VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

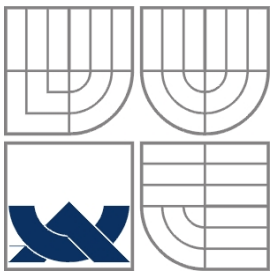
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

MODEL ROBOTICKÉHO RAMENA V PROSTŘEDÍ MICROSOFT ROBOTIC STUDIO

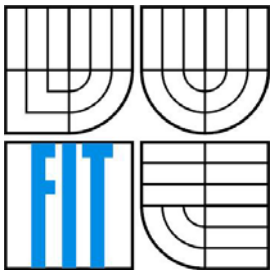
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JOSEF HUBR
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

MODEL ROBOTICKÉHO RAMENA V PROSTŘEDÍ MICROSOFT ROBOTIC STUDIO

ROBOTIC ARM MODEL IN THE MICROSOFT ROBOTIC STUDIO

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE — Bc. JOSEF HUBR
AUTHOR

VEDOUCÍ PRÁCE — Ing. FRANTIŠEK ZBOŘIL, Ph.D.
SUPERVISOR

BRNO 2008

Abstrakt

Cílem práce bylo seznámit se se způsobem modelování robotů ve vývojovém prostředí Microsoft Robotics Studio a zjistit, jaké prvky a vazby jsou nutné pro vytvoření modelu robotického ramena. Za pomoci získaných znalostí byl vytvořen funkční model ramena a byly provedeny pokusy se zvedáním předmětů o různých hmotnostech. Práce je složena z úvodu do robotiky, z popisu částí Microsoft Robotics Studia, z implementace modelu a z provedených testů.

Klíčová slova

Microsoft Robotics Studio, Model robotického ramena, Simulace, Klouby, Chapadlo, Zvedání předmětů

Abstract

The aim of this work was to introduce robot modelling in Microsoft Robotics Studio and to define the essential components and links for creation of robotic arm model. Thanks to gained knowledge has been created working robotic arm model and carried out experiments to lift up objects with various weight. The work consists of introduction to robotics, descriprion of Microsoft Robotis studio features, robotic arm model implementation and experiments to lift up objects.

Keywords

Microsoft Robotics Studio, Robotic arm model, Simulation, Joints, Gripper, Lift up object

Citace

Hubr Josef: Model robotického ramena v prostředí Microsoft Robotic Studio. Brno, 2008, diplomová práce, FIT VUT v Brně.

Model robotického ramena
v prostředí Microsoft Robotic Studio

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením
Ing. Františka Zbořila, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Josef Hubr
21.1.2008

Poděkování

Děkuji Ing. Františku Zbořilovi, Ph.D. za pomoc poskytnutou při tvorbě tohoto projektu.

Obsah

1 Úvod

Tento diplomový projekt se zabývá tematikou robotických ramen a jejich simulací. Úkolem je seznámit se s modelováním robotů v prostředí Microsoft Robotics Studio, následně identifikovat prvky a vazby nezbytné pro vytvoření modelu robotického ramena. Dalším úkolem je implementovat model a ověřit jeho chování na případech, kdy měly budou zvedána břemena s různými tvary a hmotnostmi. Nakonec má být provedena diskuze validity modelu ramena a možnost jeho uplatnění při výuce předmětu Robotika.

V kapitole Teoretická část - Roboti je popsán úvod do problematiky robotů, robotických ramen, základních částí, způsobu programování ramen a řešení úlohy přímé a inverzní kinematiky.

Část Teoretická část - Microsoft Robotics Studio se věnuje popisu hlavních částí tohoto vývojového prostředí.

Další část se nazývá Praktická část - Model ramena a ukazuje, jak byl vytvořen model robotického ramena a věnuje se popisu jeho důležitých částí.

Kapitola Praktická část - Zvedání předmětů popisuje provedené pokusy se zvedáním předmětů a shrnuje dosažené výsledky.

Diskuze validity modelu a uplatnění v předmětu Robotika se zabývá zamyšlením nad platností vytvořeného modelu a nad možnostmi použití v předmětu Robotika.

Na závěr jsou shrnuty poznatky z řešení projektu, dosažené výsledky a popsán další směr vývoje projektu.

2 Teoretická část - Roboti

Roboty obecně se zabývá robotika. Je to obor, který se zabývá studiem a konstrukcí robotů a jim podobných zřízení. Slovo robot pochází z dramatu Karla Čapka R.U.R - Rossum´s Universal Robots (slovo Robot vymyslel jeho bratr Josef Čapek). Robotem je nazýván stroj, který vykonává podobné činnosti jako člověk.

2.1 Způsoby přístupu k robotice

K robotice lze přistupovat několika způsoby, liší se podle pohledu s jakým k robotovi přistupujeme. Teoretická robotika se snaží hledat principy, možnosti a omezení robotů. Uplatňují se zde znalosti z mnoha oblastí, např. matematika, fyzika, biologie a další. Experimentální robotika se pokouší ověřovat principy z teoretické robotiky a na jejich základu staví experimentální roboty, uplatňují se v ní znalosti z kybernetiky (věda o společných vlastnostech sdělování a řízení v živých organismech, společenstvích a ve strojích) a umělé inteligence. Dalším způsobem přístupu k robotice je průmyslová robotika. Ta se snaží navrhovat, stavět a používat průmyslové roboty. Uplatňují se zde obory z elektroniky, strojírenství, automatizace a oboru z oblasti nasazení robotů. Posledním způsobem přístupu je aplikovaná robotika. Snaží se navrhovat inteligentní roboty pro průmysl (např. kontrola kvality) a jiné oblasti (např. pro vojenství nebo vesmírný výzkum, kde se jedná o různé mobilní roboty).

2.2 Manipulace

Je to jeden z nejčastějších úkolů robota. Manipulace je umožněna pomocí chapadel. Existuje několik technologií, jak uchopovat předměty: mechanická, vakuová a elektromagnetická. Mechanická chapadla jsou nejpoužívanější technologií, jsou složeny z dvou a více úchopných částí, které se mohou k sobě vzájemně přibližovat a vzdalovat a díky tomu lze předměty umístěné mezi nimi uchopovat a pouštět. Je mnoho způsobů konstrukce úchopných částí, např. vnější chapadlo, vnitřní chapadlo, čelisťové, atd. Dalším typem chapadla je chapadlo vakuové (podtlakové), to pracuje na principu přiblížení chápací části k objektu a odčerpání vzduchu, takže objekt je přichycen díky podtlaku. Tyto chapadla se využívají při přemisťování velkých objektů (např. karoserie a skla). Elektromagnetické chapadlo umožňuje uchopovat feromagnetické předměty pomocí vytvořeného magnetického pole, vypnutím tohoto pole je předmět upuštěn.

2.3 Způsoby pohybu robota

Roboti, kteří se mohou pohybovat, se nazývají mobilní roboti. Pohyb (též. lokomoce) robota je možné provádět několika způsoby v závislosti na druhu prostředí, ve kterém se robot pohybuje. Pohyb lze zajistit pomocí kol, pásů, nohou, křídel (let) a dalších technologií. S pohybem robota souvisí další problematika, jako je určení polohy, ujetá vzdálenost atd.

Kola a pásy jsou jednoduchým a nejčastějším prostředkem umožňujícím pohyb robota. U konstrukce s koly je často použito tří nebo čtyř kol a u konstrukce s pásy se nejčastěji používají pásy dva. Někdy je použita i kombinace kol a pásů.

Při pohybu pomocí nohou se musí řešit nové problémy, které se u kol a pásů nevyskytují. Hlavní problém je rovnováha (u dvounohých robotů) jak při stání, tak při pohybu (rovnováha se využívá i k rozpohybování robota). U robota s větším množstvím nohou tento problém odpadá, protože nohy slouží spíše jako opěrné body a pohyb je uskutečňován jiným způsobem (např. roboti na způsob stonožek).

Další způsoby pohybu využívají dalších technologií. Létající roboti slouží hlavně pro armádní účely, podmořští roboti slouží pro výzkumné a vyprošťovací práce atd.

2.4 Motory

Motory umožňují robotům pohyb v prostředí a nastavování částí robota do zadaných pozic. Ve většině motorů používaných v robotice se jedná o stejnosměrné elektromotory. Využívá se podmnožina těchto motorů, a to krokové rotační a krokové lineární. Krokový motor má tu vlastnost, že ho lze postupně nastavovat (krokovat). Počet kroků motoru, do kterých ho lze vystavit, závisí na počtu pólových párů v motoru. Rotační krokové motory se používají u pohonu kol a rotačních kloubů. Lineární krokové motory jsou použity u lineárních (translačních) kloubů.

2.5 Umělá inteligence

V robotice se využívá mnoho znalostí z umělé inteligence. Nejčastěji je umělá inteligence použita u mobilních robotů při hledání cesty. U robotů je snaha o stereovidění, kde se umělá inteligence stará o rozpoznávání objektů v prostředí a odhad vzdálenosti. Zpracování obrazu z kamery umělou inteligencí se využívá i při navigování robotických ramen v proměnném prostředí.

2.6 Nasazení robotů

Roboti jsou nasazováni v mnoha prostředích a oblastech. Umožňují zvýšit výkon, produktivitu práce a zkrácení časů potřebných na výrobu, takže v mnoha případech nahrazují lidskou činnost při výrobním procesu. Jedná se o různé manipulační, kontrolní a obráběcí činnosti (např. sváření, kontrola kvality, atd.). Dále jsou roboti nasazování v nebezpečných oblastech, jako je pohyb v radioaktivním prostředí nebo odstraňování různých výbušnin. Nasazování robotů probíhá i v dalších oblastech např. lékařství, vojenství, průzkum vesmíru nebo hračky.

2.7 Robotické zákony

Byly vymyšleny tři robotické zákony, kterými by se měli řídit inteligentní roboti. Jejich autorem je spisovatel sci-fi literatury Isaac Asimov. Zákony jsou následující (seřazené podle priority):

1. Robot nesmí ublížit člověku nebo svou nečinností dopustit, aby mu bylo ublíženo.
2. Robot musí uposlechnout člověka, kromě případu, kdy je to v rozporu s prvním zákonem.
3. Robot musí chránit sám sebe před zničením, kromě případů, kdy je tato ochrana v rozporu s prvním nebo druhým zákonem.

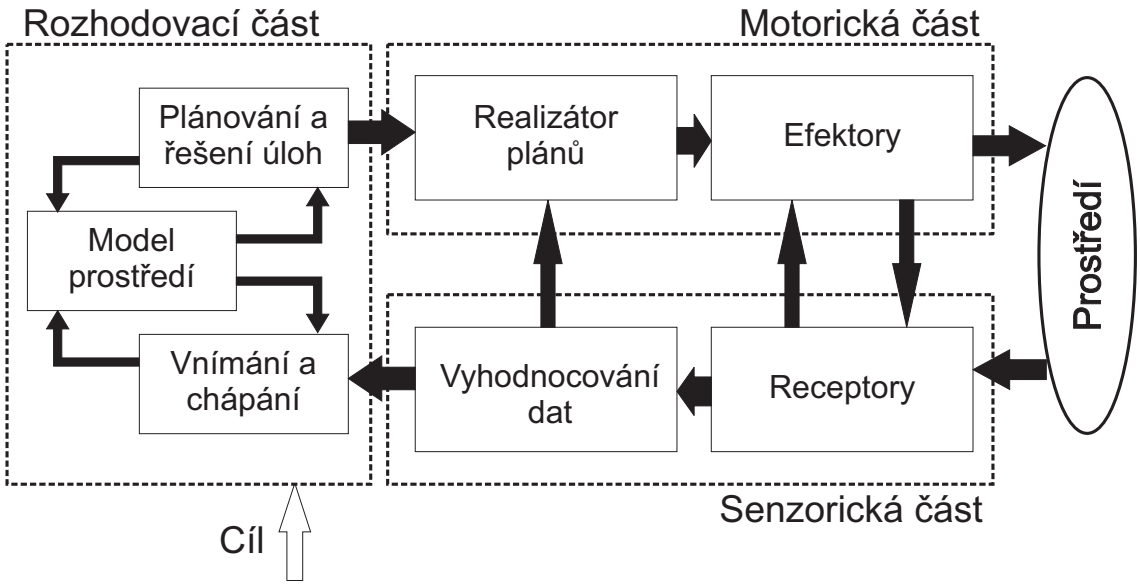
Časem byly vymyšleny ještě další tři zákony: Robot nesmí ublížit lidstvu nebo svou nečinností dopustit, aby mu bylo ublíženo (je braný jako nultý zákon); Robot se musí vždy prokazovat jako robot; Robot musí vědět to, že je robot.

2.8 Části robotického ramena

Obecného robota lze rozdělit do tří částí, podle toho, jakou činnost vykonávají. Jedná se o část motorickou, senzorickou a rozhodovací. Motorická část obsahuje efektory (slouží k fyzickému ovlivňování prostředí a pohybu robota v prostoru) a realizátor plánů (stará se o řízení efektorů). Senzorická část obsahuje receptory (slouží k snímání fyzikálních signálů z prostředí) a vyhodnocování dat z receptorů (zpracovává a vybírá data důležitá pro chování robota). Rozhodovací část zajišťuje vytváření modelu prostředí, jeho hlubší analýzu a rozhodování o vykonání akcí. Obsahuje vnímání a chápání (podle dat získaných od senzorické části), plánování a řešení úloh (zadávaných k vykonání motorické části) a model prostředí.

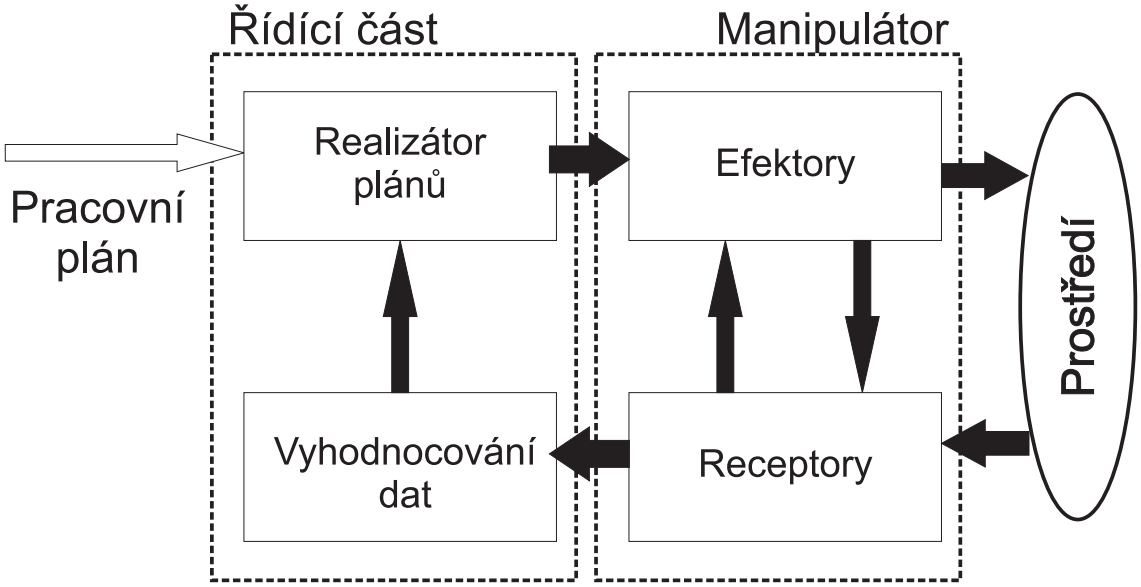
Části robota mezi sebou obsahují smyčky. Mezi efektory a senzory je to smyčka reflexní, která umožňuje rychle reagovat na určité události. Může se jednat o případ, kdy se robot dotkne předmětu. Receptory oznámí, že došlo ke kontaktu a efektory zastaví pohyb robota. Další smyčka je operační, propojuje vyhodnocení dat z receptorů a realizátor plánů. Stará se o vykonání naplánované úlohy,

např. vystavení robotického ramena do určité polohy. Poslední smyčkou je připojena rozhodovací část, která zajišťuje inteligenci robota.



Obrázek 1 - Části obecného robota

U průmyslových robotů bývá rozhodovací část nahrazena programem toho, co má robot dělat. Je také nazývaný plán práce. Ostatní části průmyslového robota obsahují stejné prvky jako u obecného robota, jen jsou trochu jinak uspořádány. Efektory a receptory tvoří manipulátor, který zasahuje do prostředí a získává z něj informace. Vyhodnocování dat z receptorů a realizátor plánů tvoří řídící část, která zajišťuje vykonávání plánu práce.



Obrázek 2 - Části průmyslového robota

Důležitou informací v robotice je počet stupňů volnosti. Poloha objektu v prostoru je dána třemi hodnotami (v osách X, Y a Z) a orientace objektu je dána natočením ve třech úhlech (α, β, γ). Pokud je možno objekt nastavovat v těchto šesti parametrech, říkáme o něm, že je volné a že má šest stupňů volnosti. Robotické rameno, které by mělo nastavovat tento objekt, by muselo mít také šest stupňů volnosti, čehož dosáhne minimálně šesti klouby.

O nastavení robotických ramen do zadaných pozic se starají klouby (osy). Existují dva typy, jedná se o rotační a translační. Pokud má rameno méně než šest kloubů, snižuje to jeho manipulační vlastnosti. Na druhou stranu, šest a více kloubů nemusí zajišťovat šest stupňů volnosti. Např. rameno má sedm kloubů, ale umožňují pohyb jen v jedné rovině (např. XY).

2.9 Základní koncepce robotických ramen

Robotické rameno bývá obvykle tvořeno třemi klouby a na konci ramena je připevněno zápěstí, které má také tři klouby, aby bylo dosaženo šesti stupňů volnosti. Základní koncepce ramen jsou kartézská, cylindrická a sférická. U kartézské koncepce se rameno pohybuje ve směru os. U cylindrické se rameno pohybuje ve směru osy Y, může se kolem ní otáčet a může se vysouvat. Sférická koncepce je založena na rameni, které se může otáčet kolem osy Y, v bodě uchycení se může vertikálně naklánět a je mu umožněno vysouvat se. Mimo těchto základních koncepcí existují i jiné způsoby tvorby robotických ramen.

2.10 Programování robotických ramen

Plán práce bývá většinou naprogramován jako sekvence drah, které má rameno vykonat. Dráha je uložena ve formě kloubových souřadnic (každý kloub má svoji hodnotu polohy nebo natočení v závislosti na jeho typu) a času. Vystavení kloubů do požadovaných pozic je úkolem přímé kinematiky. Existují tři způsoby jak vytvořit plán práce. Je to přímé programování, nepřímé programování a přímé plánování.

2.10.1 Přímé programování

V tomto případu zde řeší inverzní kinematiku přímo programátor. Přímé programování lze provádět dvěma způsoby. Prvním způsobem je, že je rameno vedeno přímo a ukládají se všechny pohyby (do tabulky), které byly vykonány. Tento způsob má jedu zásadní nevýhodu, veškeré chyby a nedostatky provedené programátorem jsou uloženy do plánu práce a bude je provádět

i naprogramované rameno. Druhým způsobem je programování ramena pomocí tlačítek na ovládacím panelu, které umožní přesnější zadávání pozice. V tomto případu může být tabulka s programem daleko menší, protože programátor si vybere pouze pozice, které jsou důležité a nechá je s patřičným časem uložit.

2.10.2 Nepřímé programování

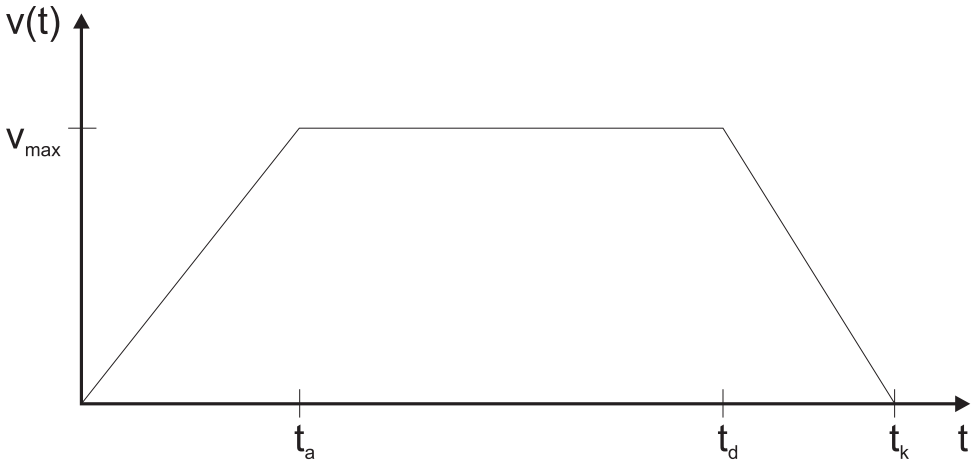
V případu nepřímého programování je pohyb ramena zadáván ve formě křivek (např. podle výkresu). Inverzní kinematika je vypočítána před započetím vykonávání plánu práce.

2.10.3 Přímé plánování

Způsob programování je téměř shodný s nepřímým programováním. Rozdíl je pouze v tom, že se inverzní kinematika vypočítává až za běhu vykonávání plánu práce. Tento způsob se používá v případech, kdy rameno provádí své pohyby na základě informací ze senzorů v proměnném prostředí. Např. chytání míčku za letu.

2.11 Akcelerace/Deakcelerace

Pokaždé, kdy je potřeba přemístit robotické rameno do nové pozice, je nutné, aby pohyby nebyly trhané a nevznikala zapružení. Proto se musí jednotlivé části plynule rozpohybovat. Celý pohyb se skládá ze tří částí: akcelerace, konstantního pohybu a deakcelerace. Jednotlivé části pohybu je zapotřebí vždy vypočítat podle konkrétního umístění ramena. Každý kloub má známou hodnotu polohy, akcelerace, deakcelerace a maximální rychlost pohybu.



Obrázek 3 - Graf rychlosti pohybu kloubu

V grafu je znázorněna rychlost pohybu kloubu podle fáze v jaké se nachází, t_a - místo konce akcelerace, t_d - místo začátku deakcelerace, t_k - konec pohybu, v_{max} - maximální rychlost.

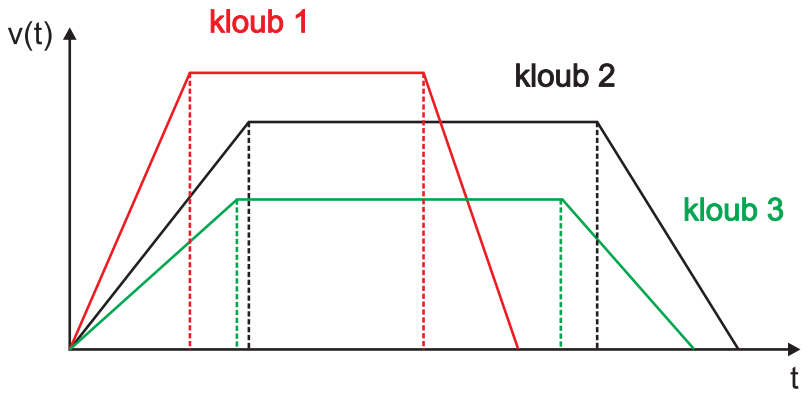
Pohyb začíná akcelerací, při které se rameno dává plynule do pohybu (rychlost závisí na hodnotě akcelerace konkrétních kloubů) dokud nedosáhne požadované nebo maximální rychlosti (liší se v závislosti na způsobu řízení). Poté následuje fáze pohybu, v níž má rameno konstantní rychlost. Celý pohyb je ukončen deakcelerační částí, u které je rameno zpomaleno z konstantní rychlosti do klidu.

2.11.1 Způsoby řízení pohybu

Řízením je myšleno umisťování koncové části ramena do zadané pozice a orientace. U řízení robotického ramena způsobem Point To Point (PTP - např. přímé programování tlačítky) existuje asynchronní a synchronní způsob vystavení do požadované pozice. U řízení způsobem Continuos Path (CP - např. nepřímé programování a přímé plánování) se jedná vždy o plně synchronní pohyb.

2.11.2 Asynchronní řízení

Při tomto způsobu řízení se každá část ramena snaží dostat co nejrychleji do požadované pozice bez ohledu na ostatní. Takže nastávají situace, kdy jsou některé části ramena vystaveny do své pozice dříve než ostatní.

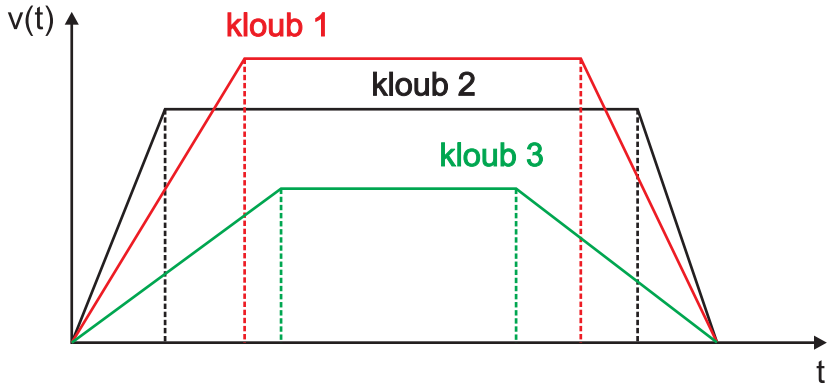


Obrázek 4 - Graf rychlostí kloubů u asynchronního řízení

2.11.3 Synchronní řízení

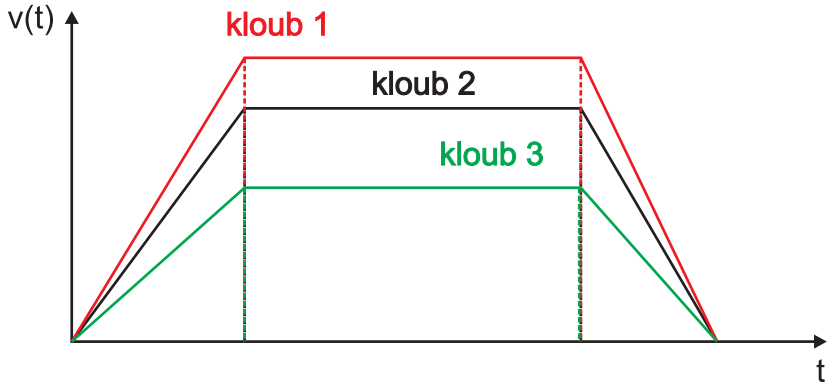
U tohoto způsobu řízení je pohyb jednotlivých částí nastaven tak, že začíná a končí u všech kloubů ve stejný čas. Takže se musí některé části pohybovat rychlostí menší než maximální. Tudíž je toto

řízení náročnější na výpočet, který probíhá tak, že se zjistí kloub potřebující nejvíce času na přemístění do cílové pozice (nazývá se řídící kloub) a podle něj se vypočítají rychlosti ostatních kloubů.



Obrázek 5 - Graf rychlosti kloubů u synchronního řízení

Další způsob synchronního řízení je tzv. plně synchronní řízení. Je podobné jako synchronní, jen se zde navíc shodují všechny časy konce akcelerace a začátku deakcelerace.



Obrázek 6 - Graf rychlosti kloubů u plně synchronního řízení

2.12 Kinematika robotických ramen

Kinematika se zabývá pohybem těles (jako je dráha, rychlost nebo zrychlení) bez ohledu na působící síly. Robotické rameno tvoří kinematický řetězec. Jedná se o řetězec složený z kinematických dvojic (dvě spojené části ramena pohybující se vzhledem k sobě, spoj se nazývá kloub), který je z jedné strany pevně upevněn v prostředí.

Kinematiku robotických ramen můžeme rozdělit na přímou a nepřímou podle toho, jaký úkol máme s kinematickým řetězcem řešit. V obou případech platí, že pozice a orientace koncového bodu ramena závisí pouze na pozici jednotlivých kloubů.

2.12.1 Přímá kinematika

Přímá kinematika se zabývá otázkou, jak zjistit z kloubových proměnných pozici konce ramena v prostoru. Tuto úlohu lze řešit analyticky.

2.12.1.1 Translace souřadných systémů

Jednotlivé části robotického ramena používají lokální souřadné systémy a pracovní prostor, ve kterém rameno operuje, používá globální souřadný systém. Pokud potřebujeme zjistit polohu pozice v souřadnicích globálního souřadného systému, musí se provést transformace ze souřadnic lokálních do globálních. Pozice bodu P v lokálním souřadném systému je udána vektorem $p_l = (x_l, y_l, z_l)$, který můžeme zapsat ve tvaru: $p_l = px_l \cdot i_l + py_l \cdot i_l + pz_l \cdot i_l$.

$$\begin{bmatrix} px_g \\ py_g \\ pz_g \end{bmatrix} = \begin{bmatrix} i_g i_l & i_g j_l & i_g k_l \\ j_g i_l & j_g j_l & j_g k_l \\ k_g i_l & k_g j_l & k_g k_l \end{bmatrix} \cdot \begin{bmatrix} px_l \\ py_l \\ pz_l \end{bmatrix}$$

Matice pro přepočet souřadnic z lokálního (l) do globálního souřadného systému (g). Tento výpočet lze zapsat i jako $(p)_g = T_g^l \cdot (p)_l$

Matice říká, že bod P přepočteme ze souřadnic lokálního souřadného systému l do souřadnic globálního souřadného systému g tak, že vynásobíme transformační matici T souřadnicemi bodu P.

2.12.1.2 Rotace souřadných systémů

Existují tři matice rotace souřadných systémů v závislosti na tom, kolem které osy rotace probíhá. U rotace kolem osy x se použije matice $R_{x,\alpha}$, u rotace kolem osy y matice $R_{y,\beta}$ a u rotace kolem osy z matice $R_{z,\gamma}$.

$$R_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \quad R_{y,\beta} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \quad R_{z,\gamma} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

V jednom výpočtu je možné použít i více rotací, např. $(p)_0 = R_{z,\gamma} \cdot R_{x,\alpha} \cdot (p)_1$, pořadí jednotlivých transformací musí být ve stejném pořadí jako je pořadí požadovaných operací.

2.12.1.3 Homogenní transformační matice

Jedná se o transformační matici, která v sobě slučuje rotaci a posunutí souřadných systémů.

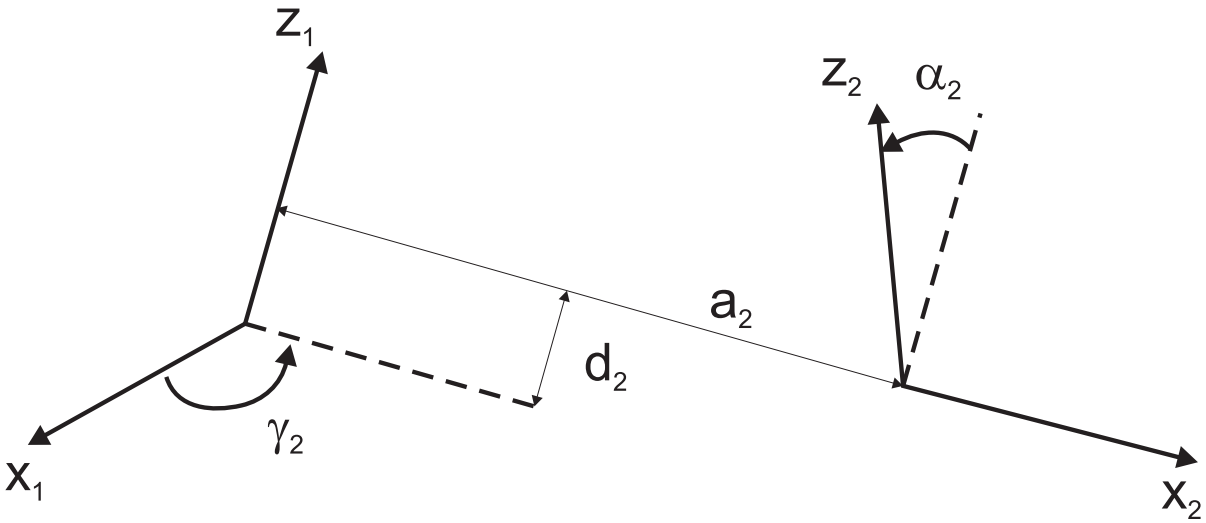
$$\begin{bmatrix} px_g \\ py_g \\ pz_g \\ 1 \end{bmatrix} = \begin{bmatrix} & R & & o_x \\ & & & o_y \\ & & & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} px_l \\ py_l \\ pz_l \\ 1 \end{bmatrix}$$

Homogenní transformační matice pro výpočet z lokálního (l) do globálního souřadného systému. Lze použít zápis $(p)_g = H_g^l \cdot (p)_l$

Homogenní transformační matice je složena z rotační podmatice R (viz. rotace souřadných systémů) a vektoru o, který udává posunutí počátku systému l v souřadnicích systému g.

2.12.1.4 Denavit-Hartenbergův princip

Jedná se o princip umisťování souřadných systémů, díky němuž lze automaticky generovat vzájemné transformační matice a vypočítávat z nich pozici bodu.



Obrázek 7 - Ukázka Denavit-Hartenbergova principu

Na obrázku je ukázáno, jak funguje Denavit-Hartenbergův princip. Jsou zde dva souřadné systémy (S_1 a S_2), které patří dvojici částí robotického ramena. Úkolem je provést transformace systému S_1 tak, aby došlo ke sjednocení se systémem S_2. Toho dosáhneme následujícími operacemi.

Provede se rotace osy x_1 kolem osy z_1 o úhel γ_2 tak, že osy x_1 a x_2 budou rovnoběžné. Další operací je posunutí osy x_1 o vzdálenost d_2 tak, že bude totožná s osou x_2. Když nyní posuneme počátek systému S_1 o a_2, ztotožníme počátky obou systémů. Nyní již zbývá jen provést rotaci kolem osy x o úhel α_2 a systémy jsou totožné.

Při dodržení pravidel rozmisťování souřadných systémů může být předchozí popis zobecněn a platí, že souřadné systémy lze sjednotit čtyřmi pohyby: rotací, translací, translací a rotací (za dodržení předchozího postupu). Dostáváme tedy čtyři matice:

$$A_{z_1,\gamma_2} = \begin{bmatrix} \cos\gamma_2 & -\sin\gamma_2 & 0 & 0 \\ \sin\gamma_2 & \cos\gamma_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1. Natočení osy x_1 kolem osy z_1 o úhel γ_2

$$A_{z_1,d_2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Posunutí osy x_1 ve směru osy z_1 o vzdálenost d_2

$$A_{x_2,a_2} = \begin{bmatrix} 1 & 0 & 0 & a_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Posunutí počátku S_1 ve směru osy x_2 o vzdálenost a_2

$$A_{x_2,\alpha_2} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_2 & -\sin\alpha_2 & 0 \\ 0 & \sin\alpha_2 & \cos\alpha_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Natočení osy z_1 kolem x_2 o úhel α_2

Spojením těchto transformačních matic vznikne nová matice Denavit-Hartenbergova principu v obecném tvaru. Spojení se provede vynásobením matic v pořadí prováděných transformací (rotace, translace, translace a nakonec rotace): $A_1 = A_{z_1,\gamma_2} \cdot A_{z_1,d_2} \cdot A_{x_2,a_2} \cdot A_{x_2,\alpha_2}$

$$A_1 = \begin{bmatrix} \cos\gamma_2 & -\sin\gamma_2\cos\alpha_2 & \sin\gamma_2\sin\alpha_2 & a_2\cos\gamma_2 \\ \sin\gamma_2 & \cos\gamma_2\cos\alpha_2 & -\cos\gamma_2\sin\alpha_2 & a_2\sin\gamma_2 \\ 0 & \sin\alpha_2 & \cos\alpha_2 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Tvar transformační matice po vynásobení

Matici lze použít u všech typů kloubů (jak rotační, tak translační), protože má ve všech případech stejný tvar. Pro zobecnění ještě upravíme indexy, (1) odpovídá (i-1) a (2) odpovídá (i).

Parametry této matice tedy zcela popisují vztahy mezi sousedními souřadnými systémy a jmenují se Denavit-Hartenbergovy parametry ($\gamma_i, d_i, a_i, \alpha_i$). Popis jednotlivých parametrů:

γ_i - úhel mezi osami x_{i-1} a x_i při otáčení kolem osy z_{i-1}

d_i - nejkratší vzdálenost mezi osami x_{i-1} a x_i, kladné ve směru z_{i-1}

a_i - nejkratší vzdálenost mezi osami z_{i-1} a z_i, kladné ve směru x_i

α_i - úhel mezi osami z_{i-1} a z_i při otáčení kolem osy x_i

Pravidla rozmisťování souřadných systémů:

- části robotického ramena se číslují následovně, základna je očíslována číslem 0, navazující části jsou postupně číslovány od čísla 1 do n
- klouby se číslují od základny, i-tý kloub spojuje i-1 a i-tou část
- osa z_{i-1} je osou otáčení i-tého kloubu, kladný směr je nejlépe do kladného kvadrantu
- osa x_i je kolmá na osy z_{i-1} a z_i, mohou nastat případy:
    - osa z_{i-1} a z_i je totožná, pak osa x_i může být umístěna kolmo na obě totožné osy v libovolném místě a směru, nejvhodněji tak, že bude rovnoběžná s předchozí osou (x_{i-1}), zjednoduší se tak transformační matice
    - osy z_{i-1} a z_i jsou mimoběžné, pak osa x_i musí ležet ve společné normále z-vých os a její kladný směr je dán směrem od osy s nižším indexem k ose s vyšším indexem
    - osy z_{i-1} a z_i jsou různoběžné, pak osa x_i vede kolmo z průsečíků os z, kladný směr určuje to, aby při rotaci kolem osy x_i přešla osa s nižším indexem (i-1) na osu z s vyšším indexem (i) v kladném směru otáčení (proti směru otáčení hodinových ručiček)
- osa z_n posledního souřadného systému by měla vést z konce koncové části ramena rovnoběžně s osou z předchozího souřadného systému
- osa x_n posledního souřadného systému je vedena tak, aby vždy protnula osu z předchozího souřadného systému, kladný směr je do pracovního prostoru

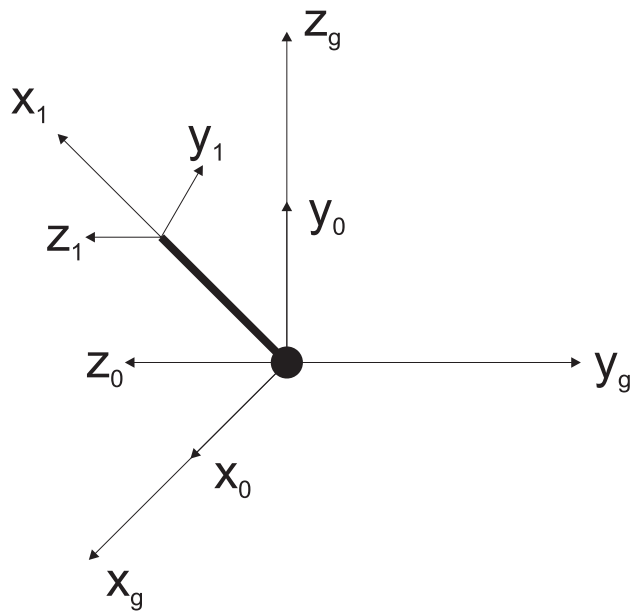
Celá transformační matice robotického ramena vznikne vynásobením jednotlivých souřadných systémů částí ramena, $T_g^n(q_1, q_1, ..., q_n) = A_g^0 \cdot A_0^1(q_1) \cdot A_1^2(q_2) \cdot ... \cdot A_{n-1}^n(q_n)$, kde q_i jsou kloubové proměnné.

Úloha přímé kinematiky řešená pomocí Denavit-Hartenbergova principu se vyřeší rovnicí

$$\begin{bmatrix} x_g \\ y_g \\ z_g \\ 1 \end{bmatrix} = T_g^n(q_1, q_2, ..., q_n) \cdot \begin{bmatrix} x_n \\ y_n \\ z_n \\ 1 \end{bmatrix}$$

podle předem vytvořených transformačních matic jednotlivých částí robotického ramena.

2.12.1.5 Příklad D-H principu u robotického ramena s jedním rotačním kloubem



Obrázek 8 - Souřadné systémy robotického ramena

Jedná se o robotické rameno, které je umístěno ve středu globálního souřadného systému. Má jediný rotační kloub, který má osu otáčení totožnou s globální osou y a umožňuje ramenem otáčet v globální rovině XZ Vytvoříme následující tabulku, hodnoty jsou doplněny podle obrázku.

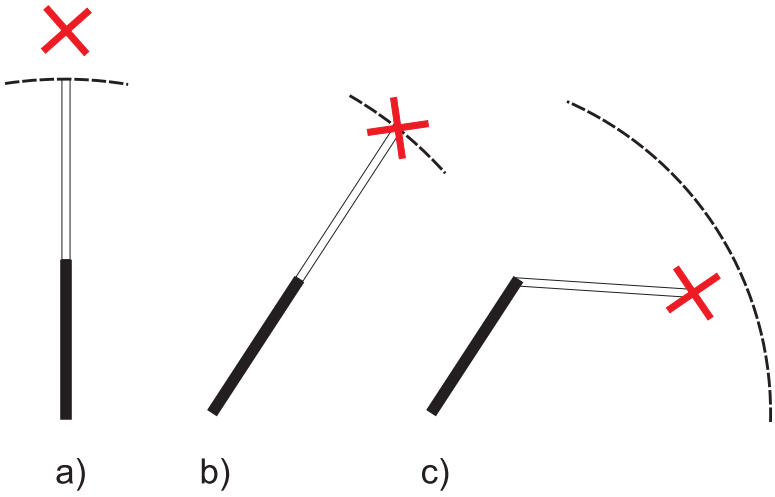
i	γ_i	d_i	a_i	α_i
0	0	0	0	$\pi/2$
1	$q1$	0	l_1	0

Hodnoty úhlů tabulky jsou udávány v radiánech. Délku ramena určuje l_1 a jediná proměnná v této tabulce je $q1$, která určuje úhel natočení ramena.

2.12.2 Inverzní kinematika

Inverzní kinematika se zabývá otázkou, jak nastavit kloubové proměnné, když známe pozici a orientaci koncového členu kinematického řetězce. Tato úloha lze v konkrétních případech řešit vektorově, u ostatních případů je potřeba numerické řešení. Proto je řešení inverzní kinematiky náročnější než v případu kinematiky přímé.

Pracovní prostor je označení pro oblast, ve které může robotické rameno operovat. Inverzní kinematika musí umět reagovat na všechna zadání v pracovním prostoru. Výsledkem řešení inverzní kinematiky může být žádné řešení, jedno řešení nebo více řešení. Řešení nedostaneme (viz obr. 9a) v případě, kdy požadované umístění pozice koncového členu ramena leží mimo oblast pracovního prostoru. Jedno řešení (viz. obr. 9b) je dosaženo v případech, kdy požadované umístění pozice leží na hranici pracovního prostoru. Inverzní kinematika může mít více řešení (viz. obr. 9c), pokud požadovaná pozice leží uvnitř pracovního prostoru.



Obrázek 9 - Možné výsledky při řešení inverzní kinematiky

2.12.2.1 Numerická řešení

Řešení inverzní kinematiky tímto způsobem existuje několik druhů, jsou to: numerické řešení soustavy transcendentních rovnic, aproximační řešení a heuristika.

Řešení soustavy transcendentních rovnic je způsob, u kterého se řeší tato soustava rovnic (např. rovnice x = cos(x) je transcendentní). K sestavení rovnic je využito Denavit-Hartenbergovy transformační matice.

Aproximační řešení se snaží úlohu inverzní kinematiky řešit pomocí soustavy rovnic, v nichž se výpočtu účastní požadovaná pozice, současná pozice a neznámé tvoří diference kloubových proměnných. Po vyřešení soustavy rovnic je počítána nová, ale za současnou pozici je dosazen spočítaný výsledek. Tento postup se provádí, dokud není dosaženo požadované pozice.

Heuristika je metoda, při které se hledá minimální chyba pozice koncové části ramena od požadované. Výpočet se provádí cyklickým vystavováním kloubů a zjišťováním, jaké by bylo dosaženo chyby při aktuální zkoumané pozici.

2.12.2.2 Vektorové řešení

Tento způsob řešení inverzní kinematiky využívá trigonometrických vztahů mezi částmi ramena. Při výpočtu jsou použity vektory a matice pro doplnění trigonometrických vztahů. S výhodou se dá použít Denavit-Hartenbergova principu pro sestavení transformačních matic.

Matice se vytvoří podle Denavit-Hartenbergových parametrů konkrétního systému. Takže lze vypočítat pozici a orientaci koncového souřadného systému. Dále je známá požadovaná pozice a orientace koncové části ramena.

Nejdříve se určí pozice předposledního souřadného systému ramena, ze znalosti požadované polohy a rozměrů poslední části ramena. Nyní jsou známé již všechny potřebné údaje pro vypočítání hodnot kloubových proměnných, které se vypočítají pomocí trigonometrie.

3 Teoretická část - Microsoft Robotics Studio

Jedná se o volně stažitelné (pro nekomerční využití) vývojové prostředí sloužící pro tvorbu robotických aplikací. Hlavní součásti Microsoft Robotics Studia (MSRS) jsou Konkurenční a koordinační funkce, Distribuovaný systém služeb a Grafické simulační prostředí. MSRS dále obsahuje Visual Programming Language (VPL).

3.1 Konkurenční a koordinační funkce (CCR)

Konkurenční a koordinační funkce (CCR - Concurrency and Coordination Runtime) slouží k řízení asynchronních operací, domluvě na souběžnosti (paralelnosti), umožňuje využít paralelní hardware a dohody při částečných chybách. Jeho základní části jsou tvořeny:

3.1.1 Porty

Port (funguje jako FIFO fronta) je jedna ze základních komponent sloužících ke komunikaci mezi komponentami. Porty se dělí podle způsobu vyjímání položek z fronty. Jsou buď pasivní, nebo aktivní. V případu pasivního vyjímání, se portu dotazujeme, jestli obsahuje položku a tu následně začneme zpracovávat. Pokud se jedná o aktivní vyjímání, je u portu zaregistrován arbiter, který položku začne rovnou zpracovávat. Příklad pasivního a aktivního portu přijímajícího int:

Pasivní

```
Port<int> iPort = new Port<int>();
// poslání hodnoty na port
iPort.post(10);
int hodnota;
// vyjmuti hodnoty a další zpracování
if(iPort.Test(out hodnota))
{
   …
}
```

Aktivní

```
Port<int> iPort = new Port<int>();
iPort.post(10);
Dispatcher dis = new Dispatcher
(0,"sd");
DispatcherQueue que = new
   DicpatcherQueue("sq",dis);
Arbiter.Activate(
   que,
   Arbiter.Receive(
// proběhne jednou a odregistruje se
      false,
// port, ke kterému se registruje
      iPort,
// user delegate
      delegate(int hodnota) { ... }
));
```

3.1.2 Arbitery

Protože CCR umožňuje a podporuje souběžnost, umožňuje také koordinaci. To spočívá v odesílání požadavků a přijímání odpovědí, o což se starají arbitery. Je jich více druhů, některé umí provádět pouze jednu operaci (Single Item Receiver), další provádí pouze jednu z několika operací (Choice Arbiter), jsou i arbitry, které provádí operace až po přijetí zpráv z více portů (Joined Receive) atd. Když se podíváme na příklad aktivního portu, vidíme, že jsou zde použity dva arbitery. Arbiter.Activate je zaregistrovaný k frontě úloh a sleduje, jestli nebyla nějaká přidána, reaguje pouze na aktivitu a o další část se stará Arbiter.Receive. Ten si zjistí, jestli nebyla přidána položka do portu, ke kterému je registrován (iPort). Pokud byla položka nalezena, začne ji zpracovávat podle příkazů zadaných v user delegate, a protože je zaregistrován pouze pro jedno použití, po vykonání kódu se od portu odregistruje (v případě, že by bylo nastaveno true, zůstal by k portu zaregistrován pořád a zpracovával by další příchozí zprávy). Zpracování v user delegate probíhá vždy paralelně.

3.1.3 Plánováním

Zajišťuje spouštění operací na základě přijatých zpráv. Hlavní částí plánování jsou ITask, DispatcherQueue, Dispatcher. Do plánování jsou zařazeny jen úlohy, které implementují rozhraní ITask. Jsou to Task, IterativeTask a všechny arbitery. DispatcherQueue je fronta (typu FIFO), do které se řadí zadané úlohy. Dispatcher spolupracuje s operačním systémem a načítá úlohy z instance/í DispatcherQueue.

3.1.4 Iterátory

Jedná se o konstrukci jazyka C# 2.0, která umožňuje programovat asynchronní chování. Spouštění většinou probíhá na základě delegace v arbiteru, např.:

```
Arbiter.Activate(_taskQueue, Arbiter.FromIteratorHandler(MyIterator))
```

Funkce iterátoru bude nejlépe popsatelná na následujícím příkladu pseudokódu:

```
IEnumerator<ITask> MyIterator()
{
	if(rameno.Stisknuto) yield return rameno.Pust();
	if(!rameno.Vystaveno(_souradnice))
		yield return rameno.Vystav(_souradnice);
	rameno.Stiskni();
}
```

IEnumerator vrací instance objektů, které implementují rozhraní ITask. Takže lze jednotlivé příkazy vykonávat postupně a asynchronně. Když si vezmeme předchozí příklad, tak se nejdříve zjistí, jestli je chapadlo ramena stisknuto. Pokud tomu tak není, je přikázáno, aby se otevřelo. Jakmile je příkaz vykonán, pokračuje se dále v příkazech funkce a zjistí se, jestli je rameno vystaveno na potřebné pozici. Pokud tomu tak není, je zadán příkaz pro vystavení na pozici a na té je chapadlo stisknuto.

3.1.5 Způsob přístupu k CCR

CCR je vytvořeno jako knihovna přístupná z jazyků pracujících s .NET 2.0.

3.2 Decentralizovaný systém služeb (DSS)

Decentralizovaný systém služeb (DSS - Decentralized System Service) zajišťuje prostředí pro komunikaci mezi službami. Služby mohou komunikovat mezi sebou na tomtéž počítači nebo po síti. Každá služba má své identifikátory, stav, partnery, port, handlery a hlášení událostí.

(Service Identifier)
Je to dynamicky přidělované URI (Uniform Resource Identifier), které identifikuje tzv. DSS Node (kontext odkud je služba spouštěna) a je tedy možno ke službě přistupovat přes webový prohlížeč.
Tvar identifikátoru: http://[počítač]:[port]/[název služby]

(Contract Identifier)
Jedná se o zhuštěný popis služby, který slouží ke generování DLL knihoven odkazů na konkrétní službu. DLL knihovny jsou poté používány při komunikaci mezi službami. I v tomto případě je identifikátor ve formě URI. Název se automaticky vygeneruje při založení projektu služby.
Tvar identifikátoru: http://schemas.tempuri.org/[rok]/[mesic]/[jmeno].html

(Service State)
Každá služba má svůj stav, který je definovaný v každém okamžiku. Může se např. jednat o rychlost otáček motoru, úhel natočení ramena, položky ve frontě atd. Každá informace uložená ve stavu služby může být čtena a upravována.

(Service Partners)

Partneři jsou další služby, které služba potřebuje mít spuštěné ke svému správnému provozu. Jde například o službu prostředí s robotem, jehož partnerem je ovladač robota (bez připojení služby s ovladačem nepůjde robota ovládat).

(Main Port)

Jde o port, na kterém služba přijímá zprávy od jiných služeb. Port přijímá pouze operace, pro které je definován. Jedná se o operace definované DSSP (LOOKUP, DROP, GET, REPLACE) a HTML (GET, POST). DSSP je jednoduchý protokol založený na SOAP (protokol pro výměnu XML zpráv).

(Service Handlers)

Pro každou operaci definovanou pro Main Port musí být zaregistrován handler, který zachytává přicházející zprávy. Handlery také umožňují zprávy odesílat. Existují dva způsoby, jakými mohou být zprávy odeslány, a to buď vyžádaně, nebo nevyžádaně. Vyžádané zprávy jsou zprávy ve formě hlášení události (Event Notification), odesílají se přes subscriber. Nevyžádané zprávy jsou zprávy s požadavkem odesílané jiné službě. V obou případech jsou zprávy odesílány přes service forwarder (lokální reprezentace Main Portu jiné služby).

(Event Notification)

Služby umožňují subscribing (přihlášení se k zasílání tohoto typu zprávy). Event notification je zpráva informující o výsledku změn stavu dané služby. Přesněji to je množina operací na Main portu služby, které byly provedeny a způsobily změnu stavu. Po přihlášení se k zasílání subscription, je pro každou službu vytvořen zvlášť port pro přijímání zpráv.

3.2.1 Způsoby spuštění služby

Službu Microsoft Robotic Studia lze spustit několika způsoby. Spuštění z vývojového prostředí probíhá klasickým spuštěním překladače (např. u Visual Studia je to klávesou F5). Další možností je spuštění přes webový prohlížeč (možné jen za běhu DSS).

3.2.2 Přístup ke službám

Jakmile je služba vytvořena a prostředí spuštěno, je automaticky přístupná přes webové rozhraní. Lze zobrazovat stav služeb (ve formátu XML).

3.3 Grafické simulační prostředí

Microsoft Robotics Studio také obsahuje grafické prostředí pro provádění simulací robotů, které nevlastníme fyzicky. Nutnou podmínkou pro to, aby prostředí fungovalo, je grafická karta pro hraní her (v manuálu je uvedeno, že by mělo prostředí spolupracovat s herníma kartami, které byly prodané v posledních dvou letech).

Základem prostředí je kamera, podlaha a osvětlení. Pomocí kamery je umožněno nahlížet do prostředí, nastavuje se její pozice a směr pohledu. Je také možno definovat více kamer, které buď běží současně (náročnější na výpočet), nebo běží pouze aktivní (náročnost stejná jako v případu jediné kamery). Protože prostředí obsahuje gravitační sílu, je nutné, aby umístěné objekty nepadaly dolů a byly o něco zaraženy, k tomu slouží podlaha. Poslední základní částí prostředí je osvětlení. Na výběr jsou dva druhy světel, směrové a bodové.

Do prostředí lze přidávat různé objekty. Jednoduché lze vytvářet přímo (kostka, koule, …) a mohou se z nich skládat složitější objekty. Dále můžeme do prostředí importovat objekty v souborech *.obj vytvořené v modelovacím programu. Je nutné k nim vytvořit i *.mtl soubor ve kterém je definován vzhledový materiál přidávaného objektu, jinak je objektu přiřazen základní materiál.

Každý objekt v prostředí se skládá ze dvou částí, vizuální a fyzikální. Vizuální část slouží pouze pro vykreslení objektu do prostředí. Fyzikální část slouží při vlastní simulaci (kolize, ...). Obě části se od sebe mohou lišit, takže složitý tvar vizuální části můžeme nahradit jednodušším tvarem (kostka, koule nebo kapsle). Pokud nechceme použít zjednodušení, je možné vytvořit fyzikální část z modelu vizuálního.

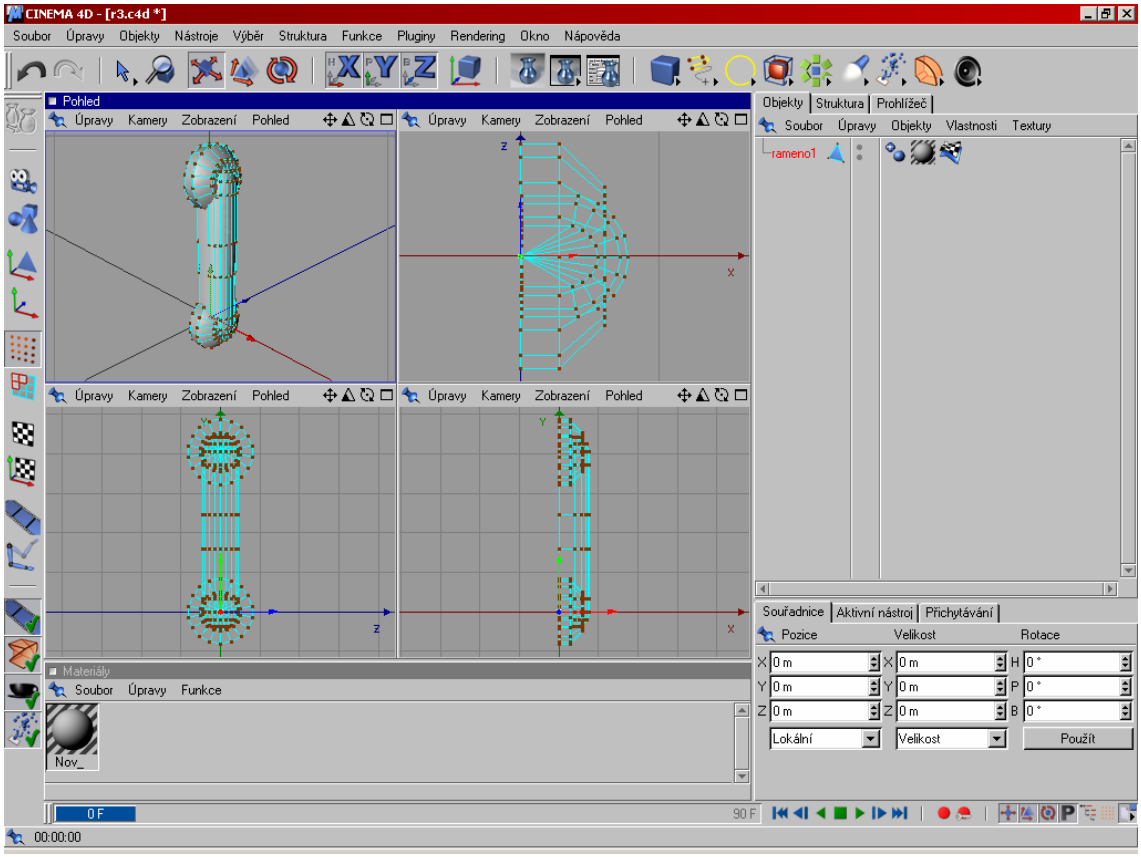
Objekty mají ještě další vlastnosti, jsou to hmotnost, hustota, materiál, pozice, orientace a rychlost. Pokud má objekt hmotnost nebo hustotu nulovou, jedná se o tzv. statický objekt, kterým nelze pohybovat. Materiál určuje, jaké bude mít objekt tření (definuje se statické a dynamické).

4 Praktická část - Model ramena

Inspirací pro práci bylo robotické rameno Mitsubischi RV-6SL. Model ramena měl být umístěn v kruhové místnosti (představuje místnost ve věžičce) a měla ho snímat kamera umístěná nad základnou. Součástí ramena mělo být pět rotačních kloubů a na konci poslední části se mělo nacházet upevněné chapadlo umožňující zvedání předmětů umístěných v místnosti.

4.1 Vytvoření modelu

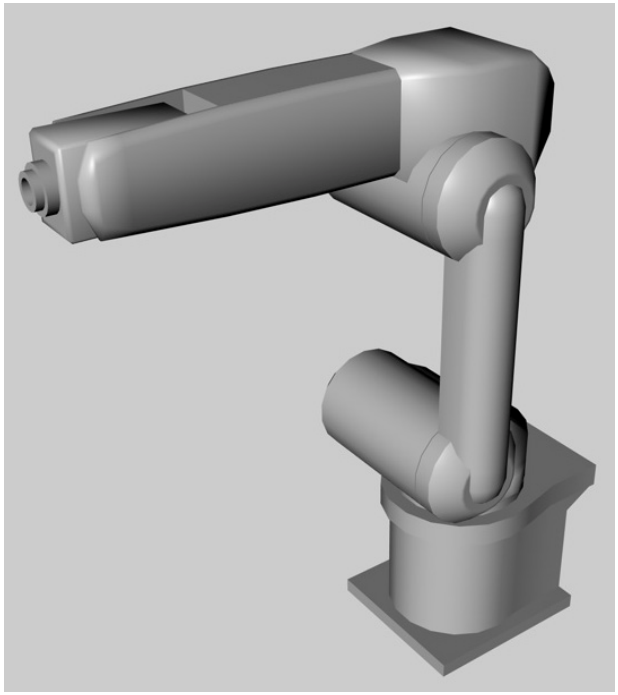
Nejdříve bylo potřeba vytvořit 3D model ramena, k čemuž byl použit program Cinema 4D CE6 od firmy MAXON. Jedná se o počeštěnou starší verzi modelovacího a animačního studia, kterou lze získat v rámci zakoupení knihy o práci s tímto programem. Současná verze tohoto programu je Cinema 4D R10.5. Jako podklad pro tvorbu byl využit reklamní materiál na robotická ramena Mitsubishi (viz příloha).



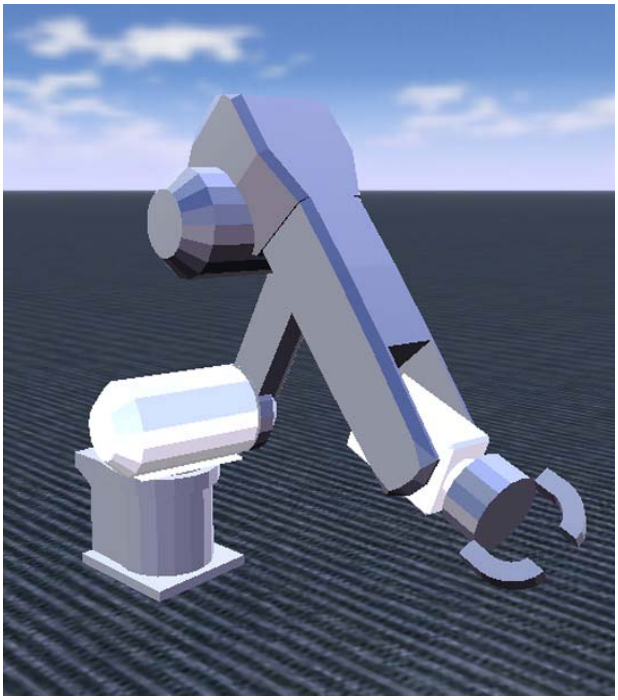
Obrázek 10 - Ukázka z programu Cinema 4D

Protože robotické rameno mělo obsahovat pět rotačních kloubů s jedním stupněm volnosti, bylo potřeba vytvořit šest částí ramena a dvě části chapadla.

Po vymodelování a rozdělení ramena na potřebné části, se musel provést export do souboru OBJ, se kterým umí MSRS pracovat. Cinema 4D CE6 neumí vyexportovat materiál do souboru MTL, který je potřebný pro správné zobrazení materiálu na objektu v MSRS. Proto bylo potřeba prostudovat formát souborů OBJ a MTL, vytvořit materiál ručně a upravit soubor objektu, aby v něm byl použit. Další možností bylo vytvořený objekt importovat do programu Blender (free open source 3D studio), který umí exportovat do souborů OBJ i MTL, vytvořit v něm materiál a provést export (čehož bylo při dalších úpravách použito).



Obrázek 11 - Celkový pohled na robotické rameno bez chapadla v Cinema 4D



Obrázek 12 - Celkový pohled na robotické rameno s chapadlem v MSRS

Robotické rameno tedy obsahuje pět rotačních kloubů, dva lineární klouby a skládá se celkem z osmi částí. Celkové rozměry ve výchozí pozici jsou 345x876x901 mm. První částí je základna o rozměrech 214x220x276 mm. Slouží jako pevně uchycená část (rám) robotického ramena a je k ní připojena pomocí kloubu část pohyblivá.

První kloub umožňuje otáčet zbytkem ramena v horizontálním směru v rozmezí $\pm$ 170° (rozsah je 340°), připojená část má rozměry 275x176x269 mm.

Druhý kloub umožňuje otáčení ramena vertikálně v rozmezí od -92° do +135° (rozsah je 227°) a připojuje rameno o rozměrech 78x569x162 mm.

Třetí kloub může vykonávat pohyb vertikálně v rozmezí od -129° do +166° (rozsah je 295°) a je přes něj připojeno předloktí o rozměrech 257x239x619 mm.

Čtvrtý kloub připojuje zápěstí o rozměrech 73x104x179 mm, které se může pohybovat vertikálně v rozmezí $\pm$ 120° (rozsah je 240°).

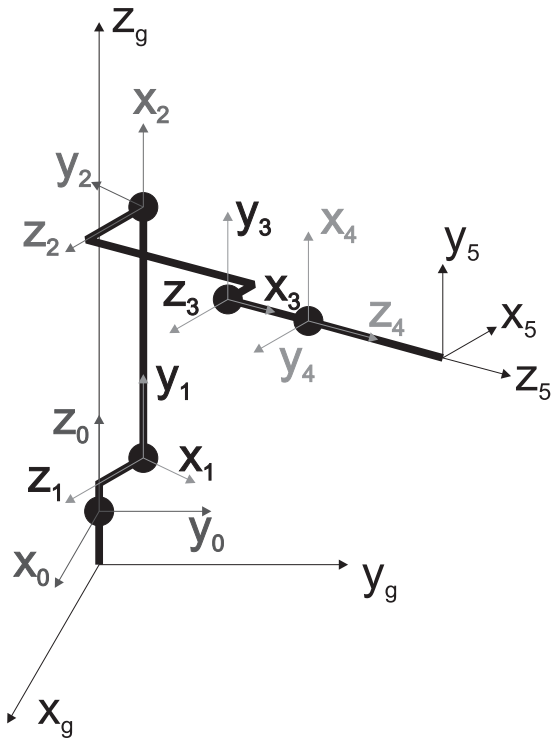
Posledním rotačním kloubem je možné otáčet volně v rozmezí 360° a je přes něj připojena část o velikosti 100x100x50 mm, ke které je připojeno chapadlo.

Chapadlo je složeno ze dvou částí o velikosti 46x22x79 mm a dvou lineárních kloubů, které umožňují chapadlo otevírat a uzavírat.

Maximální dosah robotického ramene je 1363 mm (směrem vzhůru), v ostatních směrech je hodnota dosahu menší o výšku základny. Počet stupňů volnosti je pět.

4.2 Navržení modelu souřadných systémů

Dále byly navrženy souřadné systémy kloubů podle Denavit-Hartenbergova principu. Model obsahuje šest souřadných systémů: globální (souřadný systém prostředí) a lokální (souřadné systémy jednotlivých kloubů).



Obrázek 13 - Návrh robotického ramene podle Denavit-Hartenbergova principu

Podle obrázku byla vytvořena následující tabulka, která uvádí jednotlivé parametry. Hodnoty parametrů d_i a a_i byly vzaty z vytvořeného grafického modelu.

i	γ_i	d_i	a_i	α_i
0	0	0,22	0	0
1	q1	0,094	0,095	$\pi/2$
2	q2	0	0,404	0
3	q3	-0,092	0,43	0
4	q4	0	0,085	$\pi/2$
5	q5	0	0,05	0

Pomocí této tabulky lze vytvoří transformační matice pro každý kloub ramena a lze je použít u výpočtů přímé a inverzní kinematiky.

$$A_g^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0{,}22 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_0^1 = \begin{bmatrix} \cos q1 & 0 & \sin q1 & 0{,}095 \cdot \cos q1 \\ \sin q1 & 0 & -\cos q1 & 0{,}095 \cdot \sin q1 \\ 0 & 1 & 0 & 0{,}094 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_1^2 = \begin{bmatrix} \cos q2 & -\sin q2 & 0 & 0{,}404 \cdot \cos q2 \\ \sin q2 & \cos q2 & 0 & 0{,}404 \cdot \sin q2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_2^3 = \begin{bmatrix} \cos q3 & -\sin q3 & 0 & 0{,}43 \cdot \cos q3 \\ \sin q3 & \cos q3 & 0 & 0{,}43 \cdot \sin q3 \\ 0 & 0 & 1 & -0{,}092 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^4 = \begin{bmatrix} \cos q4 & 0 & \sin q4 & 0{,}085 \cdot \sin q4 \\ \sin q4 & 0 & -\cos q4 & 0{,}085 \cdot \sin q4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad A_4^5 = \begin{bmatrix} \cos q5 & -\sin q5 & 0 & 0{,}05 \cdot \cos q5 \\ \sin q5 & \cos q5 & 0 & 0{,}05 \cdot \sin q5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Model simulovaného ramena je založen na tutoriálu, nacházejícím se v instalaci MSRS, ve kterém je ukázáno jak může být vytvořeno robotické rameno.

V projektu byly vytvořeny tři služby. První služba vytvoří scénu, která obsahuje základní prvky (podlaha, osvětlení, obloha a kamera) a robotické rameno s objekty k manipulování. Jako partnera si spouští službu ovladače, pomocí kterého se mohou robotickému ramenu vytvářet příkazy (ArticulatedArmOperations). Třetí služba přijímá operace typu ArticulatedArmOperations a předává příkazy robotickému ramenu umístěnému v simulačním prostředí.

4.3 Implementace robotického ramena

4.3.1 VisualEntity - Vytvoření třídy ramena

Jedná se o třídu, která umožňuje vykreslovat entity do simulačního prostředí. Může se také účastnit fyzické simulace, pokud je část entity instancí PhysicsEntity. Pokud tomu tak není, nemůže entita ovlivňovat ani být ovlivňována jinými objekty. Robotické rameno je reprezentováno touto třídou. Základní tvar třídy je následující:

```
public class RV6SLEntita : VisualEntity
{
       public RV6SLEntita() {...} // konstruktor
       public override void Initialize(...) {...} // inicializace objektu
       public override void Render(...) {...} // pro vykreslování
       public override void Update(...) {...} // pro aktualizaci fyz. stavu
}
```

Konstruktor slouží k počátečnímu nastavení objektu, v tomto případu je v něm vytvořena poloha (souřadnice počátku) entity, vytvořeny klouby (jejich vlastnosti a umístění) a jsou zde inicializovány všechny části robotického ramena (každá má svou funkci, která nastaví grafický i fyzický model a přiřadí klouby).

```
State.Pose.Position = new Vector3(0, 0, 0); // nastaveni pozice entity na počátek
for(int i = 0; i < _pocet_kloubu; i++) // rotační klouby
{ ... viz. Joint }
for(int i = 0; i < _pocet_kloubu_l; i++) // lineární
{ ... obdobné jako u rotačních kloubů }
if(_zakladna == null) // vytvoření jednotlivých částí ramena (viz. ArmLinkEntity)
{
   VytvorZkladnu(_joints[0]); VytvorCast1(_joints[0], _joints[1]); ...
}
for(int i = 0; i < _links.Count-1; i++) // přiřazení jmen entit ke kloubům
{
   ArmLinkEntity link = _links[i] as ArmLinkEntity; // prejmenovani
   // nastavení objektu horního přípoje
   if(link.UpperJoint != null) link.UpperJoint.State.Connectors[0].Entity = link;
   // nastavení objektu dolního přípoje
   if(link.LowerJoint != null) link.LowerJoint.State.Connectors[1].Entity = link;
}
// nastavení druhé části chapadla, protože má stejný přípojný objekt jako první
_links[_links.Count-1].LowerJoint.State.Connectors[0].Entity = _cast5;
_links[_links.Count-1].LowerJoint.State.Connectors[0].Entity = _chapadlo2;
```

Inicializace je volána, když je entita vložena do simulačního prostředí. Vloží všechny klouby robotického ramena do prostředí. Je zde dále nastavena základna jako nepohyblivá část a vypočtena obalová koule kolem ramena (jedná se o maximální dosah ramena, ve kterém může dojít ke kontaktu s jinými objekty).

```
base.Initialize(device, physicsEngine); // základní inicializace objektu
foreach(VisualEntity potomek in _links) // inicializace potomků (částí ramena)
```

```
{
    potomek.Parent = null;
    potomek.Initialize(device, physicsEngine);
}
_zakladna.PhysicsEntity.IsKinematic = true; // nastaveni základny na nepohyblivou
for(int i = 0; i < _pocet_kloubu + _pocet_kloubu_l; i++) // vložení do fyz. scény
    physicsEngine.InsertJoint((PhysicsJoint)_joints[i]);
// vytvoření kolizní koule, max. dosah ramena, ve kterém může dojít k interakci
xna.Vector3 stred = xna.Vector3.Zero; // (0,0,0)
float polomer = 1.363f; // maximální dosah ramena
EntityBoundingSphere = new xna.BoundingSphere(stred, polomer);
```

Metoda pro vykreslování je volána pro aktualizaci zobrazení entity. Zde je kromě standardního vykreslení voláno ještě vykreslení pro každou část ramena zvlášť, jinak by nebyla zobrazena.

```
base.Render(renderMode, transforms, currentCamera); // standardní vykreslení
foreach (ArmLinkEntity link in _links) // zvlášť pro každou část ramena
    link.Render(renderMode, transforms currentCamera)
```

Update je volán pro aktualizaci fyzických částí entity. V případě robotického ramena to je fyzická poloha a natočení ramen. Opět se musí volat pro všechny části ramena.

```
base.Update(update); // standardní update
foreach (ArmLinkEntity link in _links) // zvlášť pro každou část ramena
    link.Update(update)
```

4.3.2 Joint - Vytvoření kloubu

Před přiřazením kloubu k objektu je potřeba nejdříve vytvořit a nastavit jeho vlastnosti. Kloub se účastní fyzické části simulace a odtud pochází i nastavované vlastnosti. Uvádím příklad vytvoření rotačního kloubu, u lineárního se postupuje obdobně.

```
// vytvoření vlastnosti rotačního kloubu
JointAngularProperties angular = new JointAngularProperties();
```

V závislosti na typu kloubu se liší nastavované vlastnosti, rozdílné nastavení je např. u kloubu bez omezení pohybu a s limitovaným pohybem. Pokud je nastaven limitovaný pohyb, je dále potřeba nastavit horní a spodní hranici limitu. Každý stupeň volnosti se musí nastavovat zvlášť, pokud údaj není nastaven, je brán jako nepoužitý.

```
// nastavení vlastností rotačního kloubu
angular.TwistMode = JointDOFMode.Free; // nastaveni neomezeného kroutivého pohybu
angular.TwistDrive = new JointDriveProperties(
       JointDriveMode.Position, // nastaveni způsobu řízení motoru
       new SpringProperties(500000, 100000, 0), // nastavení vlastností motoru
       100 // nastavení maximálního limitu síly
       );
```

Nyní je nastaveno, že má kloub jeden stupeň volnosti (kroutivý pohyb), motor používá poziční mód (řídí se zadáváním pozice, na kterou se má vystavit). U vlastností motoru je první hodnota pružící koeficient, druhá tlumení (určuje rychlost přesunu do nové pozice) a poslední je hodnota pozice rovnováhy. Limit síly je maximální hodnota, kterou může motor použít. Další způsob nastavení motoru je mód rychlostní (řídí se zadáváním vektoru rychlosti).

```
// vytvoření kloubu
PhysicsJoint kloub = PhysicsJoint.Create(new JointProperties(angular, null, null));
// nastavení jména
kloub.State.Name = "Kloub";
```

Teď je kloub vytvořen, při vytváření se zadaly nastavené vlastnosti. Položky null jsou horní a dolní přípojný bod, které budou nastaveny později. Každý fyzický objekt v simulaci musí mít své jedinečné jméno, takže mu bylo přiřazeno dosud nepoužité.

```
// nastaveni horního přípojného bodu
kloub.State.Connectors[0] = new EntityJointConnector(
       null, // objekt, ke kterému je bod připojen
       new Vector3(0,1,0), // normála kloubu
       new Vector3(0,0,1), // osa kloubu
       new Vector3(0,0,rozmer6.Z), // pozice umístění bodu na objektu
       );
// nastaveni dolního přípojného bodu
kloub.State.Connectors[1] = new EntityJointConnector(
       null, // objekt, ke kterému je bod připojen
       new Vector3(0,1,0), // normála kloubu
       new Vector3(0,0,1), // osa kloubu
       new Vector3(0,rozmer.Y/2,rozmer.Z), // pozice umístění bodu na objektu
       );
```

Kloub je připraven pro použití, zbývá jen doplnit odkazy na entity objektů, ke kterým bude připojen.

4.3.3 ArmLinkEntity - Část ramena

Každá část robotického ramena je tvořena třídou ArmLinkEntity, která umožňuje nastavit jak fyzickou, tak i grafickou složku.

```
ArmlinkEntity _cast1;
[DataMember]
public ArmLinkEntity Cast1
{
       get { return _cast1; } // vrací proměnnou
       set { _cast1 = value; } // nastavuje proměnnou
}
```

Nyní je vytvořena proměnná, do které bude vytvořená část robotického ramena uložena. Část programu pod [DataMember] umožňuje DSSP tuto proměnnou číst a nastavovat. Dále je potřeba vytvořit fyzickou a grafickou reprezentaci ramena.

```
BoxShape fyzickyTvar = new BoxShape(
       new BoxShapeProperties(
              hmotnost, // hmotnost fyzické části
              new Pose(new Vector3(0, rozmer_z.Y+rozmer_r2.Y/2, 0.1f)), // pozice
              rozmer_r2 // rozměr fyzické části
       ));
```

ArmLinkEntity podporuje dva typy fyzických tvarů. První je BoxShape, který je použit u všech části robotického ramena a CapsuleShape.

```
ArmLinkEntity link = new ArmLinkEntity(
       "RV6SL - cast 1", // název entity
       "rr2.obj", // název souboru s grafickým modelem entity
       new Pose( // pozice grafického modelu
              new Vector3(0, -rozmer_r2.Y/2, -0.1f), // pozice
              TypeConversion.FromXNA(
                     xna.Quaternion.CreateFromAxisAngle( // orientace
                     new xna.Vector3(0, 1, 0), 0)
                     )
              )
       );
```

V proměnné link je vytvořená nová část robotického ramena, jako další objekt ve scéně musí mít své jméno. Soubory s modely entit jsou uloženy v podadresáři Microsoft Robotics Studia \store\media. Pozice grafického modelu je udána vzhledem k pozici fyzického tvaru.

```
link.LowerJoint = kloub; // nastavení dolního kloubu
link.UpperJoint = kloub2; // nastavení horního kloubu
_cast1 = link; // uložení vytvořené ArmLinkEntity do vybrané proměnné
_cast1.BoxShape = fyzickyTvar; // přiřazení fyzického tvaru
```

Zde byl nastaven horní a dolní kloub, pokud by jeden nebo druhý nebyl nastaven (hodnota null), pak by byla tato část ramena považována za koncovou. Dále byla vytvořená část uložena do proměnné přístupné přes DSSP a byl jí přiřazen fyzický tvar.

4.4 Ovládací služba

Aby se dalo robotické rameno ovládat, musela být vytvořena služba, která bude přijímat zadané příkazy a bude je předávat entitě umístěné v simulaci. Hlavní část služby tvoří arbitery a handlery. Main port služby typu je ArticulatedArmOperation, aby mohl přijímat tyto operace. Stav služby je také upraven pro ArticulatedArmState.

```
// stav služby
arm.ArticulatedArmState _state = new arm.ArticulatedArmState();
// main port služby
arm.ArticulatedArmOperations _mainPort = new arm.ArticulatedArmOperations();
```

Při startu služby se vytvoří a zaregistruje port pro subskripce, poté jsou vytvořeny arbitery, které poslouchají na portech a čekají na oznámení o vložení entit do scény nebo na ukončení.

```
// vytvoření portu pro notifikace (jedná se o port simulačního prostředí)
_notifikace = new engine.SimulationEnginePort();
engine.SimulationEngine.GlobalInstancePort.Subscribe( // zaregistrování subskripcí
   ServiceInfo.PartnerList, _notifikace);
Activate(
   new Interleave(
      new TeardownReceiverGroup(
            Arbiter.Receive<engine.InsertSimulationEntity>( // vložení
                  false, _notifikace, InsertNotificationHandler),
            // ukončení
            Arbiter.Receive<DsspDefaultDrop>(false, _mainPort, DefaultDropHandler)
      ),
      new ExclusiveReceiverGroup(),
      new ConcurrentReceiverGroup()
   ));
```

Po přijmutí oznámení o vložení entity do prostředí je přijata zpráva o této události na portu _notifikace a je spuštěn handler pro tuto událost. Ten vykoná načtení stavu objektu ze scény a provede standardní spuštění služby.

```
// načtení entity ze scény (ins je parametr handleru)
_entita = (Robotics.Test.RV6SLEntita)ins.Body;
// nastavení ovládací služby entity na tuto
_entita.ServiceContract = Contract.Identifier;
// nastavení stavu kloubů služby na shodné jak je tomu u entity
_state.Joints = _entita.Joints;
// vytvoření kolekce pro vyhledávání kloubů podle jejich jména
_jointLookup = new Dictionary<string, Joint>();
// vložení všech kloubů do kolekce
foreach (Joint j in _state.Joints) _jointLookup.Add(j.state.Name, j);
base.Start(); // standardní spuštění služby
```

Nyní je ovládací služba připravena pro přijímání úkolů pro robotické rameno. Dále je potřeba vytvořit handlery pro jednotlivé operace, které budou používány. Takže bude vytvořen handler pro nastavování pozice a orientace kloubů.

```
// nalezení kloubu, pro který je operace určena (update je parametr handleru)
Joint j = _jointLookup[update.Body.JointName];
if(j.State.Angular != null) // pro rotační klouby
   _entita.NastavOrientaci(j, update.Body.TargetOrientation); // zadání akce
else // pro lineární klouby
   _entita.NastavPozici(j, update.Body.TargetPosition); // zadání akce
// upravení update pro odeslání jako notifikace
update.ResponsePort.Post(DefaultUpdateResponseType.Instance);
// odeslání notifikace o provedené operaci
base.SendNotification(_submgrPort, update);
```

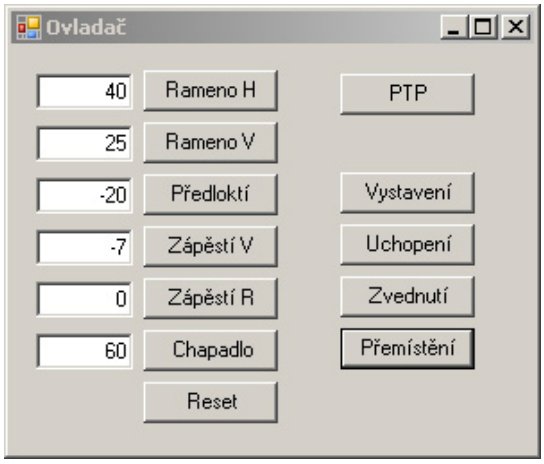
Při spuštění handleru dojde nejdříve k nalezení kloubu, pro který je operace určena. Poté se zjistí, jestli se jedná o rotační nebo lineární kloub (všechny klouby ramena kromě chapadla jsou rotační, tj. Kloub5 a Kloub6 jsou lineární). U kloubů rotačních se nastavuje natočení pomocí metody NastavOrientaci a u kloubů lineárních se nastavuje pozice pomocí metody NastavPozici. V parametrech těch metod se předává kloub, kterého se úkon týká a hodnota, o kterou se má přemístit. Tyto metody zatím nebyly zmíněny a budou uvedeny nyní. Dále není uveden typ portu _submgrPort. Jedná se o port partnera, který se stará o řízení subskripcí.

```
public void NastavOrientaci(Joint j, AxisAngle axisAngle)
{
    // vytvoření úlohy na změnu orientace určitého kloubu
    Task<Joint,AxisAngle> pozadavek = new Task<Joint, AxisAngle>
        (j, axisAngle, NastavOrientaciHandler);
    // zadání úlohy ke zpracování
    DeferredTaskQueue.Post(pozadavek);
}
public void NastavOrientaciHandler(Joint j, AxisAngle axisAngle)
{
    if(j.State.Angular != null)
    {
        ... // ošetření výjimek
        Quaternion target = TypeConversion.FromXNA( // vytvoření orientace kloubu
               xna.Quaternion.CreateFromAxisAngler(
                      TypeConversion.ToXNA(axisAngle.Axis), axisAngle.Angle)
        );
        ((PhysicsJoint)j).SetAngularDriveOrientation(target); // provedeni operace
    }
}
public void NastavPozici(Joint j, Vector3 pozice)
{
    // vytvoření úlohy na změnu pozice určitého kloubu
    Task<Joint,Vector3> pozadavek = new Task<Joint, Vector3>
        (j, pozice, NastavPoziciHandler);
    // zadání úlohy ke zpracování
    DeferredTaskQueue.Post(pozadavek);
}
public void NastavPoziciHandler(Joint j, Vector3 pozice)
{
    // provedení operace
    if(j.State.Linear != null) ((PhysicsJoint)j).SetLinearDrivePosition(pozice);
}
```

Metody mají obdobnou konstrukci, liší se jen v typu parametru a způsobu vykonávání operací v závislosti na typu kloubu. Nejdříve se vytvoří požadavek na operaci a ta je poté zadána ke zpracování. Při zpracování se přes metodu kloubu nechá změnit orientace nebo pozice. Požadavky jsou vytvořeny tímto způsobem kvůli tomu, aby se klouby mohly pohybovat paralelně.

4.5 Ovládací panel

Ovládací panel je další služba. Tvoří ji uživatelské rozhranní ve formě Windows formuláře a umožňuje zadávání příkazů entitě robotického ramena umístěné v simulačním prostředí, pro kterou je vytvořen port _armPort (typu ArticulatedArmOperations).



Obrázek 14 - Ukázka ovladače robotického ramena

Při startu služby je vytvořen arbiter, který poslouchá na portu formuláře a na základě typu operace předává úkol k vykonání.

```
Activate(
       Arbiter.Interleave(
              new TeardownReceiverGroup(),
              new ExclusiveReceiverGroup(),
              new ConcurrentReceiverGroup( // konkurenční přijímání
                     Arbiter.ReceiveWithIterator<OnApplyJoint>
                            (true, _ovladacPort, OnApplyJointHandler)
                     Arbiter.ReceiveWithIterator<OnApplyJointPosition>
                            (true, _ovladacPort, OnApplyJointPositionHandler)
              )
       ));
SpawnIterator(StartOvladani);
WinFormsServicePort.Post(new RunForm(VytvorOvladac));
```

Při startu je dále vytvořen ovládací formulář a je navázáno spojení s partnerem (Ovládací služba), kterému jsou zasílány příkazy pro robotické rameno.

```
// pro vytvoření ovládacího formuláře
System.Windows.Forms.Form VytvorOvladac() { return new Ovladac(_ovladacPort); }

// navázání spojení s ovládací službou
IEnumerator<ITask> StartOvladani()
{
    // vytvoření portu podle partnera
    DsspResponsePort<CreateResponse> resultPort = CreateService(
        Robotics.TestControlProxy.Contract.Identifier, // identifikátor partnera
        Microsoft.Robotics.Simulation.Partners.CreateEntityPartner("
               http://localhost/RV6SL - rameno")
    );
```

```
    yield return Arbiter.Choice(resultPort,
        delegate(CreateResponse response)
        {       // vytvoření portu pro zasílání operací
                _armPort = ServiceForwarder<arm.ArticulatedArmOperation>
                        (response.Service);
        }
        delegate(Fault fault){ LogError(fault); }
        );
}
```

Zaregistrované arbitery mají své handlery, které vykonají úkoly podle hodnot zadaných v parametru. Uvedený příklad je pro operaci OnApplyJoint (operace pro nastavování rotačních kloubů).

```
IEnumerator<ITask> OnApplyJointHandler(OnApplyJoint oa)
{
    arm.SetJointTargetPoseRequest req = new arm.SetJointTargetPoseRequest();
    req.JointName = oa.Kloub; // přiřazení názvu kloubu
    phymodel.AxisAngle axisAngle = new phymodel.AxisAngle(
        new phymodel.Vector3(1, 0, 0), // osa
        (float)(oa.Uhel * Math.PI / 180) // úhel v radiánech
        );
    req.TargetOrientation = axisAngle; // uložení orientace do požadavku
    _armPort.SetJointTargetPose(req);
    yield break;
}
```

Ovladač umožňuje ovládat robotické rameno pomocí nastavování kloubových proměnných. U rotačních kloubů se nastavuje hodnota ve stupních a u translačních v milimetrech. Tlačítko "Rameno H" nastavuje horizontální natočení ramena (první kloub). "Rameno V" nastavuje vertikální natočení ramena (druhý kloub). "Předloktí" nastavuje natočení předloktí (třetí kloub). "Zápěstí V" nastavuje vertikální natočení zápěstí (čtvrtý kloub). "Zápěstí R" nastavuje rotační natočení chapadla (pátý kloub). Tlačítko "Chapadlo" nastavuje rozevření chapadla v rozmezí 0-100 mm, hodnota 0 je úplné rozevření a hodnota 100 plné sevření. "Reset" nastavuje robotické rameno do výchozí polohy (všechny klouby mají nastavenou kloubovou proměnnou na 0 a chapadlo je plně otevřené). Tlačítko "PTP" umožňuje provedení asynchronního vystavení do pozic uvedených u jednotlivých kloubů. Zbylá čtyři tlačítka provádí přednastavené operace s robotickým ramenem a byla použita při pokusech o zvedání předmětu ("Vystavení" - umístění ramena do polohy pro uchopení předmětu, "Uchopení" - uchopí předmět sevřením chapadla o hodnotu 60 mm, "Zvednutí" - provede přesun předloktí o 3° nahoru (z pozice po vystavení) a "Přemístění" otočí horizontálně ramenem o 40°).

5 Praktická část - Zvedání objektů

Úkolem bylo zvedání předmětů o různém tvaru a hmotnosti. Byly zvoleny předměty kostka, koule a upravená kostka s výstupky usnadňující uchopování. Použité hmotnosti byly 0,01 kg, 0,1 kg, 1 kg, 10 kg a 100 kg. U každého předmětu mělo být provedeno deset pokusů o zvednutí (pět různých hmotností a dva různé hodnoty tření předmětu), v nejasných případech bylo provedeno pokusů více. Hodnoty tření byly 50 % a 100 %.

```
// způsob nastavení materiálu pro tření
kostka.BoxShape.State.Material = new MaterialProperties("JmenoMateriálu",0,1,1);
```

Předposlední a poslední hodnota v MaterialProperties udává hodnotu dynamického a statického tření. Rozmezí hodnot se pohybuje od 0 do 1, takže pro nastavení na hodnotu 50% tření muselo být nastaveno 0.5f a pro hodnotu 100 % bylo nastaveno 1. Při pokusech byla hodnota dynamického a statického tření stejná.

Do scény byl přidán stůl, na který byly pokusné předměty umísťované, a poté by byl proveden pokus o zvednutí. Pokus se skládal z vystavení robotického ramena do uchopovací polohy (tj. nastavení kloubových proměnných do vektoru (0, 25, -17, -7, 0, 0) a přemístění do požadované pozice asynchronním pohybem pomocí tlačítka "PTP" na ovladači), uchopení předmětu (nastavení kloubové proměnné chapadla, hodnota byla proměnná od 50 do 60), zvednutí předmětu (vystavení ramena do pozice (0, 25, -20, -7, 0, 0)) a vykonání otočení o 40° doleva s uchopeným předmětem (nastavení kloubové proměnné "Rameno H" a provedení vystavení pomocí tlačítka). Zjišťovalo se, zda je rameno schopno testovaný předmět zvednout a při přesunu neupustit.

Nastavení motorů kloubů bylo u všech pokusů shodné (u rotačních byla hodnota maximální síly nastavena na 100 a u lineárních na 1000).

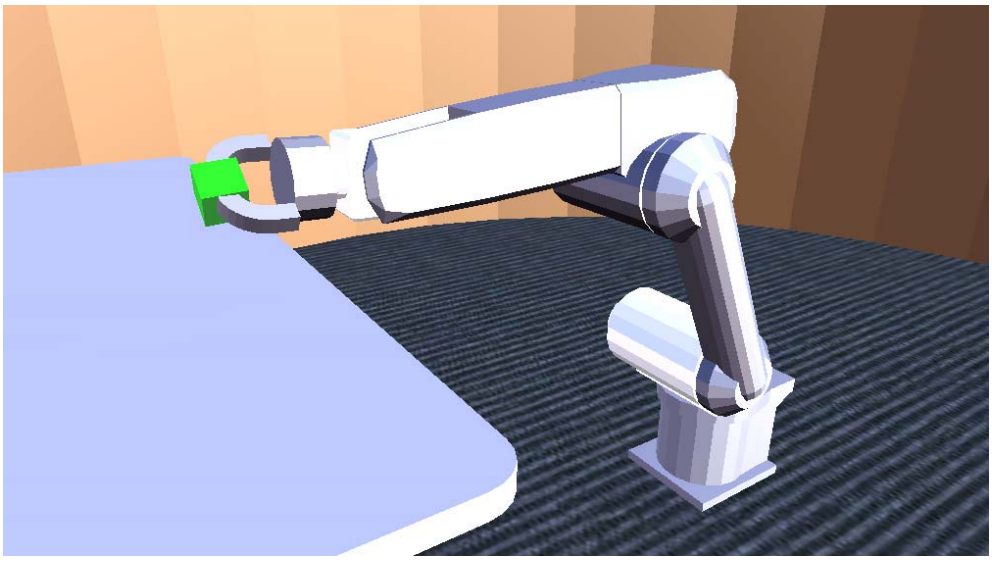
Výsledky pokusů jsou u každého předmětu shrnuty v tabulce. Ve sloupci *Zvednutí* se může nacházet hodnota *Ano* nebo *Ne* a ve sloupci *Přesun* se mohou vyskytnout hodnoty *Ano*, *Ne* nebo *Ano/Ne* (označující jiný výsledek).

5.1 Zvedání kostky

Prvním předmětem určeným ke zvedání byla vybrána kostka. Jednalo se o předmět ve tvaru krychle s rozměry 5 x 5 x 5 cm. Výsledky jednotlivých pokusů byly následující:

Hmotnost	Tření	Zvednutí	Přesun
0,01 kg	50 %	Ano	Ano
0,01 kg	100 %	Ano	Ano
0,1 kg	50 %	Ano	Ano/Ne
0,1 kg	100 %	Ano	Ano/Ne
1 kg	50 %	Ne	-
1 kg	100 %	Ne	-
10 kg	50 %	Ne	-
10 kg	100 %	Ne	-
100 kg	50 %	Ne	-
100 kg	100 %	Ne	-

U hmotnosti 0,01 kg proběhlo zvednutí a přemístění předmětu v pořádku. Hmotnost 0,1 kg nečinila problém předmět zvednout, ale problém dělal přesun. Při přesunu 0°-40° byl předmět upuštěn (kvůli příliš velké akceleraci/deakceleraci), proto byl proveden další pokus, u kterého byla dráha přesunu rozdělena po 10°. Při této úpravě byl přesun úspěšně dokončen. U hmotnosti 1 kg kostka proklouzávala již při pokusu o zvednutí. Další hmotnosti se nepodařilo zvednout.



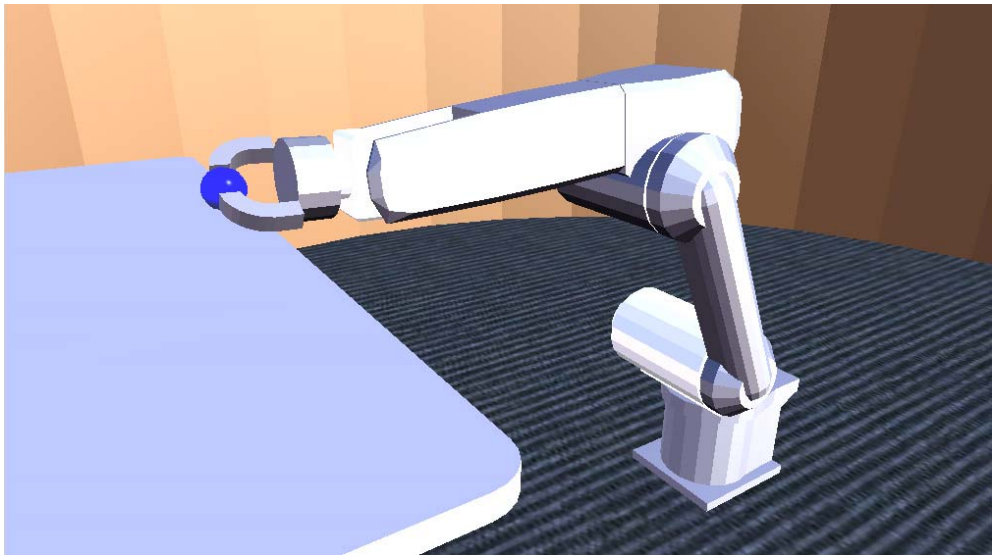
Obrázek 15 - Ukázka z pokusu při zvedání kostky

5.2 Zvedání koule

Druhý předmětem zvoleným ke zvedání byla koule, jejíž průměr byl 5 cm. Výsledky po provedení pokusů byly následující:

Hmotnost	Tření	Zvednutí	Přesun
0,01 kg	50 %	Ano	Ano
0,01 kg	100 %	Ano	Ano
0,1 kg	50 %	Ano	Ano
0,1 kg	100 %	Ano	Ano
1 kg	50 %	Ano	Ano/Ne
1 kg	100 %	Ano	Ano/Ne
10 kg	50 %	Ne	-
10 kg	100 %	Ne	-
100 kg	50 %	Ne	-
100 kg	100 %	Ne	-

Při zvedání tohoto předmětu byla upravena pozice vystavení "Zápěstí V" na hodnotu -5, která umožňovala uchopení koule přibližně v jejím průměru. Zvedání koule probíhalo v pořádku až do hmotnosti 0,1 kg. Při hmotnosti 1 kg se dařilo předmět zvednout, ale při přesunu 0°-40° byl upuštěn. Při opakovaném pokusu byla dráha rozdělena po 10°, v tomto případě byl přesun dokončen v pořádku. Větší hmotnosti rameno nedokázalo zvednout.



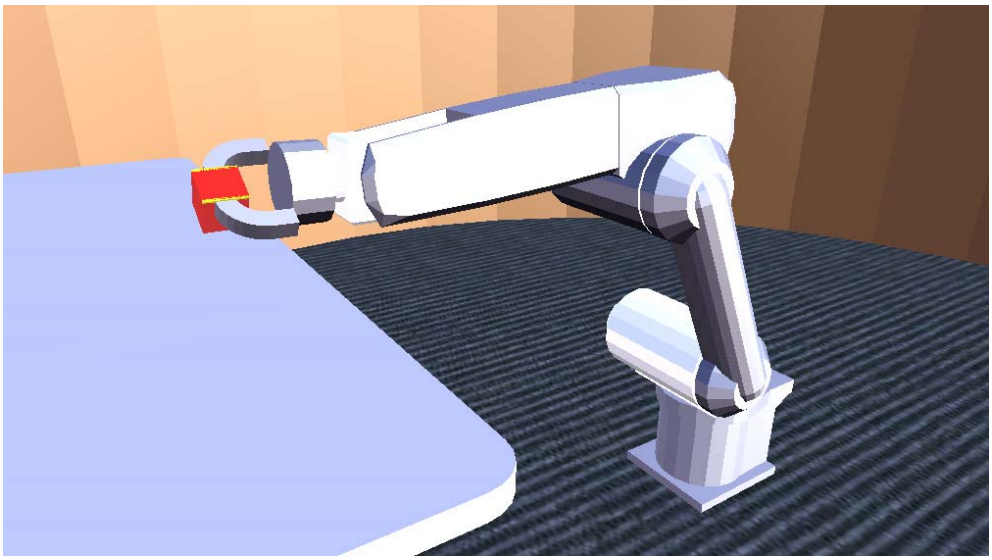
Obrázek 16 - Ukázka z pokusu při zvedání koule

5.3 Zvedání kostky s výstupky

Ze zkušeností získaných při zvedání předchozích předmětů byl vytvořen nový předmět. Jednalo se o kostku s rozměry 5 x 5 x 5 cm, ke které byly z pravé a z levé strany přidány výstupky o rozměrech 4 x 2 x 50 mm zarovnané s horní hranou kostky. Výsledky zvedání byly následující:

Hmotnost	Tření	Zvednutí	Přesun
0,01 kg	50 %	Ano	Ano
0,01 kg	100 %	Ano	Ano
0,1 kg	50 %	Ano	Ano
0,1 kg	100 %	Ano	Ano
1 kg	50 %	Ano	Ano
1 kg	100 %	Ano	Ano
10 kg	50 %	Ne	Ne
10 kg	100 %	Ne	Ne
100 kg	50 %	Ne	Ne
100 kg	100 %	Ne	Ne

Provedené pokusy probíhaly za stejných okolností jako v předchozích případech až na tu výjimku, že chapadlo nestiskávalo předmět, ale bylo sevřeno jen na jeho šířku (tj. 5 cm). Zvednutí bylo umožněno výstupky, které zamezily vyklouznutí předmětu z chapadla. Rameno bylo schopno zvednout předměty do hmotnosti 1kg, těžší se zvednout nezdařilo.



Obrázek 17 - Ukázka z pokusu při zvedání kostky s výstupky

5.4 Zvedání objektů - shrnutí

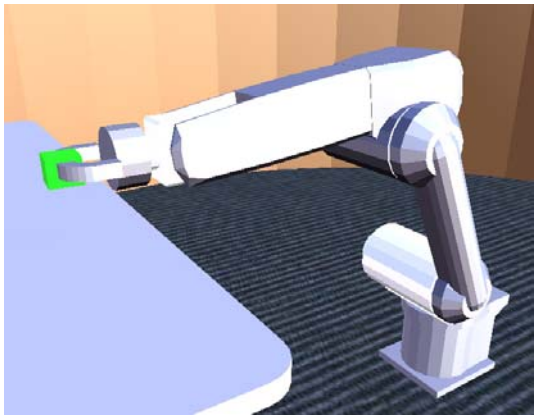
Nejlepších výsledků bylo dosaženo při zvedání kostky s výstupky, protože nemohlo dojít k proklouznutí objektu při nedostatečném stisku a tím bylo umožněno jeho lehčí zvednutí a přesunutí.

Objekty bez výstupků, o které by se mohlo chapadlo zarazit, bylo problémové zvednout při vyšší hmotnosti, protože při pokusu o zvednutí proklouzávaly mezi částmi chapadla. Proto byly provedeny další pokusy, ve kterých byla měněna míra sevření chapadla, a zjišťoval se vliv na úspěšnost zvednutí. Při těchto pokusech byl dále sledován vliv tření.

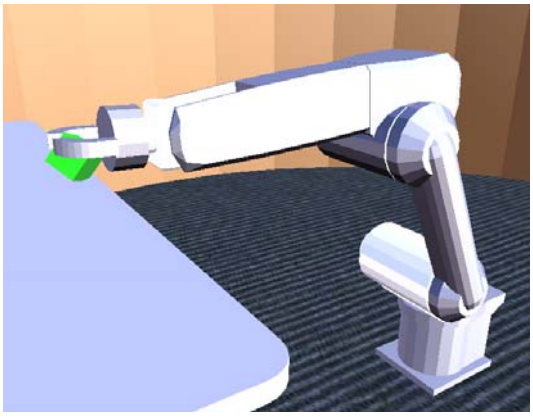
Bylo zjištěno, že tření mělo na zvedání neznatelný vliv. Uvedená tabulka ukazuje příklad, který popisuje, jak se uchopovaný předmět choval (u jiných hmotností bylo chování obdobné, jen se lišily hodnoty stisku). Sloupec *Zvednutí* udává hodnotu stisku, při kterém byl objekt úspěšně zvednut. Sloupec *Vibrace* udává hodnotu stisku, u kterého začal objekt při zvednutí vibrovat. Poslední sloupec udává hodnotu, při které se začalo projevovat chybové chování (viz. dále).

Objekt: Kostka, Hmotnost: 0,1 kg			
Tření	Zvednutí (hodnota stisku)	Vibrace (hodnota stisku)	Chyba (hodnota stisku)
50 %	53 - 56	57	od 59
100 %	53 - 56	57	od 59

Při pokusech o větší stisk objektu docházelo k neočekávanému chování objektu i chapadla. Objekt se natáčel do pozic, které by mu fyzikálně neměly být umožněny, ale i přesto se to dělo (viz. Obrázek 18 a Obrázek 19). Chapadlu se uchopovací části umisťovaly mimo zadanou polohu. Pokusy o korekci tohoto chování o mnoho výsledné chování nezlepšily.



Obrázek 18 - Rameno po uchopení kostky větším sevřením



Obrázek 19 - Stav ramena po zvednutí objektu (objekt se zkroutil do jiné pozice)

6 Diskuze validity modelu a uplatnění v předmětu Robotika

V této kapitole je diskutována validita modelu simulovaného ramena vzhledem ke skutečnému a možnost uplatnění modelu v předmětu Robotika.

6.1 Validita modelu

Velikost modelu robotického ramena odpovídá přibližně velikosti opravdového ramena. Tvar předposledního článku (umožňuje rotaci chapadla) je vymyšlený, protože nebyl dostatečně kvalitní obrázek se vzhledem. Taktéž tvar chapadla a rozmezí jeho pohybu jsou smyšlené.

Hmotnosti jednotlivých částí nebyly nikde uvedeny, byla uvedena pouze přibližná hmotnost celého robotického ramena (cca 60 kg). Proto byly hmotnosti jednotlivých částí zvoleny odhadem podle jejich velikosti a předpokládaných částí (velký motor, malý motor, atd.). Jednotlivé hmotnosti: základna - 20 kg (předpokládaný motor), otáčecí část (předpokládaný motor) - 15 kg, rameno - 5 kg (jen propojení), předloktí - 12 kg (předpokládaný motor), zápěstí - 5kg (předpokládaný menší motor), rotační část 1 kg a každá část chapadla 1 kg (celkem 60 kg).

Fyzická reprezentace modelu se skládá z kvádrových objektů, takže rozložení hmotnosti je jiné než u skutečného ramena. Dále byla upravena velikost kvádru reprezentujícího rameno a předloktí, šlo o zmenšení velikostí (ostatní části mají velikost krychle rovnu svým maximálním rozměrům, výjimku ještě tvoří chapadlo), kvůli kontaktu s dalšími částmi robotického ramena a bránění v pohybu.

Dalším odlišným prvkem od skutečnosti je síla a parametry jednotlivých motorů kloubů, protože v reklamních materiálech nejsou uvedené (uvedeny jsou pouze rychlosti otáčení).

6.2 Uplatnění v předmětu Robotika

Model robotického ramena by mohl sloužit jako ukázka při probírání kloubů a kloubových proměnných, kde by se ukázal stav kloubů v simulačním prostředí přes webový prohlížeč v xml reprezentaci, pak by se provedla změna natočení jednoho nebo dvou kloubů a předvedly by se změny v reprezentaci.

Další uplatnění robotického ramena by mohla být ukázka přímého programování, ale musel by být upraven ovladač, aby umožňoval zapamatování jednotlivých pozic kloubu s patřičným časem a po naprogramování umožnil spustit zadaný program.

7 Závěr

Díky znalostem, získaným při seznamování se s modelováním robotických ramenech ve vývojovém prostředí Microsoft Robotics Studio, byl vytvořen funkční model ramena. Dále byly vytvořeny související služby, které vytvořily scénu, vložily do ní rameno a umožnily jeho ovládání.

Model se skládal z pěti rotačních kloubů, sloužících pro vystavování do různých pozic v prostředí, a ze dvou lineárních kloubů, které ovládaly chapadlo, umožňující zvedání předmětů. S ramenem byly prováděny pokusy, při nichž byly zvedány předměty s různými tvary a hmotnostmi. Nejlépe dopadl předmět s tvarem, který neumožňoval vyklouznutí předmětu z chapadel.

Navazující práce na projektu by mohly směřovat k vývoji nových služeb, které by umožňovaly nové způsoby ovládání, implementovaly a testovaly algoritmy atd.:

- nalezení objektu pomocí obrázku ze statické kamery snímající scénu a následné vystavení ramena do pozice pro uchopení
- přidání senzorů, umožňujících přesnější orientaci a navigaci v prostoru (dálkoměr, ...)
- testovaní algoritmů pro zjišťování polohy a vystavování do nové polohy (přímá a inverzní kinematika)
- testování algoritmů pro plánování

Literatura

[1] ORSÁG, Filip. Studijní opora. *Robotika* [online]. 2006 [cit. 2008-01-02], s. 6-17.

[2] CCR User Guide. *Microsoft Robotics Studio User Guide* [instalace]. 2007 [cit. 2008-01-02].

[3] DSS User Guide. *Microsoft Robotics Studio User Guide* [instalace]. 2007 [cit. 2008-01-02].

[4] Simulation Tutorial 4 (c#). *Microsoft Robotics Studio User Guide* [instalace]. 2007 [cit. 2008-01-02].

[5] *Microsoft Robotics Studio : Class Reference* [online]. 2007 [cit. 2008-01-02]. Dostupný z WWW: <http://msdn2.microsoft.com/en-us/library/bb881626.aspx>.

[6] *Karel Čapek - O původu slova robot* [online]. 1998-2005 [cit. 2008-01-16]. Dostupný z WWW: <http://capek.misto.cz/robot.html>.

[7] *Isacc Asimov* [online]. 2008 [cit. 2008-01-16]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Isaac_Asimov>.

[8] HLAVÁČ, Václav. Přednáška. *Úvod do robotiky* [online]. 2006 [cit. 2008-01-16]. Dostupný z WWW: <cmp.felk.cvut.cz/~hlavac/TeachPresCz/51Robotika/01UvodRobotika.pdf>.

[9] Manipulace a montáž hrou. *Automatizace* [online]. 2005, roč. 48, č. 5 [cit. 2008-01-16]. Dostupný z WWW: <http://www.automatizace.cz/article.php?a=688>.

[10] Robotika. *Základy automatizace* [online]. 2006 [cit. 2008-01-16]. Dostupný z WWW: <http://sipal.utrv.ujep.cz/Auto/Auto_10.pdf>.

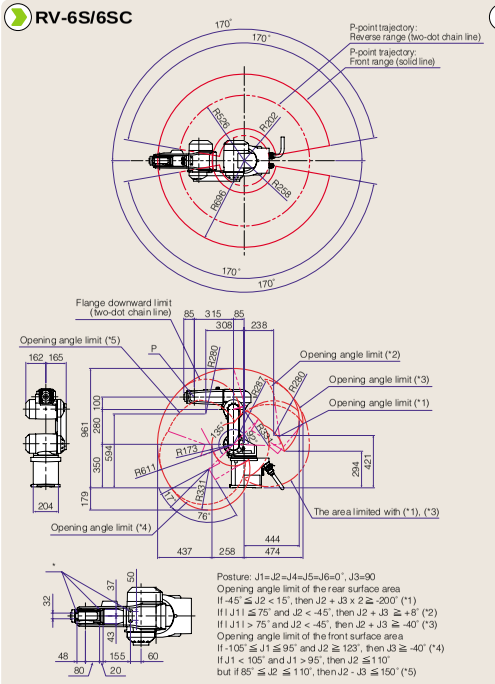
[11] *KUKA Industrial Robots - Educational Framework* [online]. 2007 [cit. 2008-01-16]. Dostupný z WWW: <http://www.kuka.com/usa/en/products/software/educational_framework>.

[12] *Teorie průmyslových robotů* [online]. 2000 [cit. 2008-01-16]. Dostupný z WWW: <www.dzego.me.cz/tul/robotika/teorie_robotu/teorie_robotu.pdf>.

Seznam příloh

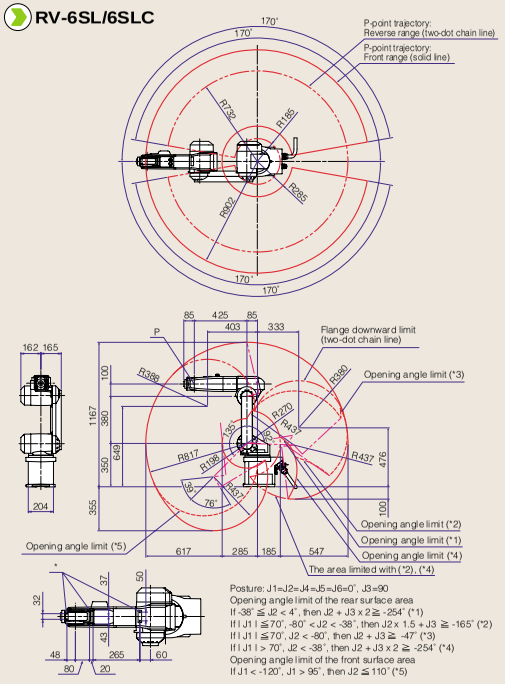
Robot Arm External Dimension/Movement Range Diagrams

RV-6S/6SC



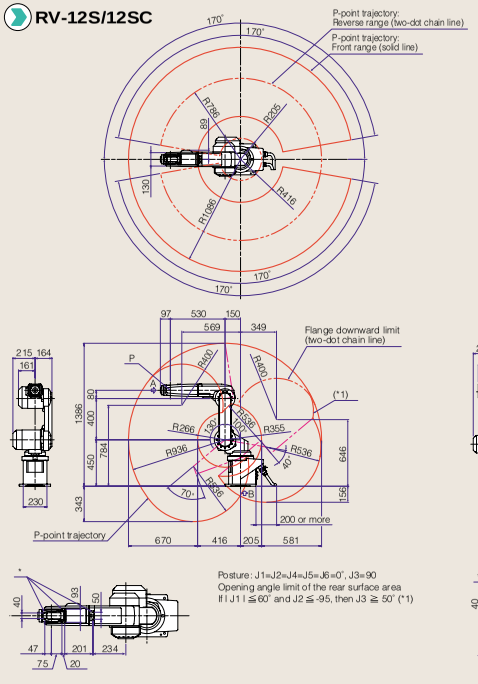
* indicates service screw holes for tooling (M4×6 places).

RV-6SL/6SLC



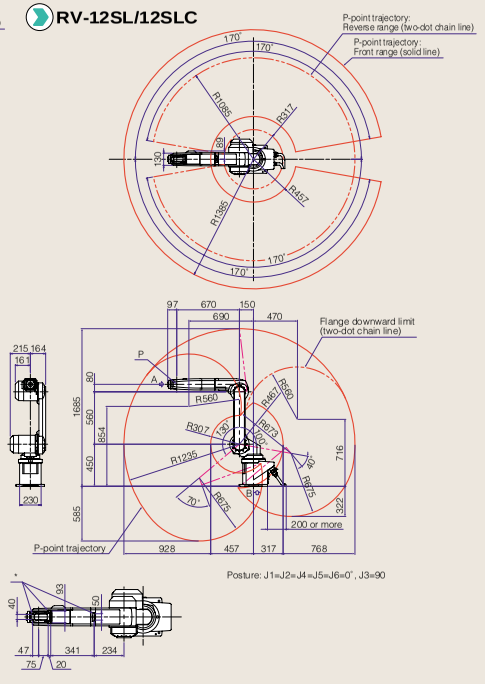
* indicates service screw holes for tooling (M4 x 6 places).

RV-12S/12SC



* indicates service screw holes for tooling (M4×6 places).

RV-12SL/12SLC



* indicates service screw holes for tooling (M4×6 places).

Specification

●Robot Body

Type			Unit	RV-6S/6SC	RV-6SL/6SLC	RV-12S/12SC	RV-12SL/12SLC
Structure				Vertical multiple-joint type			
Degrees of freedom				6			
Drive system				AC servo motor (brakes for all axes)			
Position detection method				Absolute encoder			
Maximum load capacity (rated) *2			kg	6 (5)		12 (10)	
Arm length			mm	280+315	380+425	400+530	560+670
Maximum reach radius			mm	696	902	1086	1385
Operating range	Waist	J1	degrees	340 (±170), can be limited after shipment (in 45° intervals)			
	Shoulder	J2		227 (-92 to +135)		230 (-100 to +130)	
	Elbow	J3		285 (-107 to +166)	295 (-129 to +166)	290 (+160 to -130)	
	Wrist twist	J4		320 (±160)			
	Wrist pitch	J5		240 (±120)			
	Wrist roll	J6		720 (±360)			
Maximum speed	Waist	J1	degrees/s	401	250	276	230
	Shoulder	J2		321	267	230	172
	Elbow	J3		401	267	267	200
	Wrist twist	J4		352			
	Wrist pitch	J5		450		375	
	Wrist roll	J6		660			
Maximum composite speed *3			mm/sec	Approx. 9300	Approx. 8500	Approx. 9600	Approx. 9500
Cycle time *4				Order of 0.4 seconds	Order of 0.6 seconds	Order of 0.7 seconds	Order of 0.7 seconds
Position repeatability			mm	±0.02	±0.02	±0.05	±0.05
Ambient temperature				0 to 40			
Mass			kg	Approx. 58	Approx. 60	Approx. 93	Approx. 98
Tool wiring *5				8 input points/8 output points (No.2 arm)			
Tool pneumatic pipes				Primary: φ6 x 2, Secondary: φ4 x 8		Primary: φ6 x 2, Secondary: φ6 x 8	
Installation posture				Installation on floor, hanging (hanging on wall *6)			
Machine cable				5 m (connector at both ends)		7 m (fixed on the controller side)	
Protection specification				IP65 (J4 to J6) IP54 (J1 to J3)			

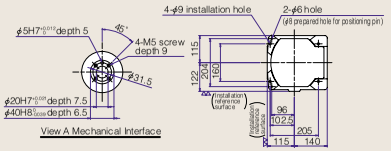
*2: The maximum load capacity is the maximum mass capacity when the wrist flange is pointing downward (±10°).
*3: Value at the hand flange surface when all the axes are combined
*4: Value at a load of 1 kg for RV-6S□and at a load of 5 kg for RV-12S□when the robot reciprocates 25 mm vertically and 300 mm horizontally
*5: To use the tool (hand) output, the (optional) pneumatic hand interface is required.
*6: The movement range of the J1 axis is limited in the special specification that allows the robot to hang on a wall.

●Controller

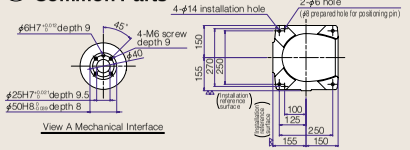
Type		Unit	CR3-535M	CR2B-574
Path control method			PTP control, CP control	
Number of axes controlled			Up to 6 axes simultaneously, and up to 8 axes for additional axis control	
CPU			64bit RISC/DSP	
Robot language			MELFA-BASIC IV	
Position teaching method			Teaching method, MDI method	
Memory capacity	Numbers of teaching points and steps	points	2,500	
		steps	5,000	
	Number of programs	steps	88	
External I/O	General-purpose I/O	points	32/32 (up to 256/256 when using the optional, additional I/O unit)	
	Dedicated I/O	points	Assigned from general-purpose I/O (one point, "STOP," is fixed)	
	Hand I/O	points	8 inputs/0 output (8/8 when the pneumatic hand interface is used)	
	Emergency stop input	points	1 (support 2 contacts)	
	Emergency stop output	points	1 (support 2 contacts)	
	Door switch input	points	1 (support 2 contacts)	
Interface	RS-232C	ports	1 (for connecting a personal computer, vision sensor etc.)	
	RS-422	ports	1 (for connecting a teaching pendant)	
	Slot dedicated to hand	slots	1 (for connecting a pneumatic hand interface)	
	Extension slot	slots	2 (for connecting optional extensions)	3 (for connecting optional extensions)
	SSCNET	ports	1 (for connecting additional axes)	0 (the optional additional axis interface is used for connection)
	Memory expansion slot	slots	1 (for connecting an optional memory cassette)	
	Robot I/O link	channels	1 (for connecting a parallel I/O unit)	
Operating temperature range		°C	0 to 40	
Relative humidity		%RH	45 to 85	
Power supply	Input voltage range	V	3-phase, AC 180 to 253	Single phase, AC 180-253
	Power capacity *7	KVA	3.0 (excluding inrush current)	2.0 (excluding inrush current)
External dimensions		mm	450(W) x 380(D) x 625(H)	460(W) x 400(D) x 200(H)
Mass		kg	Approx. 60	Approx. 20
Structure (protection specification)			Self-contained floor type/closed structure [IP54]	Self-contained floor type/closed structure [IP20]
Grounding *8			100 or less (D-class grounding)	

*7: The power capacity is the rated value at normal operation. Please be aware that the power capacity does not take inrush current applied when the power supply is turned on into consideration. The power capacity should be considered a guideline, and the guaranteed operation depends on the input power supply voltage.
*8: Grounding is conducted at the customer's own risk.

RV-6S/6SL Series Common Parts

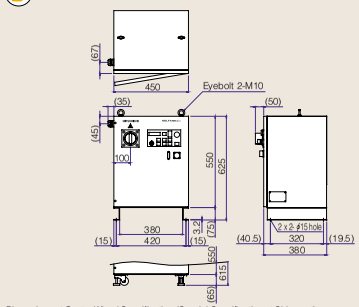


RV-12S/12SL Series Common Parts



Controller External Dimension Diagram

CR3-535M



CR2B-574