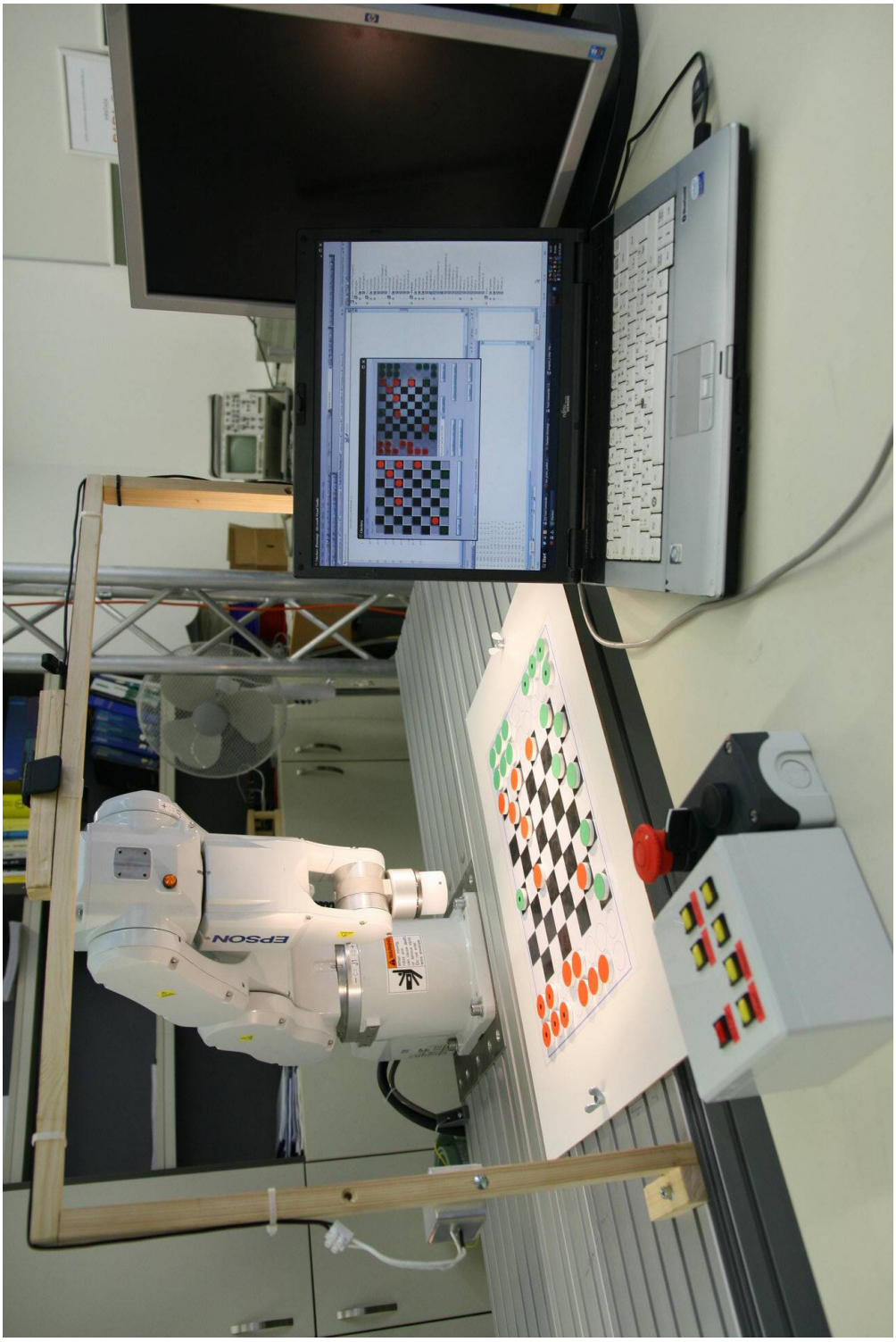
[12] *INTERTEC COMPONENTS ITS-MS-2015-12VDC.* URL:< http://www.tme.eu/cz/details/em-hs2015-12/elektromagnety/intertec-components/its-ms-2015-12vdc>[citováno 2012-05-16].

6. SEZNAM OBRÁZKŮ

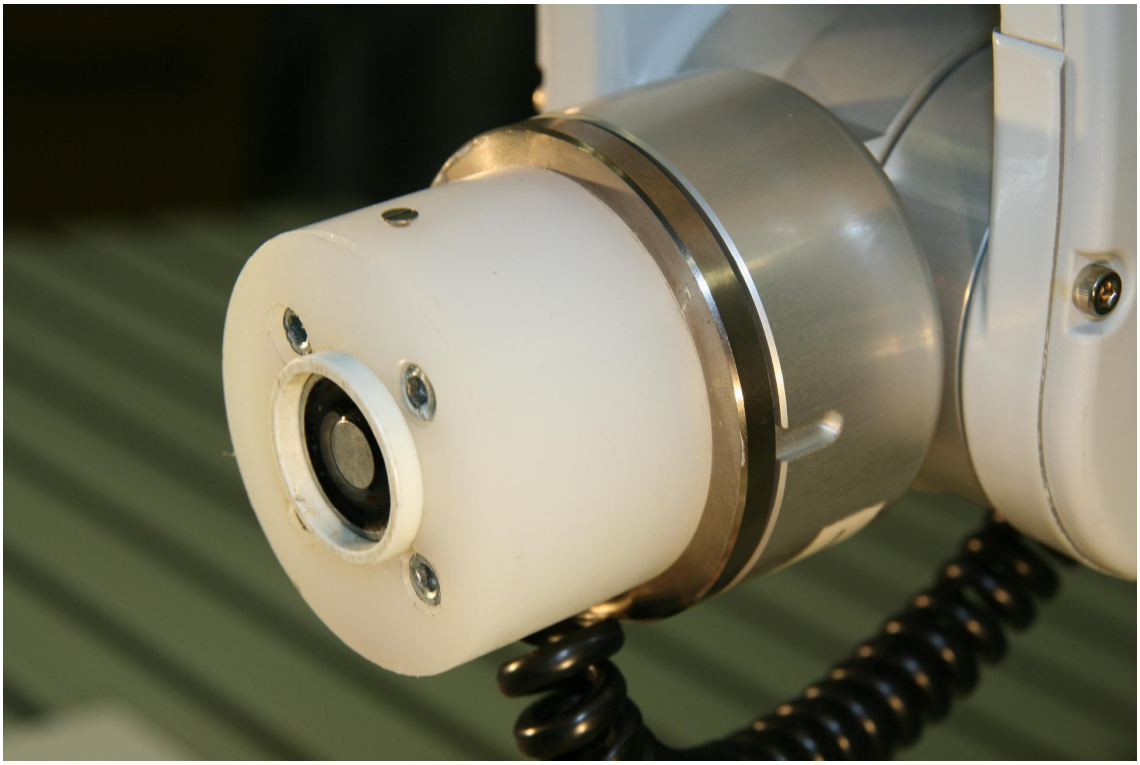
7. PŘÍLOHY

7.1 FOTOGRAFIE DEMONSTRAČNÍ ÚLOHY

Obr. A- Pohled na uspořádání úlohy zboku



Obr. B - Pohled na uspořádání úlohy zepředu



Obr. C - Elektromagnetická uchopovací hlavice

7.2 ELEKTRONICKÁ PŘÍLOHA

Elektronická příloha (CD) obsahuje:

- Kompletní projekt hlavního řídícího programu Checkers ve vývojovém prostředí Microsoft Visual Studio 2008.
- Kompletní projekt programu VisionTestProg ve vývojovém prostředí Microsoft Visual Studio 2008.
- Kompletní projekt programu EpsonC3Com ve vývojovém prostředí Microsoft Visual Studio 2008.
- Kompletní projekt s programem do jednotky ve vývojovém prostředí EPSON RC+
- Vektorové grafické podklady pro šachovnici a nálepky na hrací kameny v programu CorelDRAW 12.
- Zdrojový soubor s textem diplomové práce v programu Microsoft Word 2003, přiložený font a převedený PDF dokument.