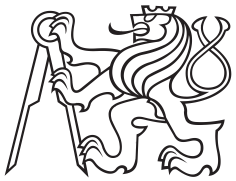
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Propojení senzoru Kinect na průmyslového robota

Lukáš Dastych
Kybernetika a robotika - Systémy a řízení
dastyluk@fel.cvut.cz

Květen 2016
Vedoucí práce: Ing. Pavel Burget, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Lukáš Dastych**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Propojení senzoru Kinect na průmyslového robota**

Pokyny pro vypracování:

1. Připojte senzor Kinect poslední generace a seznamte se s dostupnými funkcemi pro rozpoznávání postavy, případně i s funkcemi pro rozpoznávání obličeje, které Kinect SDK nabízí.
2. Seznamte se se způsobem programování průmyslového robotu Kuka.
3. Navrhněte způsob komunikace s řídicím systémem robota Kuka a navrhněte základní princip ovládání pohybu robota na základě detekce pohybu osoby senzorem.
4. Pohyby robota podle informací ze senzoru naprogramujte a vytvořte demonstrační aplikaci pro robota i pro nadřazené PLC. Navrhněte uživatelské rozhraní pro ovládání aplikace z operátorského panelu.

Seznam odborné literatury:

[1] Microsoft Kinect SDK documentation.
[2] KUKA industrial robots programming manual.

Vedoucí: Ing. Pavel Burget, Ph.D.

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 24. 2. 2016

Poděkování / Prohlášení

Chtěl bych poděkovat především vedoucímu bakalářské práce, Ing. Pavlu Burgetovi, Ph.D., za podnětné rady, ochotu a čas strávený při řešení problémů vzniklých v jejím průběhu. Dále bych chtěl poděkovat celé své rodině za podporu během zpracování této závěrečné práce, ale především během celého studia.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne 26. 5. 2016

.....

Abstrakt / Abstract

Tato práce pojednává o propojení obrazového senzoru Kinect s průmyslovým robotickým manipulátorem tak, aby robot prováděl pohyb předváděný osobou před senzorem Kinect. Cílem této práce bylo navrhnout vhodný způsob přenášení dat ze senzoru Kinect do řídicího systému robota a v návaznosti na to vytvořit robotický program, který bude vykonávat pohyb přes body zadané souřadnicemi pomocí senzoru Kinect. Jako nejvhodnější způsob přenášení dat byla zvolena síť Ethernet. V řídicím systému robota byl využit doplněk KUKA Ethernet KRL pro řízení komunikace a vykonávání pohybu. Výsledkem této práce je plně funkční demonstrační aplikace spojující obrazový senzor Kinect, externí PC, řídicí systém robota, nadřazené PLC a vizualizační operátorský panel.

Klíčová slova: Kinect, Microsoft, obrazový senzor, trajektorie, snímání pohybu ruky, robotický manipulátor, 6-ti osý robot, Kuka, KRL program, Ethernet, PLC, Profinet, vizualizace.

This work deals with the connection of the image sensor Kinect with an industrial robotic manipulator so that the robot performs motion demonstrated by the person in front of the Kinect sensor. The aim of this work was to design an appropriate way of transferring data from the Kinect sensor to the robot control system and building on it to create a robotic program that will perform the movement through the points specified coordinates entered via Kinect. Ethernet network was chosen as the most suitable way of transferring data. A supplement KUKA Ethernet KRL was used in the robot control system for communication control and execution of movement. The result of this work is fully functional demonstration application combining an image sensor Kinect, an external PC, robot controller, the superior PLC and the visualization operator panel.

Keywords: Kinect, Microsoft, image sensor, trajectory, sensing hand movement, robotic manipulator, 6–axis robot, Kuka, KRL program, Ethernet, PLC, Profinet, visualization.

Title translation: Connection of a Kinect sensor and an industrial robot

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Cílem této bakalářské práce je navrhnout a naprogramovat ovládání pohybu robota na základě detekce pohybu osoby senzorem. Jinak řečeno, pohyb lidské paže bude snímán pomocí obrazového senzoru Kinect 2.1 a pomocí Windows aplikace zpracován na souřadnice pro pohyb robotického manipulátoru 2.2. Tyto souřadnice budou odesílány po síti Ethernet do řídicího systému robotu, který bude okamžitě provádět nasnímaný pohyb.

Tato práce by měla ukázat možnost spojení robotického manipulátoru, určeného pro použití v průmyslu, se senzorem, primárně určeným k herní konzoli pro ovládání her a aplikací gesty.

Výsledný systém bude demonstrovat možnost ovládání robotické paže kopírováním pohybu lidské paže v reálném čase. Tímto způsobem je například možno manipulovat s předměty v prostoru. Systém bude možné použít pro účely prezentace katedry Řídicích systémů Fakulty elektrotechnické Českého vysokého učení technického v Praze například při Dnech otevřených dveří.

Kapitola 2
Dílčí prvky

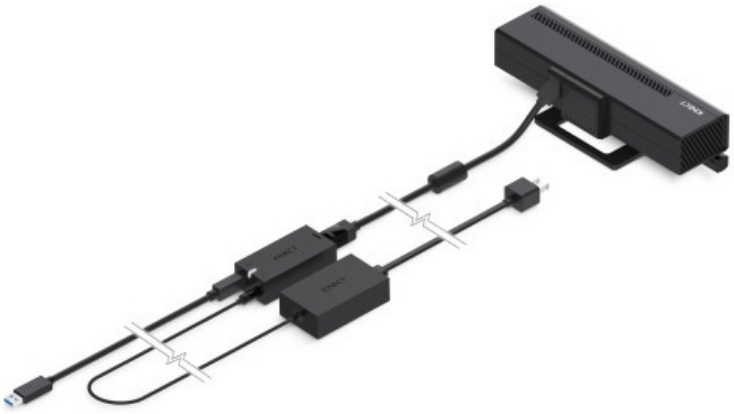
Systém je složen ze 2 základních prvků – obrazového senzoru *Microsoft Kinect* a robotického manipulátoru *KUKA KR 5 ARC*. Senzor *Kinect* je připojen k PC s operačním systémem Windows 10, který je s řídícím systémem robotického manipulátoru propojen pomocí sítě *Ethernet*.

2.1 Obrazový senzor Kinect

Obrazový senzor Microsoft Kinect je vstupní zařízení snímající pohyby osob. Kinect vyvinula společnost Microsoft pro herní konzole. První verze senzoru Kinect byla určena pro herní konzoli Xbox 360 a existovala také speciální verze pro PC s operačním systémem Windows. Kinect 2. generace (*Kinect V2*), zobrazený na obr. 2.1, je určen pro herní konzole Xbox One. Výroba speciální verze pro Windows byla zrušena. Microsoft umožnil použití *Kinectu V2* na počítačích s operačním systémem Windows pomocí *Kinect adaptéru* (viz dále).

Obrázek 2.1. Obrazový senzor Microsoft Kinect pro Xbox One (Zdroj: [1])

Přestože byl *Kinect* primárně určen k herním konzolím pro ovládání her a aplikací pomocí pohybu těla, rozpoznávání gest a hlasových instrukcí, našel své uplatnění v řadě dalších aplikací. Verze *Kinectu* pro Windows je hojně využívána různými kutily, inženýrskými a vědeckými týmy z oblasti robotiky a virtuální reality nebo marketingovými pracovníky. Škála využití je široká, od použití pro strojové 3D vidění robotů, tvorbu prostorových map přes konfigurátory a prezentace zboží na veletrzích a prodejnách, až



Obrázek 2.2. Kinect adaptér pro Windows (Zdroj: [2])

po možnost zkoušet virtuální oblečení. Fantazii využití tohoto finančně dostupného hloubkového obrazového senzoru se meze nekladou.

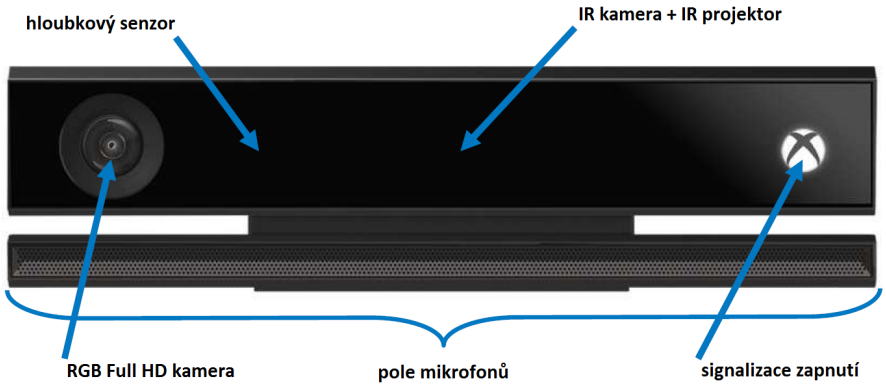
Jak již bylo zmíněno výše, ke *Kinectu V2* (pro Xbox One) neexistuje speciální varianta pro Windows. Tento zdánlivý problém je elegantně vyřešen pomocí *Kinect adaptéru* (viz obr. 2.2) taktéž vyráběný společností Microsoft. Tento adaptér slouží pro připojení senzoru *Kinect* k PC s operačním systémem Windows 8 nebo novějším a zároveň zajišťuje napájení senzoru. Adaptér je nutné k PC připojit pomocí sběrnice USB 3.0 (vysokorychlostní USB sběrnice), která umožňuje přenos dat ze všech součástí senzoru najednou a v reálném čase. Systémové požadavky na PC jsou uvedeny v tab. 2.1.

Procesor	64-bitový (x64) procesor Dual-Core 3,10 GHz nebo rychlejší
Operační systém	Windows 8 (8.1) nebo novější
Paměť	4 GB RAM
Video	Grafická karta podporující DirectX 11
Ostatní	1 volný port USB 3.0

Tabulka 2.1. Systémové požadavky na PC pro připojení senzoru *Kinect V2* pomocí *Kinect adaptéru* (Zdroj: [2])

Senzor *Kinect V2* se skládá ze 4 základních částí – barevné kamery s Full HD rozlišením, hloubkové kamery, infračervené kamery s infračerveným projektorem a pole mikrofonů. Rozmístění částí v senzoru *Kinect V2* je zobrazeno na obr. 2.3.

Hlavní kamera zajišťuje snímání barevného obrazu ve Full HD rozlišení (tedy 1920 x 1080 pixelů) s frekvencí 30 snímků za sekundu při 16-ti bitech na pixel (Bit–Per–Pixel). Tato kamera umožňuje ostření od vzdálenosti 0,8 m až do 4 m, ale doporučená minimální vzdálenost osoby od senzoru *Kinect V2* je 1,37 m. Horizontální pozorovací úhel je 70 stupňů a vertikální pozorovací úhel je 60 stupňů.



Obrázek 2.3. Obrazový senzor Kinect s popisem jeho součástí (Zdroj: [1], upraveno)

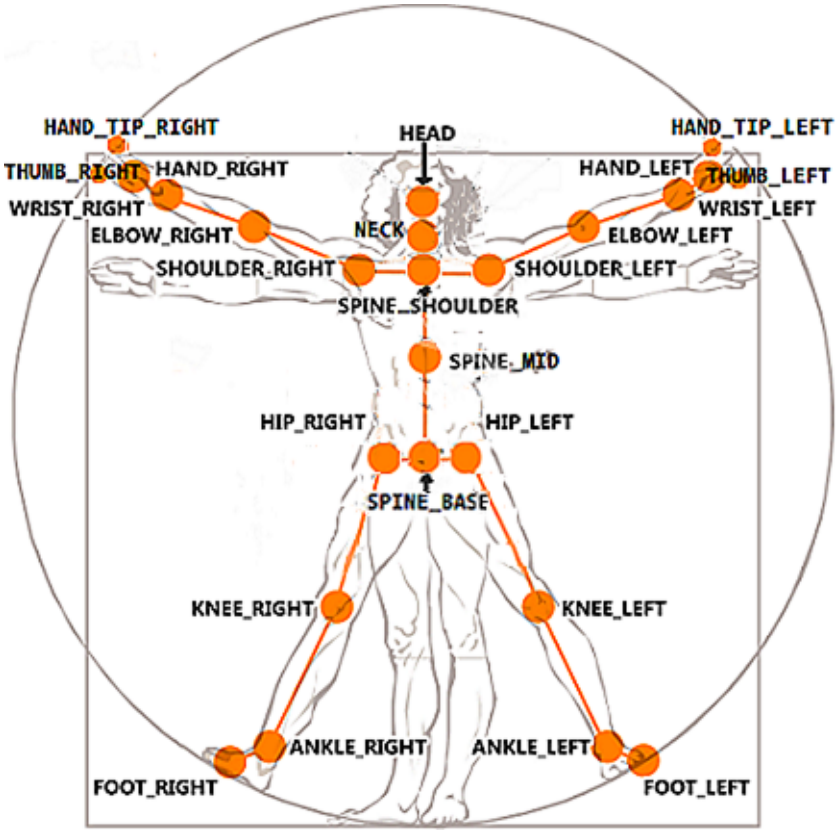
Prostorová kamera pro měření hloubky (třetího rozměru obrazu) má rozlišení 512 x 424 pixelů s frekvencí 30 snímků za sekundu při 16-ti bitech na pixel. Hloubka rozpoznaná touto kamerou je reprezentována 13-ti bitovým číslem. Princip měření hloubky vychází z technologie *Time–of–Flight*. Senzor *Kinect V2* vysílá pomocí infračerveného projektoru pulzy světla. Letící fotony dopadnou na předmět v prostoru, od kterého se odrazí a letí zpět, kde dopadnou na senzor. Vzdálenost předmětu od senzoru se určí z doby letu jednotlivých fotonů.

Infračervená kamera má rozlišení 512 x 424 pixelů s frekvencí 30 snímků za sekundu s 11-ti bitovým dynamickým rozsahem. Tato kamera je schopná snímat obraz téměř stejně dobře při plném osvětlení i ve tmě. To znamená, že je schopná částečně zastoupit barevnou kameru v případě nedostatečného osvětlení. Obraz z infračervené kamery je velmi důležitý pro určování hloubky v prostoru.

Pole mikrofonů se skládá ze 4 rovnoměrně rozmístěných mikrofonů v dolní části *Kinectu V2*. Mikrofony snímají zvuk s frekvencí 48 kHz při 24 bitech. Umístění čtyř mikrofonů v senzoru umožňuje zaměření polohy snímaného zvuku.

Senzor *Kinect V2* připojený přes *Kinect adaptér* k portu USB 3.0 počítače umožňuje datový tok až 2 Gb/s s celkovou latencí 60 ms se zpracováním.

Zásadní pro senzor *Kinect* obecně, je rozpoznávání osob a klíčových bodů lidského těla. Senzor umožňuje snímání pohybů až 6-ti osob, ale v této práci si vystačím pouze se snímáním pohybu paže jedné osoby. Klíčové body lidského těla, které je *Kinect V2* schopen rozeznat jsou graficky znázorněny na obr. 2.4 a souhrnně s označením a popisem jsou uvedeny v tab. 2.2. Senzor je schopen rozpoznat 25 klíčových bodů lidského těla a jejich natočení. Z čehož vyplývá, že známe nejen tvar a polohu lidského těla, ale také například natočení ruky, nohy, hlavy nebo dokonce i to, zda je ruka otevřená nebo zavřená a to podle toho, zda je palec ruky u dlaně nebo je vztyčený. Díky přesnému



Obrázek 2.4. Klíčové body lidského těla, které je Kinect SDK schopen rozpoznat (Zdroj: [3])

sledování těla také senzor dokáže spočítat i aktuální zatížení končetin (jednotlivých kloubů) a jejich zrychlení v prostoru.

Pro snímání pohybu lidského těla je dostupný mód stojící osoby a mód sedící osoby, při kterém se detekují pouze body určující pohyb rukou a hlavy. Přepínání mezi těmito módy probíhá automaticky. Pokud je z dostupného obrazu možné rozpoznat celé lidské tělo (všech 25 klíčových bodů), volí se automaticky mód stojící osoby, jinak se volí mód sedící osoby.

Mezi další rozpoznávací schopnosti senzoru *Kinect V2* patří rozpoznávání gest nebo hlasových příkazů, pomocí kterých lze ovládat různé aplikace. A pokud to hardware, ke kterému je senzor *Kinect V2* připojen, podporuje, tak i možnost zapnutí nebo vypnutí. Senzor je také schopen detekovat osoby podle obličeje, na kterém rozpozná důležité body pro sestavení čárové masky. Podle aktuální mimiky je senzor schopen zjistit například zda osoba mluví (pohybuje rty), směje se, má otevřenou pusu, zavřené pravé nebo levé oko, má brýle, mrká nebo zda sleduje obrazovku před sebou (dívá se proti senzoru). Pomocí detekce přesné fluktuace kůže pomocí infračerveného senzoru je *Kinect V2* dokonce schopný zjistit i to, jaký má osoba před senzorem srdeční pulz.

číslo	označení	popis
0	SpineBase	konec páteře
1	SpineMid	střed páteře
2	Neck	krk
3	Head	hlava
4	ShoulderLeft	levé rameno
5	ElbowLeft	levý loket
6	WristLeft	levé zápěstí
7	HandLeft	levá ruka
8	ShoulderRight	pravé rameno
9	ElbowRight	pravý loket
10	WristRight	pravé zápěstí
11	HandRight	pravá ruka
12	HipLeft	levá kyčel
13	KneeLeft	levé koleno
14	AnkleLeft	levý kotník
15	FootLeft	levé chodidlo
16	HipRight	pravá kyčel
17	KneeRight	pravé koleno
18	AnkleRight	pravý kotník
19	FootRight	pravé chodidlo
20	SpineShoulder	páteř mezi rameny
21	HandTipLeft	prsty levé ruky
22	ThumbLeft	palec levé ruky
23	HandTipRight	prsty pravé ruky
24	ThumbRight	palec pravé ruky

Tabulka 2.2. Označení klíčových bodů lidského těla (Zdroj: [3])

Text obsažený v této kapitole vznikl za pomoci informačních zdrojů [1], [2], [3], [4], [5], [6], [7] a [8].

2.2 Průmyslový robot Kuka

Robotem, který bude vykonávat pohyby zadávané člověkem prostřednictvím senzoru *Kinect V2*, je průmyslový robot *KUKA KR 5 ARC* (dále jen robot). Robot *KUKA KR 5 ARC* je 6-ti osý průmyslový robot s nízkou mezní zátěží (zobrazen na obr. 2.5). Je určený zejména pro obloukové svařování, ale díky kompaktním rozměrům, malé hmotnosti a vysoké dynamice je vhodný pro různé způsoby využití. Základní parametry robota jsou uvedeny v tab. 2.3.

Obrázek 2.5. Průmyslový robot *KUKA KR 5 ARC* (Zdroj: [9])

Demonstrační robot využitý pro tuto bakalářskou práci je připevněný k podlaze, ale pokud by to pro daný úkon bylo výhodné, výrobce umožňuje i připevnění robota ke stropu, což může například ušetřit pracovní prostor. Tento robot má jako nástroj použity elektricky ovládané kleště, pomocí kterých je možné uchopit předmět s maximální hmotností 5 kg, což odpovídá mezní zátěži robota.

Počet os	6
Mezní zátěž	5 kg
Hmotnost	127 kg
Maximální dosah	1412 mm
Pracovní prostor	8,4 m^3
Přesnost opakování	$\pm$ 0,04 mm
Řídicí systém	KR C4
Montážní polohy	Podlaha, strop

Tabulka 2.3. Základní parametry robota *KUKA KR 5 ARC* (Zdroj: [9] a [10])

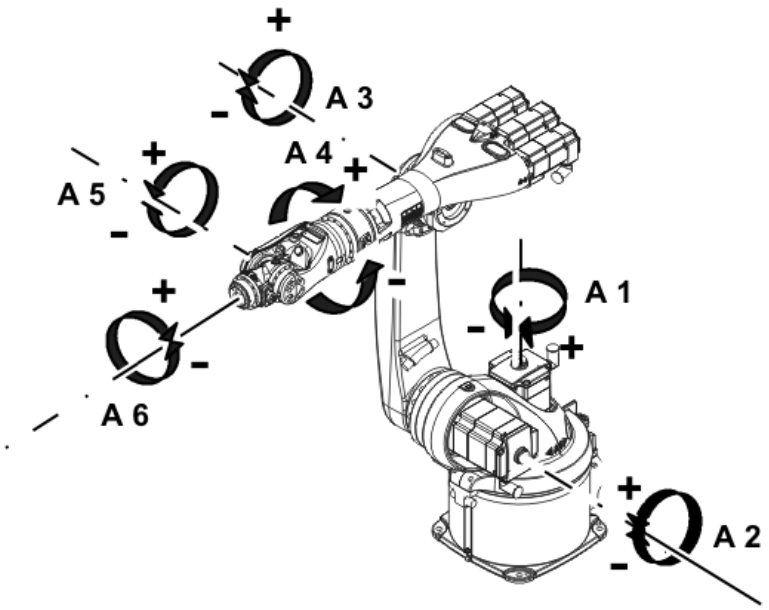
Aby robot nemohl narazit do jakéhokoli předmětu v jeho okolí, je vybaven mechanickými dorazy. Ty jsou použity jako základní ochranné prvky proti pohybu robota do míst, kde by mohl narazit. Těmito dorazy jsou vybaveny osy 1, 2 a 3. Navíc je všech 6 os opatřeno tzv. softwarovými dorazy. Jedná se o tabulku, ve které jsou uvedeny mezní úhly natočení jednotlivých os. Ty jsou menší než úhly natočení, které jsou definovány mechanickými dorazy. Pomocí těchto softwarových omezení (limitů) je možné jednoduše omezit pracovní prostor robota podle jeho umístění. Nejvyšší možné hodnoty natočení jednotlivých os, spolu s maximální rychlostí jednotlivých os, jsou uvedeny v tab. 2.4. Směr otáčení jednotlivých os robota je znázorněn na obr. 2.6.

Osa	Rozsah pohybu (softwarově omezen)	Maximální rychlost s mezní zátěží
1	±155°	154°/s
2	+65° až -180°	154°/s
3	+158° až -15°	228°/s
4	±350°	343°/s
5	±130°	384°/s
6	±350°	721°/s

Tabulka 2.4. Rozsah a rychlost pohybu jednotlivých os robota *KR 5 ARC* (Zdroj: [10])

Kompletní sestava průmyslového robota je znázorněna na obr. 2.7. Základními prvky sestavy je robotický manipulátor *KUKA KR 5 ARC* (1), propojovací kabely pro výkonové a řídicí účely (2), řídicí systém robota *KR C4* (3) a dotykový ovládací panel *KUKA smartPAD* (4).

Řídicí systém *KUKA KR C4* (KUKA Robot Controller – KRC) obsahuje řídicí PC a výkonové moduly pro ovládání servomotorů. Řídicí systém zajišťuje řízení robota podle zadaných programů a komunikaci mezi robotem a uživatelem (člověkem), externím PC pro programování, nadřazeným PLC (Programmable Logic Controller) a dalšími interagujícími zařízeními, které jsou součástí pracoviště robota (bezpečnostní prvky, senzory, další roboti).



Obrázek 2.6. Směr otáčení os robota (Zdroj: [10])

Obrázek 2.7. Kompletní sestava průmyslového robota KUKA (Zdroj: [11])

Software řídicího systému se nazývá KUKA System Software (KSS) a používaná verze je 8.2.19. S touto verzí softwaru řídicího systému je možné komunikovat přes externí PC pomocí softwaru *Work Visual 3.0*. K řídicímu systému je připojen dotykový ovládací panel *KUKA smartPAD* (někdy také nazýván KUKA Control Panel – KCP), pomocí kterého je možné plnohodnotně robota ovládat, programovat, nastavovat i testovat.

Pracoviště robota je dále vybaveno bezpečnostní optickou závorou *SICK C4000 Standard*, bezpečnostním laserovým skenerem *SICK S3000*, nadřazeným řídicím PLC *Simatic S7–300*, dotykovým vizualizačním panelem, měřícím zařízením spotřeby *WAGO 750-494* a externím PC s operačními systémy Microsoft Windows 7 a Microsoft Windows 10.

V řídicím systému robota je nainstalován různý přídavný software. Pro tuto práci je nejdůležitějším doplňkem „Ethernet KRL V2.1.2“. Tento software umožňuje vytvořit necyklické ethernetové spojení mezi řídicím systémem robota a externím systémem, v tomto případě externím počítačem. Přenos dat probíhá ve formě XML řetězců pomocí ethernetového protokolu TCP/IP. Žádná data se nemohou ztratit, protože se všechna přijatá data ukládají do vyrovnávací paměti. Tímto je splněn základní požadavek na použití sítě Ethernet pro průmyslové použití. Detailnější popis je uveden v kapitole 5.

Text obsažený v této kapitole vznikl za pomoci informačních zdrojů [9], [10], [11], [12], [13], a [14].

Kapitola 3
Programování senzoru Kinect V2

K senzoru *Kinect V2* je od výrobce (Microsoft) volně ke stažení [15] vývojářský „kit“, pomocí kterého lze snadno přistupovat ke všem dostupným snímacím a vyhodnocovacím vlastnostem senzoru *Kinect V2*. Jedná se o *Kinect for Windows SDK 2.0*. Ten nám umožní vytvářet pohodlně aplikace využívající tento senzor pomocí připravených tříd a funkcí. Tento vývojářský „kit“ umožňuje programování v jazycích C#, C++, VB.Net nebo JavaScript. Já jsem si pro další práci zvolil programovací jazyk C#.

V následujících kapitolách se budu věnovat získávání dat ze senzoru *Kinect V2*. Převážně se bude jednat o informace o bodech sledovaného lidského těla. Vývojářský kit nám dává možnost přístupu k široké škále možností, kterými tento senzor disponuje, ale protože například získávání rozpoznaných bodů obličeje je složitější a především následná realizace demonstrační aplikace na robota by byla velmi složitá, bude v této bakalářské práci využíváno pouze rozpoznávání bodů lidského těla.

3.1 Softwarové požadavky

Jak již bylo zmíněno v kap. 2.1, pro práci se senzorem *Kinect V2* je zapotřebí počítač s nainstalovaným operačním systémem Microsoft Windows 8 a nebo novější (jako např. v tomto případě Windows 10). Pro možnost přistoupení k senzoru *Kinect V2* a vytváření aplikací s jeho využitím, je zapotřebí mít na tomto počítači nainstalován vývojářský „kit“ *Kinect for Windows SDK 2.0*. Dále je potřeba pro správnou funkčnost a podporu tohoto „kitu“ mít nainstalován *Microsoft .NET Framework 4* a pro vlastní tvorbu aplikací se senzorem *Kinect V2* je doporučeno používat *Microsoft Visual Studio 2012*, nebo novější (jako např. v tomto případě Visual Studio 2015). Přehledné shrnutí softwarových požadavků na PC, ke kterému bude připojen senzor *Kinect V2*, a na kterém zároveň bude probíhat vývoj aplikace využívající tento senzor, je uvedeno v tab. 3.1.

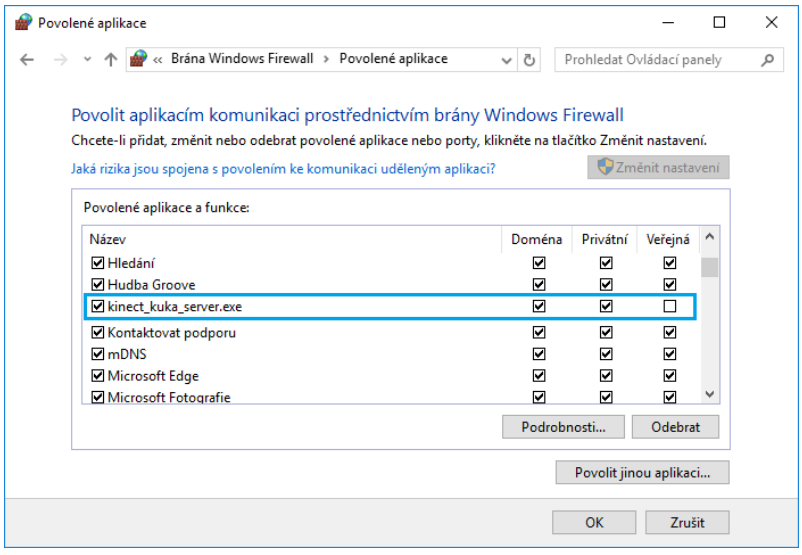
Microsoft Windows 8 nebo novější
Microsoft Visual Studio 2012 nebo novější
Microsoft .NET Framework 4 nebo vyšší
Kinect for Windows SDK 2.0

Tabulka 3.1. Softwarové požadavky na PC pro tvorbu aplikací pomocí *Windows SDK* (Zdroj: [15])

3.2 Vytvoření projektu ve Visual Studiu

Pro tvorbu aplikace s využitím senzoru *Kinect V2* je potřeba nejdříve vytvořit projekt v prostředí Visual Studio. Tento projekt vytvoříme po spuštění aplikace Visual Studio kliknutím na „File“ v horním panelu a poté zvolit „New Project“. V okně, které se otevře, zvolíme „Windows Console Application“ v programovacím jazyce C#, zapíšeme název naší budoucí aplikace a zvolíme umístění na disku. Potvrzením tohoto okna se vytvoří nový projekt se základní strukturou kódu v jazyce C#. Dále je potřeba v pravé části prostředí v záložce „Solution Explorer“ kliknout pravým tlačítkem myši na „References“ a zvolit „Add Reference...“. V okně „Reference Manager“, které se otevře, najdeme a zvolíme „Microsoft.Kinect“ a potvrdíme OK. Tím máme projekt ve Visual Studiu připraven pro tvorbu aplikace se senzorem *Kinect V2*.

Pokud plánujeme, že bude aplikace komunikovat po síti Ethernet (jako např. demonstrační aplikace této bakalářské práce), je v tuto chvíli potřeba aplikaci sestavit (zvolit „Build“ a „Build Solution“) a poté spustit (zvolit „Debug“ a „Start Without Debugging“). Po spuštění můžete aplikaci ukončit. Dále je potřeba být přihlášen jako „Administrátor“ a v nastavení Windows přejít k nastavení *Brány Windows Firewall*. Kde je potřeba tuto novou aplikaci povolit podle obr. 3.1. Tím je vše připraveno k tvorbě a testování nové aplikace.



Obrázek 3.1. Povolení nové aplikace v bráně Windows Firewall

3.3 Navázání spojení a aktivace senzoru Kinect V2

Pro získávání dat ze senzoru *Kinect V2* je zapotřebí si deklarovat 2 objekty. Prvním z nich je `Kinect Sensor`, který reprezentuje přímý přístup k tomuto senzoru a druhý je `Kinect stream reader`, pomocí kterého bude zajištěno čtení potřebného streamu (`FrameSourceTypes.Body`). Dále se provede inicializace senzoru a ověření, zda byl senzor úspěšně nalezen. Pokud by nebyl žádný senzor nalezen, což může být např. z důvodu nepřipojení senzoru k portu USB 3.0, bude program ukončen. Jinak je provedeno navázání spojení s nalezeným senzorem. Podobně jako v předchozím případě je tato operace testována a při negativním výsledku je program ukončen. Nakonec je potřeba nastavit jaké streamy se mají číst a zahájit čtení daných streamů pomocí dále definované metody `Reader_MultiSourceFrameArrived`. Pokud již nebudeme dál senzor *Kinect V2* nadále potřebovat, nebo před ukončením aplikace, je potřeba senzor deaktivovat a ukončit spojení. Všechny tyto operace jsou pro ilustraci uvedeny v následujícím výpisu kódu.

```
// Deklarace objektu Kinect sensor a Kinect stream reader
KinectSensor _sensor;
MultiSourceFrameReader _reader;

// Detekovani a inicializace prvniho pripojeneho senzoru Kinect V2
_sensor = KinectSensor.GetDefault();

// Pokud senzor nebyl nalezen, ukonceni programu
if (_sensor.IsAvailable == false) return;

// Otevreni spojeni se senzorem Kinect V2
_sensor.Open();

// Pokud nebylo spojeni se senzorem uspesne, ukonceni programu
if (_sensor.IsOpen == false) return;

// Nastaveni streamu, ktery se ma cist a zahajeni cteni stremu
//  pomoci metody Reader_MultiSourceFrameArrived.
_reader = _sensor.OpenMultiSourceFrameReader(FrameSourceTypes.Body);
_reader.MultiSourceFrameArrived += Reader_MultiSourceFrameArrived;

// Ukonceni pouzivani senzoru Kinect V2 na konci programu
_sensor.Close();
```

Text této kapitoly a zdrojový kód vychází ze zdroje [16].

3.4 Získání informace o bodech lidského těla ze senzoru Kinect V2

Informace o detekovaných osobách a bodech lidského těla jsou získávány ze senzoru *Kinect V2* pomocí volané metody `Reader_MultiSourceFrameArrived`. Ta je automaticky zavolána pokaždé, když je k dispozici nový snímek (frame), což je 30 krát za sekundu. Při každém zavolání této metody musíme nejdříve získat referenci (odkaz) na aktuální frame. Může se stát, že některý frame bude chybný, je proto nutné tuto referenci testovat na prázdnost (`frame != null`). Poté inicializujeme seznam detekovaných osob a následně zavoláme metodu `GetAndRefreshBodyData()` pro načtení informací jednotlivých detekovaných těl do seznamu osob. Dále již procházíme ve smyčce detekované osoby. Pokud je osoba rozpoznána a sledována senzorem, můžeme ve smyčce procházet jednotlivé detekované body lidského těla. Zda je bod lidského těla detekovaný, nebo ne, testujeme pomocí stavu sledování daného bodu (`TrackingState`) na hodnotu `Tracked`. V tuto chvíli již můžeme přistupovat k jednotlivým bodům lidského těla. Tuto popsanou část metody `Reader_MultiSourceFrameArrived` shrnuje následující výpis kódu.

```
IList⟨Body⟩ osoby; // Seznam detekovanych osob

// Ziskani reference na aktualni frame
using (var frame = reference.BodyFrameReference.AcquireFrame())
{
  if (frame != null) // Testovani na prazdnost framu
  {
    // Inicializace seznamu detekovanych osob
    osoby = new Body[frame.BodyFrameSource.BodyCount];
    // Nacteni informaci detekovanych tel do seznamu osob
    frame.GetAndRefreshBodyData(osoby);
    // Prochazeni jednotlivych detekovanych osob ve smycce
    foreach (var osoba in osoby)
    {
      // Testovani, zda je osoba rozpoznana a sledovana
      if (osoba.IsTracked)
      {
        // Prochazeni jednotlivych bodu lidskeho tela
        foreach (Joint joint in osoba.Joints.Values)
        {
          // Testovani, zda je bod lidskeho tela detekovany
          if (joint.TrackingState == TrackingState.Tracked)
          {
            // Pristup k parametrum jednotlivych bodu lidskeho tela
          }
        }
      }
    }
  }
}
```

Abychom ve smyčce určili, zda právě procházený bod lidského těla je ten, ke kterému chceme přistoupit, můžeme použít testování aktuálního procházeného bodu ve smyčce na rovnost s určitým bodem lidského těla. Například takto zjistíme, zda je aktuálně procházený bod lidského těla bod „SpineShoulder“.

```
joint.Equals(osoba.Joints[JointType.SpineShoulder])
```

Tím jsme zjistili, že právě přistupujeme k bodu, který požadujeme. Informace o jednotlivých souřadnicích bodu v prostoru, získaná ze senzoru *Kinect V2*, je hodnota v metrech. Získat jednotlivé souřadnice bodu v prostoru je velice jednoduché. Pomocí následujícího příkladu získáme například hodnotu souřadnice X bodu levé ruky.

```
osoba.Joints[JointType.HandLeft].Position.X
```

Při programování senzoru *Kinect V2* s následným začleněním do serverové aplikace pro demonstrační aplikaci této bakalářské práce jsem vycházel z informací a kódu uvedeného ve zdroji [16] a z referenční dokumentace k senzoru *Kinect V2* viz [17].

3.5 Určení souřadnic pro následné vykonání robotem

Souřadnice bodu jsou brány v kartézské souřadné soustavě, která má počátek v místě hloubkové kamery na senzoru *Kinect V2*, jak je ukázáno na obr. 3.2. Z tohoto obrázku vyplývá, že hodnota souřadnice X roste doprava, hodnota souřadnice Y roste nahoru a hodnota souřadnice Z roste se vzdáleností kolmo od senzoru. Tento slovní popis je brán z pohledu člověka, který se dívá na senzor Kinect, to znamená, že např. pokud pohnu pravou rukou směrem doprava (směrem od těla) hodnota souřadnice X se zvětší [18].

Obrázek 3.2. Kartézský souřadný systém senzoru Kinect V2 (Zdroj: [18])

Pro tuto práci jsem se rozhodl používat relativní souřadnice pravé, resp. levé, ruky vůči bodu `SpineShoulder` (střed mezi rameny). Výhoda tohoto přístupu spočívá

v jednoduchosti navázání na souřadný systém robota a zejména v omezení pohybu. Myšlenka vychází ze stacionárního umístění robota. To znamená, že když se základna robota nemůže pohnout, pak ani pohyb celé postavy člověka před senzorem nemůže mít vliv na zadání souřadnic pro robota. V tomto případě tedy výsledné souřadnice pro vykonání pohybu robotem nezávisí na postavení osoby před senzorem (až na hardwarové rozpoznávací schopnosti senzoru – vzdálenost osoby od senzoru), ale pouze na vzdálenosti dlaně ruky od středu mezi rameny.

Pro omezení šumu senzoru *Kinect V2* v této práci využívám průměrování hodnot souřadnic. Průměruji souřadnice aktuálního bodu se souřadnicemi třech předchozích bodů. Průměruji tedy souřadnice čtyř bodů. Tento počet byl zjištěn experimentálně jako nejlepší. Po zprůměrování souřadnice převedu z metrů na milimetry a přenásobím konstantou menší než 1 pro zajištění optimálního využití celého pracovního prostoru robota. Přičtením konstanty ke každé souřadnici bodu nakonec posunu bod do základního bodu pracovního prostoru robota, který je experimentálně zvolen tak, aby robot obsáhl s relativními příspěvky od senzoru co největší oblast. Nastavení jsem provedl tak, aby při maximálních vzdálenostech dlaně od středu mezi rameny, se robot dostal ke krajům svého pracovního prostoru, ale tak, aby z něj ve většině případů nevyjel.

Kapitola 4
Programování průmyslového robotu KUKA

Průmyslový robot *KUKA KR 5 ARC* s řídicím systémem *KR C4* je možné programovat pomocí dotykového ovládacího panelu *KUKA smartPAD* nebo v softwaru *Work Visual 3.0* nainstalovaným na externím PC.

Pro ovládání robota, vytváření a modifikace pohybových programů, diagnostiku a nastavování robota je dostupných několik uživatelských skupin. Základní uživatelské skupiny jsou uvedeny v tab. 4.1 a jsou seřazeny podle stupně oprávnění od nejnižšího po nejvyšší (seznam není kompletní – závisí na aktuální verzi KSS, popř. i vlastním nastavení). Využívání uživatelských skupin je dobré zejména pro oddělení kompetencí jednotlivých osob. Například běžný uživatel robota (*„User“*) může spouštět programy a vytvářet (resp. upravovat) programy pouze na nejzákladnější úrovni. Pokud uživatel potřebuje upravovat nebo vytvářet programy na vyšší úrovni programování, nebo zasahovat do nastavení robota, je nutné, aby se přihlásil jako *„Expert“* nebo *„Administrator“*. U těchto uživatelských skupin je vždy vyžadováno heslo pro ověření způsobilosti uživatele k provádění daných úkonů.

Název uživatelské skupiny	Popis	Vyžadováno heslo
Operator	standardní uživatel	ne
User	pokud není nastaveno jinak, stejné jako Operator	ne
Expert	určeno pro programátory	ano
Administrator	stejné jako Expert, navíc umožňuje integrovat zásuvné moduly	ano

Tabulka 4.1. Základní uživatelské skupiny dostupné pro ovládání robota (Zdroj: [13])

Programovací jazyk, ve kterém jsou napsány všechny programy pro vykonávání pohybu robotického manipulátoru, se nazývá *KUKA Robot Language – KRL*. Základní dělení programovacího přístupu je podle toho, v jaké uživatelské skupině programy vytvářím (upravuji).

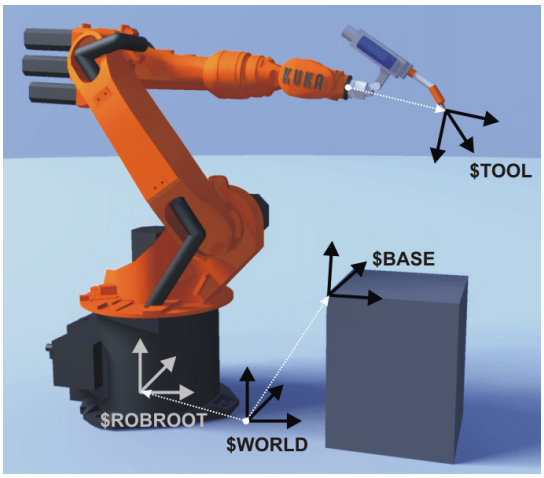
- v uživatelské skupině *„User“* – *inline formuláře* 4.3
- v uživatelské skupině *„Expert“* – *KRL syntaxe* 4.5

V následujících podkapitolách bude postupně rozebráno v jakých souřadnicových systémech (4.1) lze robotem pohybovat a jakými druhy pohybů (4.2). Dále pak programování robota v uživatelské skupině „User“ (4.3) a „Expert“ (4.5). Nakonec bude zmíněna struktura programů (4.4), mapování signálů (4.6), přerušení (4.7) a singularity (4.8).

4.1 Souřadnicové systémy

Řídicí systém robota umožňuje využívat pro programování pohybů robota 4 různé kartézské souřadnicové systémy. Jejich umístění je graficky znázorněno na obr. 4.1.

- WORLD
- ROBROOT
- BASE
- TOOL



Obrázek 4.1. Grafické znázornění umístění souřadnicových systémů robota (Zdroj: [13])

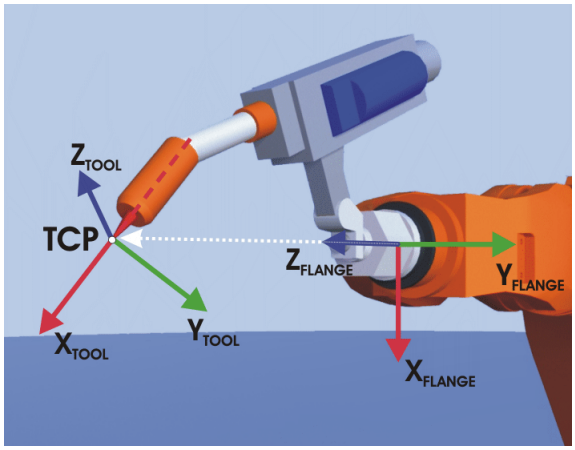
Souřadnicový systém WORLD je pevně definovaný a je to počáteční souřadnicový systém pro souřadnicové systémy ROBROOT a BASE. Ve výchozím stavu leží souřadnicový systém WORLD v noze robota.

Souřadnicový systém ROBROOT vždy leží v noze robota a definuje pozici robota ve vztahu k souřadnicovému systému WORLD. Ve výchozím stavu je souřadnicový systém ROBROOT shodný se souřadnicovým systémem WORLD. Pomocí systémové proměnné $ROBROOT je možné definovat pozici robota ve vztahu k souřadnicovému systému WORLD.

Souřadnicový systém BASE popisuje pozici obráběného dílu. Vztahuje se na souřadnicový systém WORLD. Ve výchozím stavu je souřadnicový systém BASE shodný

se souřadnicovým systémem WORLD. O jeho posunutí do obráběného dílu se musí postarat uživatel pomocí systémové proměnné $BASE.

Souřadnicový systém TOOL leží v pracovním bodu nástroje. Ve výchozím stavu leží počátek souřadnicového systému TOOL ve středu příruby (potom je nazýván souřadnicovým systémem FLANGE). O jeho posunutí do pracovního bodu nástroje se musí postarat uživatel pomocí systémové proměnné $TOOL. Tím uživatel přidělí nástroji, který je připevněn na montážní přírubě, souřadnicový systém TOOL. Vztažení souřadnicového systému TOOL na souřadnicový systém FLANGE je graficky znázorněno na obr. 4.2. Počátek souřadnicového systému TOOL se zpravidla umísťuje do pracovního bodu nástroje a nazýváme jej TCP (Tool Center Point).

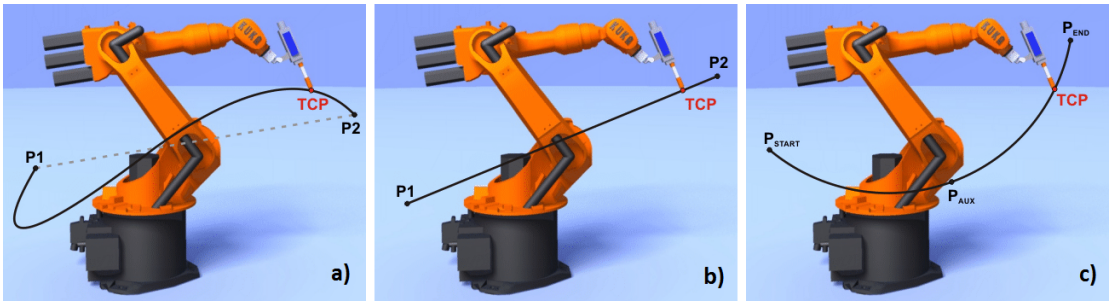


Obrázek 4.2. Vztažení souřadnicového systému TOOL na souřadnicový systém FLANGE (Zdroj: [13])

4.2 Základní druhy pohybů

Existují 3 základní druhy pohybů, a to PTP (Point To Point), LIN (Linear) a CIRC (Circular). K těmto základním pohybům můžeme zařadit ještě jeden pohyb, který není samostatným pohybem, ale již pohybovým blokem skládajícím se z dílčích pohybů, a to pohyb typu SPLINE.

Pro zadání trajektorie, kterou má bod TCP vykonat, je potřeba zadat pouze 1 nebo 2 body v závislosti na druhu pohybu (u pohybu SPLINE libovolné množství bodů). Provádění prvních 3 základních druhů pohybů je graficky zobrazeno na obr. 4.3.



Obrázek 4.3. Provádění pohybů PTP, LIN a CIRC (Zdroj: [13])

Pro pohyb PTP platí, že se bod TCP pohybuje z bodu „P1“ po nejrychlejší dráze k cílovému bodu „P2“ (viz obr. 4.3 a)). Kde bod „P1“ je koncový bod předchozí pohybové instrukce, tzn. bod, ve kterém se TCP nachází před vykonáním aktuálního pohybu (pohybové instrukce). Z toho vyplývá, že pro pohyb PTP je zapotřebí zadat pouze koncový bod „P2“. Dráha, kterou bod TCP při pohybu PTP projíždí je nejrychlejší, ale ta zpravidla není nejkratší. Protože se osy robota pohybují rotačně, je provádění obloukových drah rychlejší než drah přímých. Přesný průběh pohybu není předvídatelný, protože vždy vychází z aktuální kinematiky robota tak, aby provedení pohybu bylo nejrychlejší. Můžeme tedy usuzovat, že opakovaný pohyb PTP ze stejným způsobem najetého počátečního bodu do stejného koncového bodu bude mít stejnou dráhu.

Pro pohyb LIN platí, že se bod TCP pohybuje definovanou rychlostí podél přímky k cílovému bodu (viz obr. 4.3 b)). Kde stejně jako u pohybu PTP je nutné zadat pouze koncový bod „P2“. Lineární pohyb není pro robota přirozený. Může tedy být pro robota velice obtížně realizovatelný. V mnoha případech je dokonce, bez dodatečných omezení, nerealizovatelný. Doporučuje se používat pohyb LIN pouze tam, kde je to bezpodmínečně nutné.

Pro pohyb CIRC platí, že se bod TCP pohybuje definovanou rychlostí podél kruhové dráhy k cílovému bodu (viz obr. 4.3 c)). Kruhová dráha je definována startovním, pomocným a cílovým bodem. Kde podobně jako u pohybů PTP a LIN je nutné zadat koncový bod „P_{END}“ a navíc pomocný bod „P_{AUX}“ pro určení kruhové dráhy.

U všech třech základních pohybů je možné zvolit variantu bez aproximace a s aproximací. Bez aproximace znamená, že se do koncového bodu pohybu najede přesně. S aproximací znamená, že se do koncového bodu pohybu najede nepřesně. S jakou nepřesností se dosáhne koncového bodu pohybu je možné nastavit u každého pohybu při jeho programování. Pokud nepotřebujeme dosáhnout koncového bodu pohybu přesně, je výhodné využít možnosti aproximace, pohyb se tím stává plynulejší.

Aproximace u pohybu PTP znamená, že bod TCP opustí dráhu, na které by přesně najel na cílový bod, a pohybuje se po rychlejší dráze, přičemž aproximovaný koncový

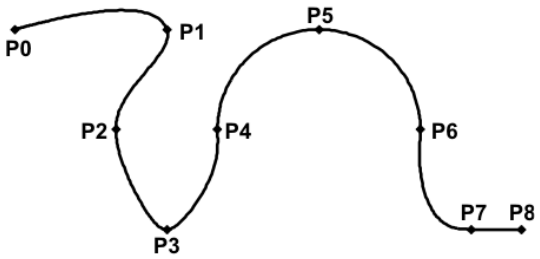
Obrázek 4.4. Pohyby PTP, LIN a CIRC s aproximací (Zdroj: [13])

bod mine maximálně o nastavenou hodnotu aproximace (viz obr. 4.4 a)). Průběh dráhy při aproximovaném PTP pohybu není předvídatelný, stejně tak jako na které straně aproximovaného bodu bude dráha probíhat.

Aproximace u pohybu LIN znamená, že bod TCP opustí dráhu, na které by přesně najel na cílový bod, a pohybuje se po kratší dráze, přičemž od původní dráhy se odchýlí nejdříve při nastavené vzdálenosti aproximace od aproximovaného koncového bodu (viz obr. 4.4 b)). Průběh dráhy v oblasti aproximace není kruhový oblouk. Pomocí aproximace u pohybu LIN je možné z nerealizovatelného pohybu vytvořit pohyb LIN, který lze provést.

Aproximace u pohybu CIRC znamená, že bod TCP opustí dráhu, na které by přesně najel na cílový bod, a pohybuje se po kratší dráze, přičemž od původní dráhy se odchýlí nejdříve při nastavené vzdálenosti aproximace od aproximovaného koncového bodu (viz obr. 4.4 c)). Průběh dráhy v oblasti aproximace není kruhový oblouk, ale pomocným bodem se vždy projede přesně.

Grafické znázornění pohybů PTP, LIN a CIRC s aproximací je na obr. 4.4 a), b), a c) v tomto pořadí. Aproximovaný bod je zakreslen červenou barvou.



Obrázek 4.5. Oblouková dráha pomocí bloku SPLINE (Zdroj: [13])

Poslední druh, který by se ještě dal zařadit mezi základní pohyby, je pohyb SPLINE (viz obr. 4.5). „Je to kartézský druh pohybu, který je obzvláště vhodný pro komplexní obloukové dráhy. Takové dráhy lze v zásadě vytvořit i pomocí aproximovaných pohybů LIN a CIRC, SPLINE však má řadu výhod.“ Na rozdíl od aproximovaných pohybů LIN a CIRC body, kterými je dráha definována, na této dráze leží a lze ji vytvořit

jednoduše. Dále se také udržuje naprogramovaná rychlost (až na pár případů, kde dochází k redukci rychlosti). Kruhy malých poloměrů a dráhy s body definovanými blízko sebe jsou projížděny s vysokou přesností. „Pohyb SPLINE se může skládat z několika jednotlivých pohybů, segmentů SPLINE. Jednotlivé segmenty jsou spojeny do celkového pohybu v tzv. bloku SPLINE. Blok SPLINE je řídicím systémem robota plánován a proveden jako 1 pohybový blok.“ [11]

4.3 Programování v uživatelské skupině „User“ pomocí inline formulářů

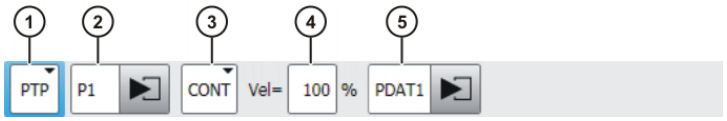
Pokud je uživatel přihlášen do uživatelské skupiny *User* popř. *Operator* má možnost programy pro vykonávání pohybu robotického manipulátoru vytvářet a upravovat pouze pomocí tzv. *inline formulářů*. Programování pomocí *inline formulářů* je samozřejmě dostupné i pro všechny ostatní uživatelské skupiny.

Pomocí *inline formulářů* lze snadno naprogramovat zejména instrukce pohybu (PTP, LIN, CIRC), které budou vysvětleny později. Dále lze pomocí *inline formulářů* naprogramovat instrukce SPLINE pohybu, instrukce vstupu a výstupu (IN, OUT, PULSE), instrukce čekání (WAIT – určitý čas, WAITFOR – čekání na definovaný signál) a několik dalších.

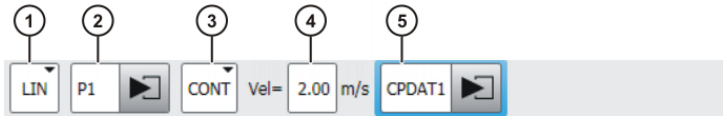
Inline formuláře jsou jednoduché graficky zpracované tabulky, které mají pouze jeden řádek. V tomto řádku je po uživateli požadováno pouze vyplnění nejdůležitějších údajů pro danou instrukci. Při zvolení *inline formuláře* určité instrukce jsou všechna jeho pole předvyplněna a uživatel je může modifikovat nebo formulář pouze potvrdit. Po potvrzení *inline formuláře* dojde k vygenerování bloku zdrojového kódu v KRL, který je ale před uživatelem skryt pomocí tzv „Foldu“. Informace obsažené ve „Foldech“ jsou před uživatelem skryté, a zobrazí se mu pouze hlavička, ve které jsou zobrazeny nejdůležitější informace o instrukci. Tato hlavička se zobrazí pouze na jeden řádek, díky čemuž se programy stávají velice přehlednými. Obsah „Foldů“ je možné zobrazit pouze na expertní úrovni (uživatel přihlášený jako „Expert“ nebo „Administrator“).

Programování základních pohybových instrukcí pomocí *inline formulářů* je znázorněno na obr. 4.6, 4.7 a 4.8. Kde číslem 1 je označen rolovací seznam s výběrem typu pohybu (PTP, LIN, CIRC). Číslem 2 je označena buňka s názvem koncového bodu pro pohyby PTP a LIN. U pohybu CIRC je číslem 2 označena buňka s názvem pomocného bodu a číslem 3 je označena buňka s názvem koncového bodu. Číslem 3 (resp. 4 u pohybu CIRC) je označen rolovací seznam s výběrem aproximace (CONT) nebo bez aproximace (prázdné pole). Číslem 4 (resp. 5 u pohybu CIRC) je označena

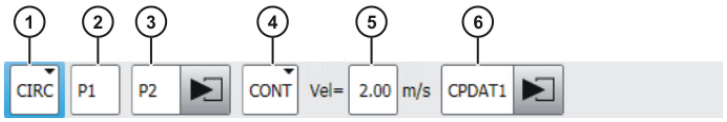
buňka pro zadání požadované rychlosti pohybu. Pro pohyb PTP se zadává v rozsahu 1 % až 100 % a pro pohyb LIN a CIRC se zadává v rozsahu 0,001 m/s až 2 m/s. Číslem 5 (resp. 6 u pohybu CIRC) je označena buňka s názvem pro blok dat pohybu.



Obrázek 4.6. Inline formulář PTP pohybu (Zdroj: [13])

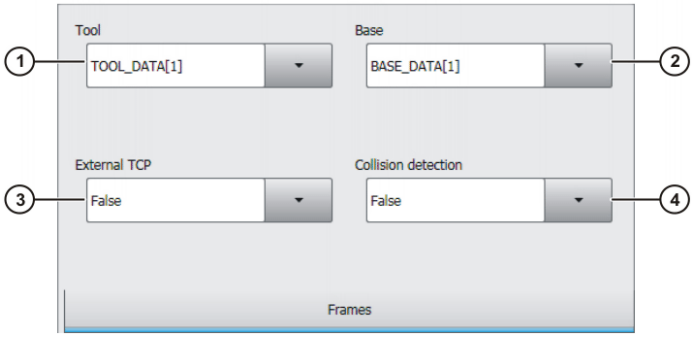


Obrázek 4.7. Inline formulář LIN pohybu (Zdroj: [13])

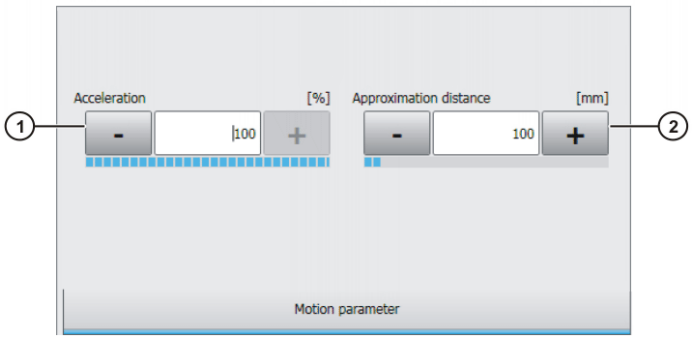


Obrázek 4.8. Inline formulář CIRC pohybu (Zdroj: [13])

Kliknutím na ikonku se šipkou vedle buňky s číslem 2 (resp. 3 u pohybu CIRC) se otevře okno možností „Frames“. Ukázka tohoto okna je na obr. 4.9. Je zde možné zvolit nástroj (TOOL_DATA[1] až TOOL_DATA[16]) a bázi (BASE_DATA[1] až BASE_DATA[32]). Dále pak pod číslem 3 můžeme nastavit interpolační režim na hodnotu „False“ (nástroj je namontován na montážní přírubě robota) nebo „True“ (nástroj je nepohyblivý – je umístěn mimo robota). A poslední možností, kterou lze nastavit, je „Rozpoznání kolize“ (možno nastavit na „True“ nebo „False“).



Obrázek 4.9. Okno možností „Frames“ (Zdroj: [13])



Obrázek 4.10. Okno možností pro nastavení dalších parametrů PTP pohybu (Zdroj: [13])

Kliknutím na ikonku se šipkou vedle buňky s číslem 5 (resp. 6 u pohybu CIRC) se otevře okno možností pro nastavení dalších parametrů. Ukázka tohoto okna pro pohyb PTP je na obr. 4.10 (pro ostatní pohyby je okno podobné). Je zde možné nastavit zrychlení v % a vzdálenost aproximace od koncového bodu v mm.

Princip práce s *inline formuláři* u ostatních instrukcí je obdobný, proto je zde nebudu uvádět. Podrobnější informace o práci s *inline formuláři* naleznete v [11] kap. 8, nebo v [13] kap. 9.

4.4 Struktura KRL programu

Každý robotický KRL program (dále jen program) se skládá ze 2 souborů. Je to soubor SRC a soubor DAT. Zda uživatel program v Navigátoru (souborový manager) vidí jako 2 soubory (SRC a DAT) nebo jen jako jeden programový modul, záleží na tom, zda je uživatel přihlášen jako „User“ nebo „Expert“. Protože uživatel typu „User“ nemá možnost upravovat (textově) KRL programy, tak se mu program zobrazí pouze jako programový modul skrývající v sobě oba dva soubory. Uživateli přihlášenému jako „Expert“ se jeden program v Navigátoru zobrazí jako 2 soubory se stejným názvem (název programu), pouze s rozdílnými příponami („.SRC“ a „.DAT“).

Soubor SRC obsahuje pouze programový kód, tzn. instrukce určující co má robot vykonávat. Permanentní data a souřadnice bodů využívané programem nejsou uvedeny přímo v instrukcích v souboru SRC, ale jsou odděleny do zvláštního souboru DAT, který je někdy také nazýván „seznamem dat“. Toto oddělení je výhodné zejména pro zachování dobré čitelnosti programu. Pokud upravujeme (nebo vytváříme) program, ve většině případů upravujeme pouze soubor SRC, takže nás neobtěžují například dlouhé definice souřadnic bodů. Pokud potřebujeme upravit zejména souřadnice bodů otevřeme soubor DAT a zde je modifikujeme.

Každý soubor SRC obsahuje nejdříve tzv. úvodní hlavičku, která je uvedena níže. Tuto hlavičku není potřeba ve většině případů měnit, kromě pátého řádku, kde je potřeba vyplnit název programu. Název programu musí být v rámci řídicího systému unikátní. Za touto hlavičkou následuje KRL program pro řízení robota, který je ukončen klíčovým slovem „END“.

```
&ACCESS RVP
&REL 1
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM EDITMASK = *
DEF nazev_programu( )
;FOLD INI
  ;FOLD BASISTECH INI
    GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
    INTERRUPT ON 3
    BAS (#INITMOV,0 )
  ;ENDFOLD (BASISTECH INI)
  ;FOLD USER INI
    ;Make your modifications here
  ;ENDFOLD (USER INI)
;ENDFOLD (INI)

; Vlastni KRL program pro rizeni robota

END
```

Vlastní KRL program by měl začínat a končit najetím do pozice HOME (základní pozice nastavená v řídicím systému, konkrétně v souboru `KRC:\R1\System\$config.dat`). Instrukce pohybu PTP do pozice HOME v jazyce KRL je uvedena níže.

```
;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,%P
 1:PTP, 2:HOME, 3:, 5:100, 7:DEFAULT
$BWDSTART = FALSE
PDAT_ACT=PDEFAULT
FDAT_ACT=FHOME
BAS (#PTP_PARAMS,100 )
$H_POS=XHOME
PTP  XHOME
;ENDFOLD
```

Každý soubor DAT obsahuje nejdříve, podobně jako soubor SRC, tzv. úvodní hlavičku, která je uvedena níže. Tuto hlavičku není potřeba ve většině případů měnit, kromě pátého řádku, kde je potřeba vyplnit název programu. Název programu musí být stejný jako název programu uvedený v souboru SRC. Za touto hlavičkou následuje deklarace dat a souřadnic bodů využívaných v souboru SRC. Kód je ukončen klíčovým slovem „ENDDAT“.

```
&ACCESS RVP
&REL 1
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM EDITMASK = *
DEFDAT  nazev_programu
;FOLD EXTERNAL DECLARATIONS;%{PE}%MKUKATPBASIS,%CEXT,%VCOMMON,%P
  ;FOLD BASISTECH EXT;%{PE}%MKUKATPBASIS,%CEXT,%VEXT,%P
    EXT  BAS (BAS_COMMAND  :IN,REAL  :IN )
    DECL INT SUCCESS
  ;ENDFOLD (BASISTECH EXT)
  ;FOLD USER EXT;%{E}%MKUKATPUSER,%CEXT,%VEXT,%P
    ;Make your modifications here
  ;ENDFOLD (USER EXT)
;ENDFOLD (EXTERNAL DECLARATIONS)

; deklarace dat a souradnic bodu

ENDDAT
```

4.5 Programování v uživatelské skupině „Expert“ pomocí KRL syntaxe

V uživatelské skupině „Expert“ lze samozřejmě využívat již zmíněné *inline formuláře*, ale základ programování tvoří KRL syntaxe. KRL (KUKA Robot Language) je programovací jazyk pro tvorbu robotických programů pro řídicí systémy společnosti KUKA Roboter GmbH.

Programy v KRL se zapisují velkými písmeny. Názvy programů, proměnných a bodů nesmí být delší než 24 znaků a smí se skládat pouze ze znaků (A-Z), číslic (0-9), znaku „_“ a „$“ (pro systémové účely). Názvy nesmějí začínat číslicí a samozřejmě se nesmějí shodovat s klíčovými slovy KRL jazyku.

K dispozici jsou 2 skupiny datových typů:

- jednoduché datové typy (uvedené v tab. 4.2),
- datové typy pro programování pohybů.

Jednoduché datové typy spolu s klíčovým slovem, popisem a popř. i rozsahem jsou uvedeny v tab. 4.2. Proměnné těchto datových typů můžeme deklarovat jak v souborech DAT, tak i v souborech SRC. Pokud proměnnou deklarujeme v datovém souboru, můžeme tuto proměnnou pomocí klíčového slova `GLOBAL` deklarovat jako globální. To znamená, že její platnost se rozšíří i na podprogramy uvedené v souboru SRC stejného jména jako soubor DAT.

Datový typ	Klíčové slovo	Popis a rozsah
Integer	INT	Celá čísla $(-2^{31}-1 \quad ... \quad 2^{31}-1)$
Real	REAL	Čísla s plovoucí řádovou čárkou $(+1,1\mathrm{E}-38 \quad ... \quad +3,4\mathrm{E}+38)$
Boolean	BOOL	Logická hodnota {True, False}
Characeter	CHAR	ASCII znak

Tabulka 4.2. Jednoduché datové typy jazyja KRL (Zdroj: [13])

Datových typů pro programování pohybů, nebo lépe pro deklaraci souřadnic bodů, je 5. Datovým typem `AXIS` můžeme bod deklarovat pomocí úhlů A1 až A6 natočení os robota pro tzv. osově specifický pohyb os robota.

```
STRUC AXIS REAL A1, A2, A3, A4, A5, A6
```

Datový typ `E6AXIS` je shodný s datovým typem `AXIS`, s možností deklarovat i 6 externích os (E1 až E6).

```
STRUC E6AXIS REAL A1, A2, A3, A4, A5, A6, E1, E2, E3, E4, E5, E6
```

Datovým typem `FRAME` můžeme bod deklarovat pomocí kartézských souřadnic X, Y, Z a hodnot A, B, C, což jsou natočení os X, Y, Z (A je rotace kolem Z, B je rotace kolem Y a C je rotace kolem X).

```
STRUC FRAME REAL X, Y, Z, A, B, C
```

Datový typ `POS` je podobný datovému typu `FRAME`, s možností deklarovat i hodnoty S (Status) a T (Turn). Údaje S a T slouží k zadání jednoznačně definované pozice robota v prostoru vzhledem k existujícím singularitám kinematiky.

```
STRUC POS REAL X, Y, Z, A, B, C, INT S, T
```

A poslední datový typ `E6POS` je shodný s datovým typem `POS`, s možností deklarovat i 6 externích os (E1 až E6).

```
STRUC E6POS REAL X, Y, Z, A, B, C, E1, E2, E3, E4, E5, E6, INT S, T
```

Pomocí klíčového slova `DECL` můžeme deklarovat například proměnné a pole. Pokud deklarujeme proměnnou v programu (SRC souboru), má deklarace tuto syntaxi.

```
⟨DECL⟩ Data_type Name1 ⟨, ..., NameN⟩
```

Pokud deklarujeme v seznamu dat (DAT souboru), můžeme navíc proměnnou současně inicializovat a deklarace má tuto syntaxi.

```
⟨DECL⟩ ⟨GLOBAL⟩ Data_type Name1 ⟨, ..., NameN⟩
⟨DECL⟩ ⟨GLOBAL⟩ Data_type Name = Value
```

Pokud deklarujeme proměnnou jednoduchého (základního) datového typu, není nutné uvádět klíčové slovo `DECL`.

Můžeme uvést několik příkladů deklarace proměnných a polí pro ilustraci použití příkazu DECL.

```
DECL INT X
DECL INT X1, X2
DECL REAL ARRAY_A[7], ARRAY_B[5], A
DECL BOOL CONFIRM = FALSE
```

Jednou z nejdůležitějších instrukcí pro tuto práci je instrukce pohybu PTP. Nejzákladnější tvar této instrukce pro pohyb PTP do bodu zadaného v kartézských souřadnicích je velice jednoduchý.

```
DECL FRAME pozice={X 709.8,Y 100.0,Z 1083.8,A 0.0,B 90.0,C 0.0}
PTP pozice
```

Takto můžeme zapsat instrukci pohybu PTP do programu (SRC soubru) bez nutnosti externího zápisu souřadnic bodu do souboru DAT.

Můžeme porovnat tento zjednodušený zápis s kódem generovaným *inline formulářem*. Následující blok kódu reprezentuje jednu instrukci pohybu PTP zapsanou v SRC souboru.

```
;FOLD PTP P1 Vel=100 % PDAT1 Tool[1]:kleste7 Base[0];%{PE}%R 8.2.20,
 %MKUKATPBASIS,%CMOVE,%VPTP,%P 1:PTP, 2:P1, 3:, 5:100, 7:PDAT1
$BWDSTART=FALSE
PDAT_ACT=PPDAT1
FDAT_ACT=FP1
BAS(#PTP_PARAMS,100)
PTP XP1
;ENDFOLD
```

Následující blok kódu reprezentuje data pro pohyb PTP a souřadnice koncového bodu zapsaná do souboru DAT.

```
DECL BASIS_SUGG_T LAST_BASIS={POINT1[] "P1                      ",
 POINT2[] "P1                       ",
 CP_PARAMS[] "CPDAT0                  ",
 PTP_PARAMS[] "PDAT1                    ",
 CONT[] "                         ",
 CP_VEL[] "2.0                      ",
 PTP_VEL[] "100                      ",
 SYNC_PARAMS[] "SYNCDAT                   ",
 SPL_NAME[] "S0                       "}
DECL E6POS XP1={X 1021.23999,Y 83.1957474,Z 992.817017,
 A 104.953003,B 89.9727325,C 100.209602,S 2,T 10,
 E1 0.0,E2 0.0,E3 0.0,E4 0.0,E5 0.0,E6 0.0}
DECL FDAT FP1={TOOL_NO 1,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " ",
 TQ_STATE FALSE}
DECL PDAT PPDAT1={VEL 100.0,ACC 100.0,APO_DIST 100.0,GEAR_JERK 50.0}
```

Tyto dva zápisy pohybové instrukce PTP, ale nejsou zcela totožné. Zatímco první uvedený zápis (jednoduchý) pouze vykoná pohyb PTP na zadané souřadnice, u druhého zápisu je nastaveno mnoho dalších parametrů, jako např. rychlost, zrychlení, aproximace a aproximační vzdálenost, zvolený nástroj a báze.

Pokud bychom nechtěli používat oddělení parametrů pohybu do datového souboru, je možné parametry PTP pohybu nastavit přímo v SRC souboru před vykonáním PTP pohybu. Tato možnost je vhodná pouze pro malé množství nastavovaných parametrů (např. rychlost a zrychlení). Toto nastavení se provádí pomocí instrukce BAS. Parametry pro pohybové instrukce nastavené pomocí instrukce BAS mají platnost do konce programu, nebo do změny parametru pomocí dalšího volání instrukce BAS.

Volba aproximace koncového bodu u instrukce PTP pohybu se provádí doplněním parametru „C_PTP“ za souřadnice přímo do instrukce PTP pohybu. Aproximační vzdálenost již bohužel nelze jednoduše nastavit pomocí volání instrukce BAS, ale je nutné ji definovat v datovém souboru.

Komplexnější volání pohybové instrukce PTP s aproximací koncového bodu (za hodnotu aproximační vzdálenosti je brána defaultně nastavená hodnota) a s nastavením rychlosti provádění pohybu a zrychlení je uvedeno v následujícím výpisu kódu.

```
DECL FRAME pozice={X 12.3,Y 100.0,Z 50,A 9.2,B 50,C 0}
BAS(#VEL_PTP,50)  ;nastaveni rychlosti na 50%
BAS(#ACC_PTP,20)  ;nastaveni zrychleni na 20%
PTP pozice C_PTP  ;pohyb PTP na souradnice urcene promennou 'pozice'
                  ;s aproximaci koncoveho bodu
```

Pro deklaraci souřadnic bodu je možné použít místo deklarace datového typu E6POS datový typ FRAME, u kterého se nezadávají čísla „S“ (Status), „T“ (Turn) a externí souřadnice „E1“ až „E6“.

```
DECL FRAME XP1={X 1000.0,Y 200.0,Z 1000.0,A -90.0,B 90.0,C -90.0}
```

Ostatní instrukce pohybu mají podobnou syntaxi jako dříve uvedená instrukce pohybu PTP. Nemá smysl dělat z této bakalářské práce manuál k programování robota, ale pouze nastínit nejdůležitější principy, a to zejména ty, které jsou použité v této práci. Proto pro podrobnější informace o dostupných instrukcích a jejich syntaxi odkáži na manuál [13] kap. 10.

4.6 Mapování signálů

Signál v pojetí řídicího systému KUKA je proměnná sdružující 1 a více fyzických vstupů nebo výstupů pod jedno jméno. Jeden vstup nebo výstup se může objevit i v několika deklarovaných signálech, potom je možné například jeden výstup ovládat pomocí dvou různých signálů (např. jeden signál 1 bitový a druhý 8-mi bitový). Pro práci s digitálními vstupy a výstupy není zapotřebí používat signály, ale jejich použití zpřehledňuje programy. Naopak k analogovým vstupům a výstupům je nutné přistupovat přes deklarované signály. Signály se velmi často používají pro ovládání nástroje připevněného k robotu (např. kleště, svařovací zařízení, atd.). Pomocí signálů můžeme také například posílat data o aktuální pozici TCP robota (při vykonávání programu) do PLC pro další zpracování, nebo naopak posílat různá data z PLC pro řízení běhu programu robota.

Deklarace signálu musí být provedena v deklarační sekci hlavičky souboru typu DAT, a má tuto syntaxi.

```
SIGNAL Signal_name Signal_variable (TO Signal_variable)
```

Kde `Signal_name` je název signálu, a `Signal_variable` je označení vstupu, resp. výstupu, na který se má signál namapovat. Dostupné `Signal_variables` jsou `$IN[x]`, `$OUT[x]`, `$ANIN[x]` a `$ANOUT[x]`, kde „`x`“ je číslo vstupu, resp. výstupu.

Můžeme uvést několik příkladů deklarace signálů pro ilustraci použití příkazu `SIGNAL`.

```
SIGNAL INPUT_FLAG $IN[3]
SIGNAL START_PROCESS $OUT[7]
START_PROCESS = TRUE
SIGNAL OUTWORT $OUT[1] TO $OUT[8]
OUTWORT = 'B01011100'
```

Jako příklad použití signálů pro zpřehlednění programu můžeme uvést například ovládání kleští, které jsou nainstalované na robotu. Otevření kleští bez použití signálů může vypadat takto.

```
$OUT[302] = TRUE
$OUT[301] = TRUE
WAIT FOR $IN[301] AND NOT $IN[302]
```

Tento způsob programování se nedoporučuje, protože je velmi nepřehledný. Pokud ovládání daného vstupu nebo výstupu používáme pouze na jednom místě v jednom programu, je toto řešení ještě schůdné, ale mělo by být doplněno komentáři o tom, k čemu který vstup a výstup slouží. Ale nejelegantnější řešení je nahradit přímé ovládání vstupů a výstupů signály. Signály je nutné deklarovat v souboru typu DAT. Pokud chceme deklarované signály využívat v různých programech, jako například pro ovládání kleští, je

potřeba signály deklarovat v souboru `KRC:\R1\System\$config.dat` v oblasti pro uživatelské definice. Kompletní deklarace signálů pro ovládání kleští (otevření a zavření) může vypadat takto.

```
SIGNAL SI_GR_OPEN $IN[301]
SIGNAL SI_GR_CLOSE $IN[302]
SIGNAL SO_GR_ENABLE $OUT[301]
SIGNAL SO_GR_OPEN $OUT[302]
```

A upravený program pro otevření kleští pomocí signálů potom vypadá takto.

```
SO_GR_OPEN = TRUE
SO_GR_ENABLE = TRUE
WAIT FOR SI_GR_OPEN AND NOT SI_GR_CLOSE
```

Signály deklarované pro systémové účely robota a řídicího systému jsou uloženy v souboru `KRC:\STEU\Mada\$machine.dat` a `KRC:\R1\System\$config.dat`. Deklarace těchto signálů by neměla být nikdy modifikována, ale je možné tyto signály používat (uvědoměle).

4.7 Přerušení

Přerušení, neboli interrupt je konstrukce, která při provádění programu hlídá nějaký stav nebo událost. Pokud tento stav nastane, je vyvolána tzv. obslužná rutina přerušení. Vyvolání obslužné rutiny přerušení zastaví vykonávání programu, vykoná se obslužná rutina přerušení, a pokračuje se ve vykonávání programu v místě, kde přerušení vzniklo.

Hlídaný stav (událost) může být např. signálový vstup (signál, viz kapitola 4.6), logická (booleovská) proměnná nebo logický výraz (porovnání, použití základních logických operací – NOT, OR, AND, EXOR). Obslužná rutina přerušení je podprogram definovaný obvykle za hlavním programem ve stejném SRC souboru.

Deklarace přerušení patří mezi tzv. „statement“ položky a musí být uvedena v sekci statement a nikoli v sekci deklarace. Deklarace přerušení má tuto syntaxi.

⟨*GLOBAL*⟩ `INTERRUPT DECL priorita WHEN udalost DO podprogram`

Kde `priorita` je číslo udávající prioritu deklarovaného přerušení, `udalost` je již zmíněný hlídaný stav (událost) a podprogram je již zmíněná obslužná rutina přerušení. Číslo `priorita` zároveň slouží k identifikaci daného přerušení. Toto číslo může být v rozsahu 1 až 128, kromě čísel 3 a 4 až 80, které jsou vyhrazeny pro systémové účely. V jeden okamžik smí být deklarováno maximálně 32 přerušení. Klíčovým slovem `GLOBAL` je možno přerušení deklarovat jako globální. To znamená, že toto přerušení bude deklarováno pro hlavní program i pro podprogramy definované v tomto programu.

Z toho vyplývá, že pokud chceme na danou událost reagovat přerušením kdykoli během provádění programu, tzn. i když je zrovna vykonáván podprogram volaný z hlavního programu, je nutné toto přerušení deklarovat jako globální.

Deklarované přerušení se aktivuje v programu následujícím příkazem.

```
INTERRUPT ON priorita
```

Kde `priorita` je číslo deklarovaného přerušení. Deaktivace přerušení je obdobná.

```
INTERRUPT OFF priorita
```

4.8 Singularity

Při vytváření pohybových programů (programování) zadáváme souřadnice bodů pomocí datových typů pro deklaraci souřadnic bodů (viz kapitola 4.5). Pokud používáme zadávání pomocí datových typů `AXIS` nebo `E6AXIS`, tzv. osově specifické pohyby, k žádným problémům s nejednoznačností nedochází. Pokud ale použijeme kartézské zadávání souřadnic (nejčastěji používané) pomocí datových typů `FRAME`, `POS` nebo `E6POS` mohou nastat tzv. *singulární polohy.*

Singulární poloha je taková poloha, u které není možné jednoznačně provést zpětnou transformaci (přepočet kartézských souřadnic na osově specifické hodnoty), i přesto, že bod byl zadán s údaji `S` (Status) a `T` (Turn). Singulární poloha nastává také při kartézském zadání bodu v bezprostřední blízkosti singulární polohy, což vede k příliš velkým (rychlým) změnám úhlů os.

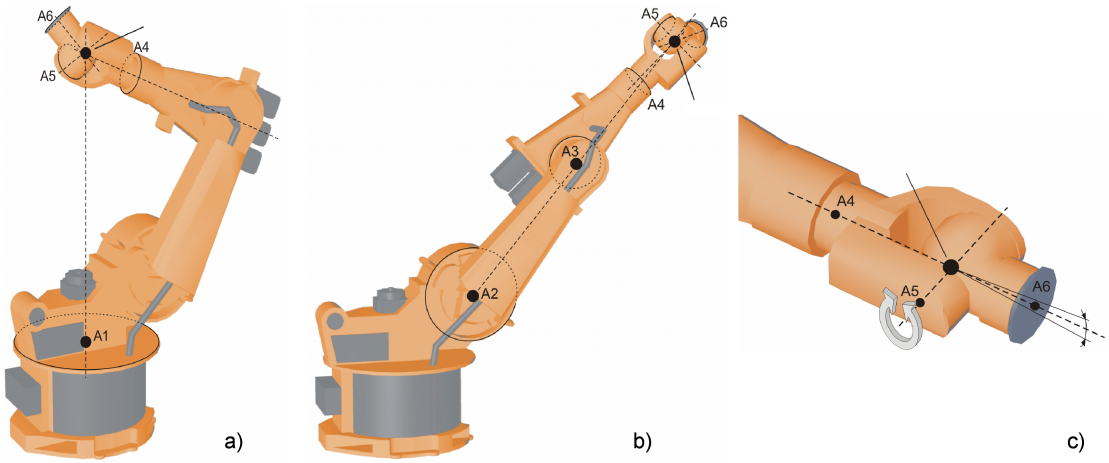
Robot *KUKA KR 5 ARC* se 6-ti stupni volnosti má 3 různé singulární polohy, které jsou zobrazeny na obr. 4.11.

1. Singularita „nad hlavou“ (viz obr. 4.11 a))
   Při této singularitě leží bod kořenu ruky (průsečík os A4, A5 a A6) na ose A1 robota. Pozici osy A1 nelze pomocí zpětné transformace jednoznačně určit, a proto může nabývat libovolných hodnot.

2. Singularita „natažených poloh“ (viz obr. 4.11 b))
   Při této singularitě leží bod kořenu ruky (průsečík os A4, A5 a A6) na prodloužení osy A2 a A3 robota, který se nachází na okraji svého pracovního prostoru. Zpětná transformace dává sice jednoznačné úhly os, ale nízké kartézské rychlosti mají za následek vysoké rychlosti osy A2 a A3.

3. Singularita „osy ruky“ (viz obr. 4.11 c))
   Při této singularitě leží osy A4 a A6 rovnoběžně a natočení osy A5 je v rozmezí

$\pm 0,01812°$. Polohu obou os není možné jednoznačně stanovit pomocí zpětné transformace. Existuje nekonečně mnoho možných poloh os A4 a A6, jejichž součet osových úhlů je stejný.



Obrázek 4.11. Druhy singulárních poloh (Zdroj: [19])

Obecný postup pro řešení problémů se singulárními polohami neexistuje. Pokud by byl námi nastavený bod singulární polohou, řídicí systém robota nám to oznámí, a my na to můžeme reagovat. Nejjednodušší nápravou je posunout zadávaný bod, ale to ve většině případů není možné. Singularitu „osy ruky“ je možné vyřešit najetím námi zadávaného bodu s jiným natočením os A4 a A6 (tzn. natočit osu A5 o více než $\pm 0,01812°$). Všechny singularity lze částečně omezit snížením zrychlení na velmi malou hodnotu, ale to je opět ve většině případů nepřijatelné. Řešení singulárních poloh nám nabízí také přímo řídicí systém, kde pomocí systémové proměnné `$SINGUL_POS[1-3]` lze nastavit co se má při které singulární poloze provést. Více informací naleznete v manuálu [13] kap. 8.10, odkud je také částečně převzat text této kapitoly.

Obsah celé kapitoly 4 vznikl za pomoci informačních zdrojů [11], [13] a [19].

Kapitola 5
Propojení řídicího systému robota s PC

Zásadní otázkou této práce, podle mého mínění, je: „Jakým způsobem propojit senzor *Kinect V2* s řídicím systémem robota.“. Protože Kinect není průmyslový senzor, tak nemá ani podporu průmyslových sběrnic, které řídicí systém robota podporuje (např.: Profibus, Profinet, EtherCAT nebo RS–232). A kvůli nutnosti připojení *Kinectu V2* k portu USB 3.0, není možné přemýšlet ani o možnosti připojit senzor *Kinect V2* přímo k USB portu řídicího systému. Z této myšlenky vyplývá jediný rozumný závěr, a to připojit senzor *Kinect V2* k portu USB 3.0 externího počítače s operačním systémem Windows 10 (možno i verze 8, viz tab. 2.1).

Pokud porovnáme společné sběrnice dostupné jak na externím PC, tak i v řídicím systému robota, dospějeme k závěru, že tyto 2 systémy by bylo možné propojit pomocí sériové linky (RS–232) nebo sítě Ethernet. Přenosové rychlosti sériové linky nejsou velké, ale protože množství dat přenášené za sekundu bude dostatečně malé, dalo by se o použití sériové linky uvažovat. Ale protože se sériová linka v dnešní době pro nově vznikající aplikace téměř nevyužívá a v zařízeních zůstává spíše kvůli zpětné kompatibilitě, tuto možnost zavrhnu.

Nejrozumnější způsob propojení řídicího systému robota se senzorem připojeným k externímu PC se zdá být síťové rozhraní Ethernet. Řídicí systém robota je schopen používat ethernetové rozhraní pro řídicí účely pouze se softwarovým doplňkem *KUKA Ethernet KRL*. Tento doplněk je již v řídicím systému robota nainstalován, takže nic nebrání jeho použití.

5.1 KUKA Ethernet KRL

KUKA Ethernet KRL je softwarový doplněk pro řídicí systémy robotů od společnosti KUKA Roboter GmbH. Tento doplněk využívá klasickou fyzickou vrstvu sítě Ethernet, která je dostupná již v základu řídicího systému robota. Základní verzi Ethernetu, která je dlouholetým standardem osobních počítačů, ale nelze použít, protože není vhodná pro průmyslové použití. Ale softwarový doplněk *KUKA Ethernet KRL* vytvoří z běžného ethernetového spojení deterministické ethernetové spojení, které je vhodné i pro průmyslové použití.

Síťové spojení a přenos dat je zajištěno pomocí TCP/IP protokolu. Doba jednoho komunikačního cyklu velice závisí na objemu odesílaných a přijímaných dat a způsobu jejich zpracování pomocí KRL programu. Vhodnými programovacími technikami lze dosáhnout doby komunikačního cyklu až 2 ms. Dosažení takto nízké doby komunikačního cyklu nebude zapotřebí, protože senzor *Kinect V2* pracuje s periodou přibližně 33 ms.

Základní způsob komunikace je založen na výměně dat ve formátu XML. Forma v jaké se mají XML data odesílat, nebo naopak v jaké formě čekat příchozí XML data, je uvedena vždy na začátku programu v SRC souboru. Konfigurace ethernetového spojení je provedena pomocí XML souboru.

Data, odesílaná řídicím systémem robota externímu systému (v tomto případě počítač), je nutné nejdříve uložit do paměti se strukturou danou pomocí XML formy definované na začátku programu. Jakmile jsou všechna data pro odeslání uložená v paměti, vyvolá se instrukce, která je načte jako jeden blok a odešle najednou do externího systému.

Data, která jsou přijata řídicím systémem robota od externího systému, jsou uložena do mezipaměti se strukturou danou pomocí XML formy definované na začátku programu. Tato forma musí být stejná jako ta, se kterou jsou data odesílána externím systémem. Programátor potom již pouze přistupuje ke strukturovaným datům v paměti pomocí KRL instrukcí. Tyto přijatá data uložená v mezipaměti si může programátor uložit do vlastních KRL proměnných pro následné využití v programu. Protože jsou všechna přijatá data ukládána do paměti ve frontě, kde čekají na zpracování programem, nemůže se stát, že by se některá data ztratila. Tím je zajištěna jedna z podmínek pro použití sítě Ethernet v průmyslových aplikacích.

5.2 Konfigurace ethernetového spojení

Konfigurace ethernetového spojení je provedena na základě konfiguračního XML souboru. Tento konfigurační XML soubor musí mít stejný název jako název spojení používaný v KRL programu a musí být umístěn v adresáři `C:\KRC\ROBOTER\Config\user\Common\EthernetKRL` v řídicím systému robota. Například inicializační příkaz pro navázání spojení `EKI_Init("EthConn")` se pojí s XML konfiguračním souborem `...\EthConn.XML`.

Konfigurační XML soubor pro komunikaci přes Ethernet má základní strukturu složenou ze tří částí: `CONFIGURATION`, `RECEIVE` a `SEND`. V těchto částech se definují parametry ethernetového spojení a tvar odesílaných a přijímaných dat. U přijímaných

dat robotem se navíc uvádí ještě datový typ a je možné využít možnosti automatického nastavení zadaného výstupu nebo tzv. „flagu“ při přijetí prvku, resp. všech prvků (dat).

Myslím si, že účelem této práce není detailní popis možností různých nastavení komunikace pomocí *KUKA.Ethernet KRL*, proto pro ukázku konfiguračního XML souboru uvedu pouze konfigurační XML soubor použitý pro nastavení komunikace u demonstrační aplikace (viz kap. 6) této bakalářské práce. Kód tohoto konfiguračního XML souboru je uveden v příloze E. Více informací o konfiguračním XML souboru pro nastavení komunikace po Ethernetu naleznete v manuálu k doplňku *KUKA.Ethernet KRL* [14] v kap. 6.1.

5.3 Používání komunikace přes Ethernet v KRL programech

Navázání spojení s externím systémem po síti Ethernet a následná komunikace (vzájemná výměna dat) probíhá v KRL programu pomocí speciálních KRL funkcí doplňku *KUKA.Ethernet KRL*. Kompletní přehled těchto funkcí s jejich popisem je uveden v manuálu k tomuto doplňku viz [14] kap. 9.5.

Předpokládejme, že máme uložený XML konfigurační soubor ve správném umístění s obsahem uvedeným v příloze E pojmenovaný například `XMLkonfigurace.XML`. Potom navázání a ukončení komunikace s externím systémem provedeme v KRL programu následovně.

```
RET=EKI_Init("XMLkonfigurace") ;inicializuje kanal ethernetoveho rozhrani
                                 pro pripojeni k externimu systemu
RET=EKI_Open("XMLkonfigurace") ;otevre kanal a tim navaze spojeni s
                                 externim systemem
...
 ;Odesilani dat, prijimani dat a vykonavani ostatnich KRL instrukci
...
RET=EKI_Close("XMLkonfigurace") ;uzavre kanal, a tim ukonci spojeni s
                                  externim systemem
RET=EKI_Clear("XMLkonfigurace") ;uvolni kanal pro dalsi pouziti
```

Kde `RET` je proměnná deklarovaná v deklarační oblasti jako `DECL EKI_STATUS RET` a používá se pro uložení návratových hodnot volaných KRL funkcí doplňku *KUKA.Ethernet KRL*. Pomocí návratových hodnot uložených v této proměnné můžeme dále testovat správnost provedené funkce (např. zda bylo spojení správně navázáno).

Odesílání dat robota externímu systému je rozděleno do dvou kroků. Nejdříve je nutné uložit odesílaná data, ve struktuře definované na začátku SRC souboru s KRL

programem (viz příloha B), do bufferu (mezipaměti) pro odeslání. Jakmile jsou všechna data připravena v bufferu, je možné zavolat funkci pro odeslání dat, která všechna data z bufferu načte a následně odešle externímu systému jedním ethernetovým paketem. Odesílání dat je dále ukázáno na výňatku kódu z KRL programu demonstrační aplikace (viz kap. 6).

```
; zapsani dat robota do bufferu ethernetoveho spojeni
RET=EKI_SetFrame("XMLkonfigurace","Robot/Data/PoziceTCPRobota", $POS_ACT)
; odeslani dat robota externimu systemu
RET=EKI_Send("XMLkonfigurace","Robot")
```

Přijímání dat od externího systému je v zásadě jednodušší, protože o správné přijetí a postupné uložení přijatých dat do bufferu se stará řídicí systém. Programátor pouze pomocí funkcí vyčítá přijatá data z bufferu a ukládá si je do vlastních proměnných pro další využití. Ukládání přijatých dat je dále ukázáno na výňatku kódu z KRL programu demonstrační aplikace (viz kap. 6).

```
FRAME kinectFrame   ;deklarace vlastni promenne pro data typu FRAME
BOOL kinectPripojen ;deklarace vlastni promenne pro data typu BOOL
RET=EKI_GetFrame("XMLkonfigurace","Sensor/Pozice/NoveSouradnice",
                                                        kinectFrame)
RET=EKI_GetBool("XMLkonfigurace","Sensor/Status/KinectPripojen",
                                                        kinectPripojen)
```

Kapitoly 5.1, 5.2 a 5.3 vznikly za pomoci informačního zdroje [14].

5.4 Struktura dat odesílaných serverovou aplikací

V tuto chvíli se budu zabývat pouze strukturou odesílaných dat které serverová aplikace odesílá po síti Ethernet do řídicího systému robota, a zbytek aplikace bude popsán v kapitole 6.4.

XML struktura dat, která se má odeslat je vytvořena pomocí šablony. Šablona představuje XML soubor (uvedený v příloze G), který obsahuje pouze textovou XML strukturu bez hodnot. Tento soubor je v aplikaci načten a následně doplněn o aktuální hodnoty. Poté může být jako celek odeslán řídicímu systému robota pro zpracování a vykonání pohybu na nově zadané souřadnice.

5.5 Komunikace pomocí doplňku KUKA.RSI

Nakonec této kapitoly, ke způsobům propojení senzoru *Kinect V2* s průmyslovým robotem, ještě krátká úvaha o využití doplňku *KUKA.RobotSensorInterface* zkráceně

KUKA.RSI místo doplňku *KUKA.Ethernet KRL*. Doplněk *KUKA.Ethernet KRL* je určený spíše pro přenos informací s možností použít tyto informace k vykonání určitých pohybů, ale není přímo určený pro real-time aplikace. Dá se říci, že tato demonstrační aplikace není tzv. „hard real-time“ aplikace (aplikace citlivá na přesné řízení v krátkých, stejně dlouhých, časových intervalech), proto je možné doplněk *KUKA.Ethernet KRL* použít. Pokud bychom ale chtěli dosáhnout přesnější a rychlejší reakce, byli bychom nuceni použít doplněk *RSI*.

Tento doplněk umožňuje provádět pohyb vykonávaný na základě dat ze senzoru, a to v reálném čase. Pro komunikaci využívá také sítě Ethernet a data posílá také v XML struktuře. Změna v komunikaci je v použitém protokolu, místo TCP využívá protokol UDP, který má mnohem menší režii, tzn. velikost výsledných posílaných paketů je menší.

Pro použití tohoto doplňku je nutné vytvořit XML konfigurační soubor ethernetového spojení, který je podobný zmíněnému v kapitole 5.2. Dále potom 3 konfigurační soubory tzv. RSI diagramu, což je objektově nakonfigurované spojení a akce prováděné s přijatými daty (např. vykonání pohybu). Tato konfigurace se provádí pomocí softwaru *RSIVisual* dodaného spolu s doplňkem. Z vytvořené konfigurace tento software vygeneruje potřebné 3 soubory.

Samotný KRL program je potom velice jednoduchý. Je v něm pouze inicializace RSI diagramu, zapnutí vykonávání podle nastaveného RSI diagramu a poté již jen jedna instrukce, která zajistí provádění pohybu na základě dat ze senzoru, podle konfigurace RSI diagramu. Na konci programu se pouze zajistí vypnutí a odpojení používaného RSI diagramu.

Vytvoření RSI diagramu a spolupracujícího XML konfiguračního souboru není zcela jednoduché a bohužel nebyl dostatek času na vyzkoušení tohoto doplňku, který je již velice komplexní. Další menší problém v použití tohoto doplňku vidím v tom, že je potřeba zaručit ze strany externího systému (senzoru) doručení nové informace pro pohyb prováděný na základě dat ze senzoru každé 4 ms. Jakékoli prodloužení tohoto intervalu by vedlo k okamžitému konci programu.

Text této kapitoly vznikl za pomoci informačního zdroje [20], kde je také možné najít více informací k tomuto doplňku.

Na instalaci, oživování, testování a seznamování se s tímto (*KUKA.RSI*) i předchozím (*KUKA.Ethernet KRL*) doplňkem jsem spolupracoval s kolegou Pavlem Králem, který tento typ komunikace mezi externím PC a řídicím systémem potřeboval také pro svou bakalářskou práci.

Kapitola 6
Demonstrační aplikace

Tvorba demonstrační aplikace byla hlavním cílem této bakalářské práce. Jak již bylo zmíněno v úvodu, demonstrační aplikace by měla ukázat, že spojení robotického manipulátoru s obrazovým senzorem primárně určeným k herní konzoli, je možné a má smysl. Tuto aplikaci bude také možné využít jako zpestření prezentace katedry Řídicích systémů Fakulty elektrotechnické Českého vysokého učení technického v Praze například při Dnech otevřených dveří. Název demonstrační aplikace je „Robotické provádění pohybu snímaného senzorem Kinect V2“, ale dále bude zmiňována pouze jako demonstrační aplikace.

Tato demonstrační aplikace mohla být vytvořena pouze po důkladném seznámení se s programováním senzoru Kinect V2 a zejména možnostmi a způsobem programování průmyslového robota KUKA. Získané znalosti a zkušenosti při testování jednotlivých systémů jsou obsaženy v předchozích kapitolách. Protože programování nadřazeného PLC a tvorba uživatelského rozhraní pro ovládání aplikace z operátorského panelu (vizualizační dotykový panel připojený k PLC) bylo použitou pouze okrajově, nebylo zapotřebí hlubší seznámení se s těmito prvky. Proto jim není věnována zvláštní kapitola a bude popsáno pouze jejich použití v této demonstrační aplikaci (viz kap. 6.5 a 6.6).

Výsledná demonstrační aplikace nabízí 2 varianty provozu. První, a hlavní, variantou je *online* varianta, která pohyb snímaný senzorem Kinect V2 okamžitě převádí na pohyb robota. Druhou variantou je *offline* varianta, u které je možné pohyb nasnímat senzorem Kinect V2 a poté celý pohyb přenést do řídicího systému robota a provést kompletní předvedený pohyb robotem. Hlavní nevýhodou této varianty je neprovádění pohybu v reálném čase. Na druhou stranu velkou výhodou této varianty je souvislost a plynulost pohybu prováděného robotem (na rozdíl od *online* varianty) a také možnost libovolného množství opakování načteného pohybu.

Softwarová část demonstrační aplikace se skládá z několika částí umístěných a spouštěných v různých zařízení. V externím systému se jedná o serverovou aplikaci (viz kap. 6.4), v řídicím systému robota o 2 soubory robotického KRL programu (viz kap. 6.2 a 6.3) a poslední částí je společný projekt v softwaru Siemens TIA portal Step 7 pro nadřazené PLC a operátorský panel (viz kap. 6.5 a 6.6). Jednotlivé části budou podrobněji rozebrány v následujících kapitolách.

6.1 Konfigurace ethernetového spojení demonstrační aplikace

Jak již bylo vysvětleno v kap. 5.2, ethernetové spojení a struktura přenášených dat musí být nastavena pomocí konfiguračního XML souboru. Název tohoto souboru je poté použit jako identifikátor použitého ethernetového spojení v robotickém programu (SRC souboru). Konfigurační XML soubor použitý pro tuto demonstrační aplikaci je uveden v příloze E a jeho název je `KinectAplikace.XML`. Na začátku tohoto souboru je definována IP adresa externího PC, číslo portu, ke kterému se má řídicí systém robota připojit a informace o tom, že externí PC je v tomto spojení považováno za „server“, z čehož vyplývá, že řídicí systém je v roli „klienta“. Dále je v souboru specifikován XML tvar přijímaných a odesílaných dat.

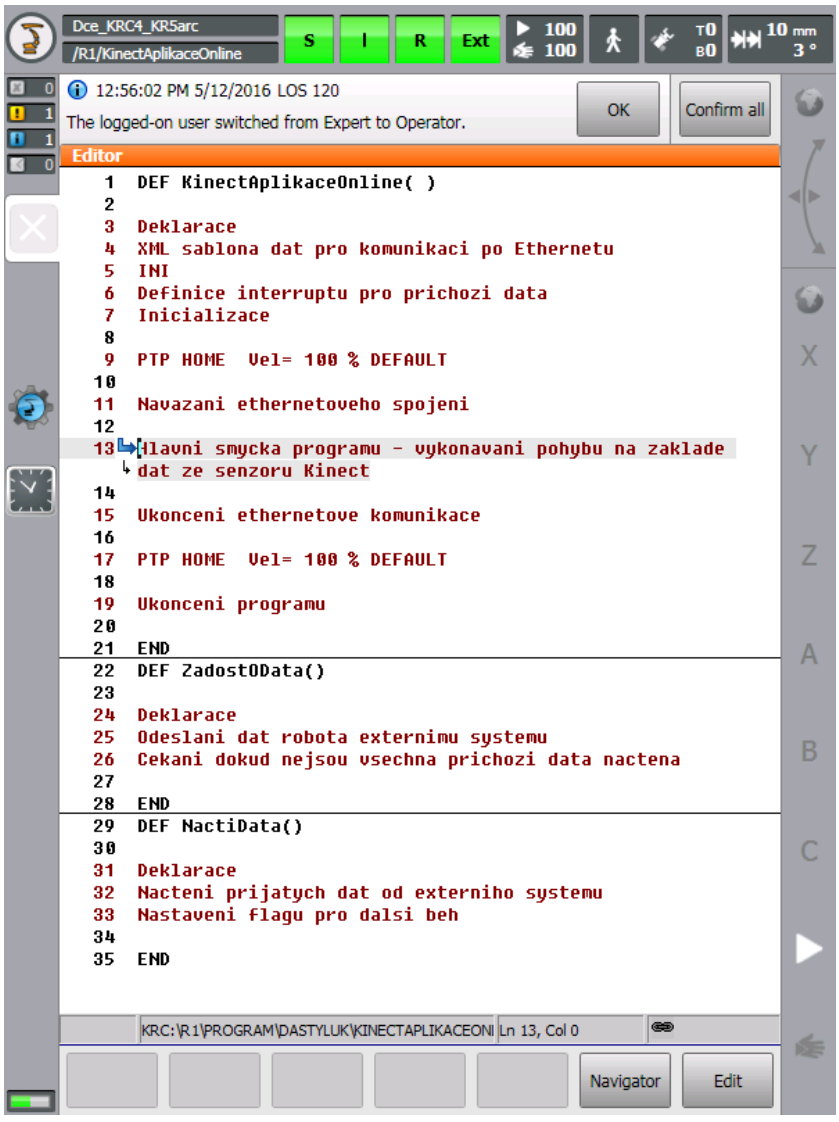
Řídicí systém robota přijímá souřadnice nového bodu (`X, Y, Z, A, B, C`) ve tvaru datového typu `FRAME` a dále pak 2 hodnoty typu boolean nesoucí informaci o připojeném senzoru Kinect V2 a zda mají být kleště otevřeny, nebo zavřeny. Řídicí systém robota odesílá pouze souřadnice aktuální pozice bodu `TCP`. Tvar XML přijímaných a odesílaných dat je také uveden v úvodu robotického programu (SRC souboru) pro zlepšení přehlednosti.

V robotickém programu jsou pak používány funkce doplňku *KUKA.Ethernet KRL* pro navázání a ukončení spojení, odesílání a přijímání dat. Návratová hodnota těchto funkcí je ukládána do proměnné `RET` typu `EKI_STATUS`, což je struktura se 4 prvky. Prvním prvkem je `Buff` typu `Integer`, který nese informaci o počtu prvků, které jsou přijaté a dostupné (ještě nepřečtené) v mezipaměti (bufferu). Druhým prvkem je `Read` typu `Integer`, který nese informaci o počtu již přečtených přijatých prvků z mezipaměti. Třetím prvkem je `Msg_No` také typu `Integer`. Obsahem tohoto prvku je číslo chyby, k níž došlo při volání funkce. Posledním prvkem je `Connected` typu `Boolean`, který nese aktuální informaci o ethernetovém spojení. Ne všechny funkce, ale vracejí všechny prvky této struktury. Pouze prvek `Msg_No` vracejí všechny funkce. Prvky této struktury se dají po volání funkce v KRL programu testovat a rozhodovat podle nich další běh programu. V této demonstrační aplikaci je použit pouze prvek `Connected` pro kontrolu navázání spojení. Pokud by spojení s externím systémem nebylo navázáno, KRL program se ukončí. Pro testování prvků této struktury v programu je nutné v souboru `KRC:\R1\TP\EthernetKRL\EthernetKRL.DAT` změnit hodnotu proměnné `SHOWMSG` na hodnotu `FALSE` (běžně je nastavena hodnota `TRUE`). Tím zakážeme zobrazování chybových hlášek doplňku *KUKA.Ethernet KRL* a převezmeme nad nimi kontrolu vlastním vyhodnocením v programu.

Text této kapitoly vznikl za pomoci informačního zdroje [14].

6.2 Robotický program v řídicím systému robota – online varianta

V řídicím systému robota se nachází robotický program v KRL jazyce. Zdrojový kód a jeho umístění v řídicím systému robota je uvedeno v příloze B. Tento program zajišťuje komunikaci po síti Ethernet se server programem spuštěným v externím PC se současným vykonáváním pohybu robota. Zdrojový kód je plně komentovaný a používá proměnné, jejichž název plně vystihuje jejich význam, proto orientace v programu není složitá ani pro nového („expertního“) uživatele. Program je vytvořen tak, aby byl srozumitelný i pro běžného uživatele, který program pouze spouští. To je zajištěno skrytím všech instrukcí programu do tzv. „FOLDů“ (byly vysvětleny již dříve), které pouze popisují českým komentářem co program právě vykonává. Snímek obrazovky ovládacího panelu robota s tímto právě běžícím programem je uveden na obr. 6.1.



Obrázek 6.1. Snímek obrazovky ovládacího panelu robota s právě běžící demonstrační aplikací – varianta online

Všechny proměnné a signály, které jsou v programu používány, jsou deklarovány v souboru `$config.dat` pro globální použití ve všech programech a podprogramech této demonstrační aplikace. Výjimkou je pouze lokální proměnná sloužící pro uložení návratové hodnoty KRL funkcí doplňku *KUKA.Ethernet KRL*, která by mohla být přepsána cizím programem, pokud by byla deklarována globálně. Výtah kódu z tohoto souboru, kde jsou deklarované proměnné a signály pro tuto demonstrační aplikaci, je uveden v příloze F. K tomuto programu také náleží příslušný datový soubor se stejným názvem jako SRC soubor. V tomto souboru je deklarovaný pouze vzor pro instrukci `BAS`, pomocí které se nastavují parametry pohybových instrukcí. Tento soubor je uveden v příloze D.

Na začátku programu je provedena inicializace všech proměnných a stavu kleští (provede se jejich otevření). Poté je provedeno navázání ethernetového spojení se serverovým programem spuštěným na externím PC. Pokud je toto spojení navázáno, program pokračuje, jinak je ukončen. Dále je provedeno načtení první souřadnice a stavu senzoru Kinect z externího PC. Poté následuje nastavení rychlosti a zrychlení pro prováděný pohyb. Následuje hlavní smyčka programu, ve které je vykonán pohyb na první načtené souřadnice, a je provedena akce otevření, popř. zavření kleští pokud je to externím PC signalizováno. Na konci hlavní smyčky je externí PC požádáno o nové souřadnice, a jakmile je řídicí systém robota dostane a všechny správně načte, může se hlavní smyčka opakovat. Do hlavní smyčky se program dostane pouze po stisknutí tlačítka „Spustit provádění pohybu“ na operátorském panelu. Ukončení provádění hlavní smyčky je možné stisknutím tlačítka „Ukončit provádění pohybu“ na operátorském panelu, nebo pokud od externího PC přijde informace o odpojeném senzoru Kinect V2 při provádění pohybu. Po ukončení hlavní smyčky programu následuje ukončení ethernetového spojení s externím PC, najetí do pozice HOME (výchozí pozice robota) a ukončení programu. Ukončení programu je potřeba potvrdit stisknutím tlačítka „Ukončit program“ na operátorském panelu.

Přenos nových dat je po síti Ethernet mezi řídicím systémem robota a externím PC proveden vždy bezprostředně před jeho vykonáním robotem. Proces přenosu dat je iniciován vždy robotem. Řídicí systém robota zažádá externí PC o nová data odesláním zprávy, ve které uvede svou aktuální polohu bodu `TCP`. Žádost o data je vykonána zavoláním podprogramu `ZadostOData()`. V tomto podprogramu se uloží aktuální pozice robota do mezipaměti k odeslání, a poté se zpráva odešle externímu PC a následně se čeká než bude nastaven `$FLAG[1]`. Externí PC tuto zprávu přijme, informaci nesenou touto zprávou použije k výpočtu souřadnic nového bodu, které i s ostatními informacemi (zda je senzor Kinect V2 připojen a zda mají být kleště zavřeny, nebo oteřeny) ihned odešle řídicímu systému robota. V XML konfiguračním souboru ethernetového spojení (viz příloha E) je uvedeno, že při přijetí všech dat od externího PC se má nastavit `$FLAG[998]`. Na tento „flag“ je nastaveno přerušení,

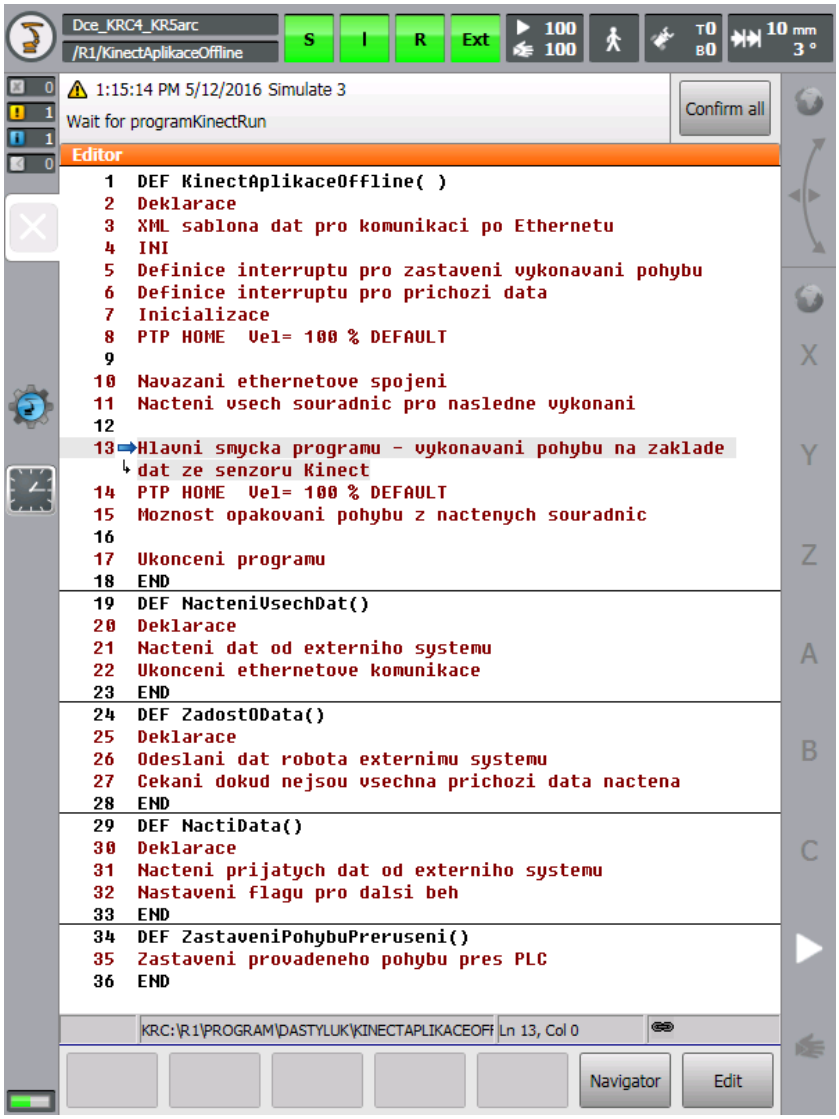
tzn. že pokud řídicí systém robota přijme všechna požadovaná data od externího PC, nastaví `$FLAG[998]=TRUE` a je vyvolána obslužná rutina přerušení (v tomto případě podprogram `NactiData()`). V tomto podprogramu se přijatá data uloží do pracovních proměnných programu a nastaví se `$FLAG[1]=TRUE` pro ukončení přenosu dat a umožnění návratu do hlavní smyčky programu.

Nastavení správné kombinace rychlosti a zrychlení prováděného pohybu robotem je velice důležité. Závisí na něm plynulost a přesnost prováděného pohybu. Pokud je u této varianty nastaveno příliš velké zrychlení, dochází k rychlým a prudkým změnám pohybu, což je pro vizuální stránku i pro robota samotného nevhodné. Nastavené zrychlení tedy musí být relativně malé. U této varianty byla experimentálně zjištěna jako nejlepší hodnota 15 %. U nastavení rychlosti je to jiné. Pokud by byla nastavena nízká hodnota, robot by nestíhal provádět předváděný pohyb. Experimentálně bylo ověřeno, že pro tuto variantu je nejvhodnější nastavit maximální rychlost provádění pohybu (100 %).

V této variantě demonstrační aplikace lze ovládat pohyb kleští robota v jeho pracovním prostoru pomocí předvádění pohybu pravou rukou před senzorem Kinect V2. Pomocí levé ruky lze ovládat otevření a zavření kleští. Pokud je levá ruka nad úrovní ramen, jsou kleště otevřené, pokud je levá ruka pod úrovní ramen, jsou kleště zavřené. V obou dvou případech je slovem „ruka“ myšlena dlaň dané paže. Tímto způsobem ovládání je možné pomocí kleští robota uchopit předmět umístěný v jeho pracovním prostoru a přemístit ho na jiné místo v prostoru.

6.3 Robotický program v řídicím systému robota – offline varianta

Podobně jako u *online* varianty demonstrační aplikace je v řídicím systému robota umístěn robotický program v KRL jazyce. Zdrojový kód a jeho umístění je uvedeno v příloze C. Tento program zajišťuje komunikaci po síti Ethernet se server programem spuštěným v externím PC s následným vykonáváním pohybu robota. Stejně jako zdrojový kód *online* varianty je i tento kód plně komentovaný. Některé používané proměnné jsou shodné s proměnnými využívanými u *online* varianty, ostatní jsou deklarovány také v souboru `$config.dat`. K tomuto programu také náleží příslušný datový soubor se stejným názvem jako SRC soubor. V tomto souboru je deklarovaný pouze vzor pro instrukci `BAS`, pomocí které se nastavují parametry pohybových instrukcí. Tento soubor je uveden v příloze D. I tento program má skryté všechny instrukce do „FOLDů“ pro snadný přehled nad probíhajícím programem. Snímek obrazovky ovládacího panelu robota s tímto právě běžícím programem je uveden na obr. 6.2.



Obrázek 6.2. Snímek obrazovky ovládacího panelu robota s právě běžící demonstrační aplikací – varianta offline

Na začátku programu je provedena inicializace všech proměnných. Poté je provedeno navázání ethernetového spojení se serverovým programem spuštěným na externím PC. Pokud je toto spojení navázáno, program pokračuje, jinak je ukončen. Poté je provedeno načtení všech souřadnic a ukončení ethernetového spojení s externím PC. Dále pak nastavení rychlosti a zrychlení pro prováděný pohyb. Následuje hlavní smyčka programu, ve které je vykonán aproximovaný pohyb přes načtené souřadnice. Do hlavní smyčky se program dostane pouze po stisknutí tlačítka „Spustit provádění pohybu“ na operátorském panelu. Ukončení provádění hlavní smyčky je možné stisknutím tlačítka „Ukončit provádění pohybu“ na operátorském panelu. Tím dojde k vyvolání obslužné rutiny přerušení `ZastaveniPohybuPreruseni()`, která ukončí hlavní smyčku, neboli prováděný pohyb. Po ukončení hlavní smyčky programu následuje najetí do pozice HOME (výchozí pozice robota) a čekání na zvolenou odpověď na operátorském panelu, kde se

zobrazí dotaz, zda se má provedený pohyb opakovat nebo ne. Pokud uživatel zvolí možnost *opakovat pohyb*, program přejde na začátek hlavní smyčky a aproximovaný pohyb se zopakuje přes načtené body. Pokud uživatel zvolí možnost *neopakovat pohyb*, dojde k ukončení programu. Ukončení programu je potřeba potvrdit stisknutím tlačítka „Ukončit program“ na operátorském panelu.

Jediný rozdíl v přenosu nových dat po síti Ethernet mezi variantou *online* a *offline* je v tom, že u varianty *online* se přenáší pouze souřadnice jednoho bodu a ten je vzápětí vykonán robotem, ale u varianty *offline* jsou přeneseny všechny body před tím, než je pohyb vykonán. Proces přenosu dat je iniciován vždy robotem. Řídicí systém robota zažádá externí PC o nová data odesláním zprávy, ve které uvede svou aktuální polohu bodu `TCP`. Žádost o data je vykonána zavoláním podprogramu `ZadostOData()`. V tomto podprogramu se uloží aktuální pozice robota do mezipaměti k odeslání, a poté se zpráva odešle externímu PC a následně se čeká než bude nastaven `$FLAG[1]`. Externí PC tuto zprávu přijme, informaci nesenou touto zprávou použije k výpočtu souřadnic nového bodu, které i s informací, zda je bod poslední či ne, ihned odešle řídicímu systému robota. Pro přenos informace, zda je bod poslední či ne, je použit prvek XML struktury `KinectPripojen`. V XML konfiguračním souboru ethernetového spojení (viz příloha E) je uvedeno, že při přijetí všech dat od externího PC se má nastavit `$FLAG[998]`. Na tento „flag“ je nastaveno přerušení, tzn. že pokud řídicí systém robota přijme všechna požadovaná data od externího PC, nastaví `$FLAG[998]=TRUE` a je vyvolána obslužná rutina přerušení (v tomto případě podprogram `NactiData()`). V tomto podprogramu se přijatá data uloží do pracovních proměnných programu a nastaví se `$FLAG[1]=TRUE` pro ukončení přenosu dat a umožnění návratu do podprogramu `NacteniVsechDat()`. V tomto podprogramu běží smyčka načítání nových dat do té doby, než je prvek XML struktury `KinectPripojen` nastaven externím PC na hodnotu `TRUE`. Tím jsou načteny souřadnice všech nových bodů, a může dojít k ukončení spojení a následnému vykonání pohybu z těchto souřadnic.

Nastavení správné kombinace rychlosti a zrychlení prováděného pohybu robotem je velice důležité. Závisí na něm především plynulost prováděného pohybu. Pokud by u této varianty byla nastavena příliš vysoká hodnota rychlosti, robot by předvedený pohyb provedl rychleji, než byl předveden, což je nežádoucí. Nastavená rychlost tedy musí být relativně malá. U této varianty byla experimentálně zjištěna jako nejlepší hodnota 30 %. U nastavení zrychlení je to jiné. Čím nižší zrychlení by bylo nastaveno, tím méně plynulý by byl aproximovaný pohyb, což je nežádoucí. U této varianty je důležité, aby se profil rychlosti v průběhu vykonávání pohybu nejlépe vůbec neměnil. Pokud by tedy bylo zrychlení malé, docházelo by při dojezdu na daný bod a při odjezdu z tohoto bodu ke snížení rychlosti prováděného pohybu. Experimentálně bylo ověřeno, že pro tuto variantu je nejvhodnější nastavit maximální zrychlení provádění pohybu (100 %).

V této variantě demonstrační aplikace lze ovládat pohyb kleští robota v jeho pracovním prostoru pomocí předvádění pohybu pravou rukou před senzorem Kinect V2. Pomocí levé ruky je ovládáno zahájení a ukončení záznamu pohybu. Záznam předváděného pohybu obstarává pouze serverová aplikace bez spuštěného robotického programu. Záznam předváděného pohybu je zahájen zvednutím levé ruky nad úroveň ramen. Během záznamu pohybu musí být levá ruka celou dobu nad úrovní ramen, protože pohybem levé ruky pod úroveň ramen dojde k ukončení záznamu předváděného pohybu. U ovládání pohybu i ovládání záznamu je slovem „ruka“ myšlena dlaň dané paže.

6.4 Serverová aplikace pro externí PC

Serverová aplikace pro externí PC musí zajišťovat obsluhu senzoru *Kinect V2* a zároveň komunikaci s řídicím systémem robota po síti Ethernet. Z toho vyplývá, že program musí pracovat se dvěma vlákny. Protože obsluhu senzoru *Kinect V2* je nejlépe provádět v programovacím jazyce `C#`, je tedy i celý program psaný v tomto programovacím jazyce.

Protože se jedná o aplikaci, která vyžaduje pouze minimální nastavení při spuštění bez dalších větších možností ovládání uživatelem za běhu, zvolil jsem standardní konzolovou aplikaci bez grafického uživatelského rozhraní. Grafické uživatelské rozhraní je použito na vizualizačním dotykovém operátorském panelu pro ovládání běhu demonstrační aplikace (zejména robotického programu) přes PLC, kde je naopak velmi vhodné. Tím se ale zabývá kapitola 6.6.

Zdrojový soubor serverové aplikace v jazyce `C#` je součástí projektu v Microsoft Visual Studio 2015 a z důvodu rozsahu se nachází pouze v elektronické příloze této práce.

V této kapitole se budu zabývat pouze hlavní funkcionalitou serverové aplikace, kromě obsluhy senzoru *Kinect V2*, které se věnovala kapitola 3. V zásadě se jedná o jednoduchou aplikaci spustitelnou v operačním systému *Windows* (kvůli obsluze senzoru *Kinect V2* pouze verze 8 a vyšší). Po spuštění serverové aplikace je zahájeno hledání připojeného senzoru *Kinect V2*. Pokud by nebyl detekován žádný připojený senzor *Kinect V2*, aplikace to uživateli oznámí a ukončí se. V tomto případě je potřeba ověřit správné připojení senzoru *Kinect V2* a opakovat spuštění serverové aplikace. Pokud by problém přetrvával, doporučuji podívat se do přílohy N, ve které jsou obsaženy některé chyby a jejich řešení. Pokud je senzor Kinect V2 správně detekován, aplikace pokračuje zobrazením seznamu dostupných síťových rozhraní Ethernet (síťové karty), a uživatel je vyzván k zadání čísla síťové karty, ke které je připojen řídicí systém robota. Bohužel není možné jednoduchým způsobem automaticky zjistit, ke které síťové kartě je řídicí systém robota připojen. Následuje výběr varianty demonstrační

aplikace. Uživatel je vyzván k zadání čísla varianty demonstrační aplikace. Číslo 1 pro *online* variantu s okamžitým prováděním předváděného pohybu, nebo číslo 2 pro *offline* variantu, u které robot opakuje pohyb až po jeho plném předvedení.

Pokud uživatel zvolí *online* variantu demonstrační aplikace, je následně pouze vyzván, aby spustil demonstrační aplikaci na operátorském panelu. Tím dojde k navázání spojení a započne vzájemná výměna dat. V tuto chvíli je celý systém připraven na okamžité vykonávání předváděného pohybu před senzorem *Kinect V2*. Pro ukončení této varianty demonstrační aplikace je zapotřebí ukončit aplikaci na operátorském panelu a následně stisknutím klávesy `'q'` na klávesnici externího PC, tím dojde k celkovému ukončení celé demonstrační aplikace. Pokud by ale uživatel chtěl pouze „pozastavit" provádění předváděného pohybu, je možné pouze ukončit aplikaci na operátorském panelu a serverovou aplikaci na externím PC nechat spuštěnou. V tomto případě nedoporučuji před ukončením serverové aplikace spouštět na operátorském panelu jinou aplikaci než tuto demonstrační aplikaci ve variantě *online*.

Pokud uživatel zvolí *offline* variantu demonstrační aplikace, je následně vyzván k výběru, zda bude chtít pohyb pro vykonání předvést před senzorem *Kinect V2*, nebo načíst ze souboru dříve předvedený pohyb. Při výběru první z možností je spuštěn senzor *Kinect V2* a aplikace čeká na zahájení záznamu pohybu (zvednutím levé ruky nad úroveň ramen). Po zahájení záznamu je předváděný pohyb zaznamenáván ve formě bodů do vnitřní proměnné serverové aplikace a zároveň ukládán do souboru pro budoucí načtení. Počet zaznamenaných bodů se při záznamu zobrazuje v okně aplikace, a počet těchto bodů je z technického důvodu řídicího systému robota omezen na 1000 bodů. Protože je ale zaznamenáván pouze každý desátý bod (při vzorkování senzoru *Kinect V2* 30 bodů/s), odpovídá 1000 zaznamenaných bodů přibližně 5-ti minutám a 33 sekundám záznamu předváděného pohybu, což si myslím, že bude pro většinu případů dostačující. Po ukončení záznamu pohybu (pohybem levé ruky pod úroveň ramen) je uživatel vyzván, aby spustil demonstrační aplikaci na operátorském panelu. Tím dojde k navázání spojení a přenesení všech dat potřebných pro realizaci pohybu robotem do řídicího systému robota. Serverová aplikace v tuto chvíli dokončila svojí činnost, a čeká na ukončení stisknutím klávesy `'q'` na klávesnici externího PC. To lze provést kdykoli po přenesení všech dat (signalizováno rozsvícením zelené kontrolky na operátorském panelu) do řídicího systému robota. Pro vlastní provádění pohybu robotem (popř. opakování pohybu) již není spuštěná serverová aplikace zapotřebí.

Pokud uživatel zvolí *offline* variantu demonstrační aplikace, a v následném výběru zvolí možnost načtení dříve předvedeného pohyb ze souboru, zobrazí se uživateli seznam očíslovaných dostupných souborů s dříve předvedenými uloženými pohyby. Uživatel je vyzván k zadání čísla souboru s uloženým pohybem, který chce načíst pro následné vykonání robotem. Podobně jako u předchozí varianty je uživatel vyzván, aby

spustil demonstrační aplikaci na operátorském panelu. Tím dojde k navázání spojení a přenesení všech dat potřebných pro realizaci pohybu robotem do řídicího systému robota. Serverová aplikace v tuto chvíli dokončila svojí činnost, a čeká na ukončení stisknutím klávesy `'q'` na klávesnici externího PC. To lze provést kdykoli po přenesení všech dat (signalizováno rozsvícením zelené kontrolky na operátorském panelu) do řídicího systému robota. Pro vlastní provádění pohybu robotem (popř. opakování pohybu) již není spuštěná serverová aplikace zapotřebí.

V hlavním vlákně spuštěné serverové aplikace běží uživatelská obsluha serverové aplikace s výpisy do konzole a obsluha senzoru *Kinect V2*. V určitých okamžicích (požadavek na navázání spojení s řídicím systémem robota) je spuštěno druhé vlákno pro komunikaci s řídicím systémem robota. V tomto vlákně dojde k vytvoření socketu podle nastavených parametrů a čekání na spojení od řídicího systému robota. Po navázání spojení dochází ke vzájemné komunikaci, která je vždy iniciována řídicím systémem robota (tzv. žádost o nová data). Pokud tedy serverová aplikace přijme zprávu od řídicího systému robota, zjistí v této zprávě aktuální hodnoty natočení kartézských os (`A, B, C`), a ty zkopíruje k souřadnicím `X, Y, Z` nového bodu. Pomocí XML šablony, těchto souřadnic a dalších odesílaných informací zformuluje zprávu pro odeslání řídicímu systému robota, kterou ihned odešle. Program v tomto vlákně dále čeká na další příchozí zprávu, nebo ukončení spojení ze strany řídicího systému robota. Šablona XML dat pro odeslání serverovou aplikací je uvedena v příloze G.

Při tvorbě této části aplikace jsem vycházel ze zdrojového kódu (v programovacím jazyce C#) ukázkové serverové aplikace dodávané společností KUKA Roboter GmbH k doplňku RSI ve starší verzi (V 2.3).

6.5 Externí spouštění robota pomocí nadřazeného PLC

Nadřazené PLC značky Siemens (typ S7-300) slouží k externímu řízení spouštění programů řídicího systému robota (pokud je přepnutý do režimu `AUT EXT`), sdílení informací s řídicím systémem robota a také pro hlídání bezpečnostního prostoru pomocí bezpečnostních prvků (světelná závora a laserový skener). Přístup k tomuto PLC a jeho programování je prostřednictvím vývojového prostředí Siemens TIA Portal.

Pro tuto práci jsem využil dostupný již vytvořený funkční projekt v prostředí Siemens TIA Portal, který jsem pouze doplnil o funkcionalitu potřebnou pro tuto demonstrační aplikaci. Tento kompletní projekt upravený pro tuto demonstrační aplikaci je součástí elektronické přílohy této bakalářské práce.

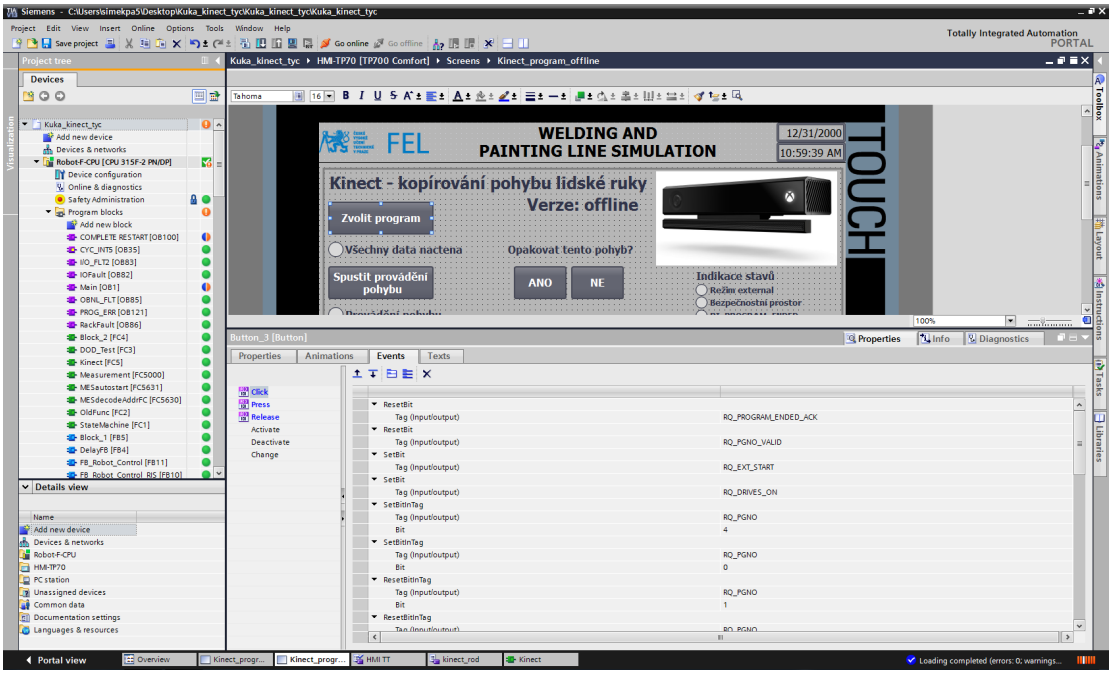
Moje úprava spočívá pouze ve vytvoření jednoho funkčního bloku (`Kinect [FC5]`), který je spouštěný v hlavním funkčním bloku (`Main [OB1]`). V tomto funkčním bloku jsou pomocí LAD diagramů definovány podmínky, které musejí být splněny pro nastavení daných výstupů (SET nebo RESET daného bitu). Těmito výstupy (jednotlivými bity) jsou tzv. „Tagy“, což jsou textové pojmenování bitů adres. Takže např. místo nepřehledného zápisu `%Q70.0` můžu psát `RQ_ProgramKinectRun`. Je zde pracováno se třemi druhy adres. Vstupy PLC mají adresy značené `‘I‘`, výstupy PLC mají adresy značení `‘Q‘` a poslední značení `‘M‘` je pro adresy míst v paměti (Marker). Seznam „Tagů“ a LAD diagramy, které jsou obsahem funkčního bloku `Kinect [FC5]`, jsou obsaženy v příloze J.

6.6 Uživatelské rozhraní pro ovládání demonstrační aplikace z operátorského panelu

Operátorský panel je stojan s plochou osazenou ovládacími a informačními operátorskými prvky. Tento panel je bohužel pro tuto demonstrační aplikaci umístěn nevhodně, je totiž na opačné straně bezpečnostního prostoru robota, než je umístěn externí PC s připojeným senzorem *Kinect V2* a řídicí systém robota s ovládacím panelem (ten je ale v případě potřeby možné přemístit k operátorskému panelu). Nejdůležitějšími prvky operátorského panelu je 3 barevný maják signalizující stav pracoviště robotického manipulátoru (červená barva signalizuje narušení bezpečnostního prostoru, oranžová spolu se zelenou signalizují volný bezpečnostní prostor a čekání na potvrzení potvrzovacím – bílým tlačítkem a zelená barva signalizuje připravené pracoviště k práci). Dále pak již zmíněné bílé potvrzovací tlačítko pro potvrzení uvolnění bezpečnostního prostoru robota, červené tlačítko „TOTAL STOP“ s aretací pro nouzové zastavení robotického manipulátoru (stejné tlačítko je také na ovládacím panelu řídicího systému robota) a dotykový vizualizační panel (Simatic HMI TP700 Comfort) s úhlopříčkou 7”, o kterém bude zbytek této kapitoly.

Dotykový vizualizační panel je přímo spojen s PLC, přes které je možné ovládat externí spouštění programů řídicího systému robota. Na pozadí tohoto vizualizačního panelu je spuštěn operační systém *Microsoft Windows CE 6.0*, ve kterém je spuštěna aplikace pro vizualizaci procesů a možnost uživatelského ovládání. Tato aplikace se skládá z jednotlivých obrazovek, mezi kterými je možné přepínat pomocí dolních dotykových tlačítek. Prostřednictvím dotykového tlačítka `Kinect Rod` je možné se dostat na rozdělovací obrazovku pro 2 různé demonstrační aplikace (této demonstrační aplikace se senzorem *Kinect V2* a demonstrační aplikace kolegy se světelnou tyčí – jiná bakalářská práce). Zvolením dotykového tlačítka `Online` nebo `Offline` se uživatel dostane na obrazovku ovládání zvolené varianty této demonstrační aplikace.

Konfigurace tohoto vizualizačního panelu, vytváření nových obrazovek s informačními a funkčními prvky se provádí, stejně jako programování PLC, v prostředí Siemens TIA Portal. V tomto prostředí je do převzatého a upraveného projektu, zmíněného již v kap. 6.5, přidána rozdělovací obrazovka pro 2 různé demonstrační aplikace a 2 obrazovky pro ovládání *online* a *offline* varianty této demonstrační aplikace. Na tyto nově vytvořené obrazovky, s využitím připravené šablony pro sjednocení vzhledu všech vytvářených obrazovek, se vkládají jednotlivé grafické prvky informačního a funkčního charakteru, jako např. textové popisky, obrázky, symboly, jednoduché tvary, tlačítka, pole pro zobrazení procesních dat, atd. U většiny těchto grafických prvků lze nastavit jejich barvu a viditelnost v závislosti na hodnotě zvoleného tagu. U funkčních grafických prvků lze pomocí hodnoty zvoleného tagu nastavit i povolení nebo zakázání funkce tohoto prvku a to nejdůležitější, např. tlačítka umožňují nastavení akcí po jejich stisknutí (akce je možné nastavit po stisknutí tlačítka, uvolnění tlačítka nebo nejběžnější je po kliknutí na tlačítko – po stisknutí a následném uvolnění). Nastavení akcí u jednoho tlačítka jako příklad je zobrazeno na obr. 6.3.



Obrázek 6.3. Nastavení akcí po stisknutí tlačítka na obrazovce vizualizačního panelu

Kombinací předchozích uvedených možností nastavení prvků jsou vytvořeny ovládací obrazovky této demonstrační aplikace. Tagy zde používané jsou vlastně PLC tagy přemapované pouze na vizualizační panel, ale všechny adresy zůstávají zachovány. Zobrazování stavů a povolení, či zakázání funkce tlačítek je prováděno prostřednictvím pomocných tagů. Zásadní roli zde ale hrají tagy vstupů a výstupů PLC, na které jsou přivedeny signály řídicího systému robota. Tyto signály jsou ze vstupů a výstupů řídicího systému robota (1 – 256) připojeny na vstupy a výstupy PLC prostřednictvím

Profinetu. Vstup řídicího systému robota č. 1 je v PLC dostupný na adrese `%Q70.0`. Podobně výstup řídicího systému robota je v PLC dostupný na adrese `%I70.0`. Pomocí těchto tagů určených pro externí ovládání běhu programů řídicího systému robota je možné robotické programy spouštět, řídit jejich běh a ukončovat je. Zvolení a spuštění programu se provede postupným nastavením výstupů. Nejdříve je potřeba nastavit výstupy `RQ_DRIVES_ON` a `RQ_EXT_START` na hodnotu TRUE. Poté pomocí 8-mi bitového signálu `RQ_PGNO` nastavíme požadované číslo programu. Zvolený program spustíme impulzem na výstupu `RQ_PGNO_VALID`. Po dokončení programu řídicí systém robota čeká na aktivování potvrzovacího signálu `RQ_PROGRAM_ENDED_ACK`. Po ukončení programu provádím také RESET všech používaných tagů, čímž uvedu systém do výchozího stavu pro spuštění dalšího programu. Snímky obrazovek vizualizačního panelu použité pro tuto demonstrační aplikaci jsou uvedené v příloze H a v příloze I. V přílohách K, L a M jsou uvedeny kompletní návody k obsluze demonstrační aplikace ve všech variantách.

Kapitola 7
Závěr

Podle zadání jsem se seznámil s možnostmi a programováním senzoru Kinect V2 a průmyslového robotu KUKA. Základní principy a možnosti těchto dvou systémů jsem se pokusil shrnout v kapitole 3 a 4.

Jedním ze základních úkolů této práce bylo navrhnout způsob komunikace senzoru Kinect V2 s řídicím systémem robota KUKA. Pro tento případ jsem navrhl komunikaci po síti Ethernet prostřednictvím paketů obsahující přenášená data v XML formě. Řídicí systém robota tuto komunikaci zpracovává pomocí softwarového doplňku KUKA Ethernet KRL, který ale bohužel není určený pro řízení pohybů robota v reálném čase. Jinou variantou by bylo použití softwarového doplňku KUKA RSI s podobnou formou komunikace také po síti Ethernet, který podporuje řízení pohybů robota v reálném čase. Tento doplněk pro tuto práci nebyl použit.

Nejdůležitějším úkolem této práce bylo vytvořit demonstrační aplikaci, na které by bylo možné ukázat principy spojení různých systémů v jeden kompletní celek. Tuto demonstrační aplikaci jsem úspěšně vytvořil, a to ve dvou variantách. První varianta je určena pro okamžité vykonávání pohybu, předváděného osobou před senzorem Kinect V2, robotem s možností ovládat otevření kleští, což umožňuje přemisťování předmětů v pracovním prostoru robota. Druhá varianta je určena pro vykonání pohybu, který byl celý předveden před samotným vykonáním robotem. U této varianty nezáleží na době mezi předvedením pohybu před senzorem Kinect V2 a jeho provedením robotem.

Důvod, proč vznikly dvě varianty demonstrační aplikace je v používání softwarového doplňku KUKA Ethernet KRL, který není určen pro řízení v reálném čase pro aplikaci okamžitého provádění předváděného pohybu. Pro ukázku funkčnosti tato varianta postačuje, ale pohyb je bohužel lehce trhaný. Tento problém byl vyřešen ve druhé variantě demonstrační aplikace tím, že jsou před vykonáním pohybu robotem všechna data přenesena do řídicího systému robota. Pokud má robot všechna potřebná data pro běh programu je schopný naplánovat trajektorii pohybu přes zadané body aproximovaně, tzn. je schopný trajektorii projet plynule, aniž by se v jednotlivých bodech zastavil.

Umožnit plynulé projetí trajektorie zadávané pomocí bodů aktuálně získávaných ze senzoru Kinect V2 by mělo být možné využitím softwarového doplňku KUKA RSI. Bohužel tuto možnost nebylo možné z provozních a časových důvodů vyzkoušet. Jinou

variantou, která mě v této souvislosti napadla, je využít stávajícího softwarového doplňku KUKA Ethernet KRL s oddělením řízení komunikace od provádění pohybu. Pokud by komunikace probíhala v jiném programu běžícím na pozadí (například v programu sps.SUB) a provádění pohybu bylo řízeno ze spuštěného hlavního programu, myslím si, že by byl řídicí systém robota v tomto případě schopen plánovat trajektorii pohybu aproximovaně jako ve druhé variantě demonstrační aplikace. Při tomto řešení by sice prováděný pohyb byl zpožděný o dva body zaslané senzorem Kinect, ale při zasílání dostatečného počtu souřadnic nových bodů za jednotku času by toto zpoždění nemuselo být lidským zrakem postřehnutelné. Největším problémem tohoto řešení je synchronizace hlavního programu provádějící pohyb robota s programem řídicím komunikaci na pozadí. Jelikož úkolem této práce bylo pouze navrhnout funkční způsob komunikace a realizace této varianty by byla složitá a časově náročná, do této práce jsem ji nezahrnul.

Závěrem tedy můžu říci, že jsem vytvořil kompletní demonstrační aplikaci, která spojuje obrazový senzor Kinect V2, externí počítač, řídicí systém robota, nadřazené PLC a vizualizační operátorský panel. Obrazový senzor Kinect V2 připojený k externímu PC je obsluhován serverovou aplikací spuštěnou na externím PC, která zároveň zajišťuje komunikaci mezi externím PC a řídicím systémem. V řídicím systému robota se nacházejí robotické programy pro řízení komunikace s externím PC a prováděním pohybu robota. Spouštění těchto programů a řízení jejich běhu je umožněno pomocí vizualizačního operátorského panelu připojeného k nadřazenému PLC, které komunikuje s řídicím systémem robota po síti Profinet. Kompletní demonstrační aplikaci tedy tvoří celek programů, serverové aplikace, konfiguračních souborů a projektu v prostředí Siemens TIA Portal. Všechny tyto součásti demonstrační aplikace jsou v elektronické příloze této bakalářské práce.

Literatura

[1] *Kinect pro Xbox One* [online]. Microsoft – XBOX. [vid. 19.5.2016]. Dostupné z: http://www.xbox.com/cs-CZ/xbox-one/accessories/kinect-for-xbox-one

[2] *Kinect Adapter for Windows* [online]. Microsoft. [vid. 19.5.2016]. Dostupné z: http://www.microsoftstore.com/store/msusa/en_US/pdp/Kinect-Adapter-for-Windows/productID.308803600

[3] *JointType Enumeration* [online]. Microsoft. [vid. 19.5.2016]. Dostupné z: https://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx

[4] *Xbox One (Durango) Next-Generation Kinect Sensor* [online]. VGLeaks. [vid. 19.5.2016]. Dostupné z: http://vgleaks.com/durango-next-generation-kinect-sensor/

[5] *Kinect For Windows v2* [online]. iNFINITE Production. [vid. 19.5.2016]. Dostupné z: http://www.infinite.cz/blog/kinect-v2

[6] Kinect. In: *Wikipedia: the free encyclopedia* [online]. Wikimedia Foundation, 2003. Stránka naposledy edit. 18.5.2016 v 21:36. [vid. 19.5.2016]. Anglická verze. Dostupné z: https://en.wikipedia.org/wiki/Kinect

[7] Kinect for Xbox One. In: *Wikipedia: the free encyclopedia* [online]. Wikimedia Foundation, 2003. Stránka naposledy edit. 5.3.2016 v 23:52. [vid. 19.5.2016]. Anglická verze. Dostupné z: https://en.wikipedia.org/wiki/Kinect_for_Xbox_One

[8] *Xbox One: technologický rozbor nového hardwaru* [online]. Živě.cz. [vid. 19.5.2016]. Dostupné z: http://www.zive.cz/clanky/xbox-one-technologicky-rozbor-noveho-hardwaru/vsudypritomny-kinect-20/sc-3-a-168957-ch-86935/

[9] *KR 5 ARC* [online]. KUKA průmyslové roboty. [vid. 19.5.2016]. Dostupné z: http://www.kuka-robotics.com/czech_republic/cs/products/industrial_robots/low/kr5_arc/start.htm

[10] KUKA Robot Group. [PDF] *KR 5 ARC, Operating Instructions*. 03/2011. [vid. 19.5.2016]. Dostupné z: elektronická příloha této práce (CD).

[11] KUKA Roboter GmbH. [PDF] *KUKA System Software 8.2, Návod k obsluze a programování pro konečné uživatele*. 07/2012. [vid. 19.5.2016]. Dostupné z: elektronická příloha této práce (CD).

[12] KUKA Roboter GmbH. [PDF] *KR C4, Operating Instructions*. 05/2012. [vid. 19.5.2016]. Dostupné z: elektronická příloha této práce (CD).

[13] KUKA Roboter GmbH. [PDF] *KUKA System Software 8.2, Operating and Programming Instructions for System Integrators*. 07/2012. [vid. 19.5.2016]. Dostupné z: elektronická příloha této práce (CD).

[14] KUKA Roboter GmbH. [PDF] *KUKA.Ethernet KRL 2.1, For KUKA System Software 8.2*. 03/2012. [vid. 19.5.2016]. Dostupné z: elektronická příloha této práce (CD).

[15] *Kinect for Windows SDK 2.0* [online]. Microsoft. [vid. 19.5.2016]. Dostupné z: `https://www.microsoft.com/en-us/download/details.aspx?id=44561`

[16] *Kinect for Windows version 2: Body tracking* [online]. Vangos Pterneas. [vid. 18.5.2016]. Dostupné z: `http://pterneas.com/2014/03/13/kinect-for-windows-version-2-body-tracking/`

[17] *Kinect for Windows SDK* [online]. Microsoft. [vid. 18.5.2016]. Dostupné z: `https://msdn.microsoft.com/en-us/library/dn799271.aspx`

[18] *Coordinate mapping* [online]. Microsoft. [vid. 18.5.2016]. Dostupné z: `https://msdn.microsoft.com/en-us/library/dn785530.aspx`

[19] KUKA Roboter GmbH. [PDF] *KR C2 / KR C3 Expert Programming*. 09/2003. [vid. 19.5.2016]. Dostupné z: elektronická příloha této práce (CD).

[20] KUKA Roboter GmbH. [PDF] *KUKA.RobotSensorInterface 3.1 for KUKA System Software 8.2*. 12/2010. [vid. 19.5.2016]. Dostupné z: elektronická příloha této práce (CD).

Příloha A
Zkratky použité v této práci

PC Personal Computer, v překladu osobní počítač, je běžně používaná zkratka pro dnešní běžné stolní počítače používané v kancelářích a domácnostech.

USB Universal Serial Bus, je univerzální sériová sběrnice používaná pro připojení periferií k počítači.

HD High Definition, je označení pro kvalitu snímků videa – rozlišení.

SDK Software development kit, je sada vývojových nástrojů umožňující vytváření aplikací.

KRC KUKA Robot Controller, je zkratka označující řídicí systém robota KUKA.

PLC Programmable Logic Controller, je průmyslový počítač využívaný pro řízení průmyslových procesů.

KSS KUKA System Software, je zkratka označující verzi softwaru řídicího systému robota.

KRL KUKA Robot Language je programovací jazyk pro tvorbu robotických programů v řídicích systémech KUKA. V této práci je KRL program také nazýván robotickým programem nebo SRC souborem, všechna tato pojmenování jsou synonyma.

XML Extensible Markup Language, je značkovací jazyk pro formulaci dat pro přenos mediem.

TCP/IP Transmission Control Protocol/Internet Protocol, je rodina protokolů obsahující sadu protokolů pro komunikaci v síti Ethernet.

UDP User Datagram Protocol, je protokol sítě Ethernet pro nezabezpečený přenos datagramů. Je mnohem jednodušší než protokol TCP z rodiny protokolů TCP/IP.

TCP Tool Center Point, je pracovní bod nástroje nastavený v řídicím systému robota pro daný nástroj.

PTP Point To Point, označuje druh pohybu od bodu k bodu po nejkratší dráze.

LIN Linear, označuje druh pohybu od bodu k bodu po přímce.

CIRC Circular, označuje kruhový druh pohybu.

SRC Je zdrojový soubor robotického programu, který je psaný v programovacím jazyce KRL, nebo vytvářený pomocí in-line formulářů. V této práci

je SRC soubor také nazýván robotickým programem nebo KRL programem, všechna tato pojmenování jsou synonyma.

DAT Je datový soubor robotického programu, který obsahuje deklarace proměnných a souřadnice bodů. Spolu se SRC souborem tvoří kompletní robotický program.

RSI Robot Sensor Interface, je zkratka pro název softwarového doplňku KUKA.RobotSensorInterface.

TIA Portal Totally Integrated Automation Portal, je označení vývojového prostředí společnosti Siemens pro průmyslové řídicí systémy.

LAD Ladder logic je grafický programovací jazyk používaný pro programování PLC zařízení. Původně byl používaný pro schémata reléové logiky.

Příloha B
KRL program demonstrační aplikace – online varianta

Náplní této přílohy je zdrojový kód robotického programu v KRL jazyce pro *online* variantu demonstrační aplikace „Robotické provádění pohybu snímaného senzorem Kinect V2“. Tento kód se nachází v souboru `KinectAplikaceOnline.SRC`, který je umístěn v řídicím systému robota v umístění `KRC:\R1\Program`, popř. nějaké podsložce (v době odevzdání se nachází v podsložce `\dastyluk`).

```
1  &ACCESS RVO1
2  &REL 144
3  &PARAM SensorITMASK = *
4  &PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
5  &PARAM EDITMASK = *
6  DEF KinectAplikaceOnline( )
7
8  ;FOLD Deklarace
9    DECL EKI_STATUS RET
10 ;ENDFOLD (Deklarace)
11
12 ;FOLD XML sablona dat pro komunikaci po Ethernetu
13   ;FOLD Prijimane od externiho systemu
14     ; ⟨Sensor⟩
15       ; ⟨Pozice⟩
16         ; ⟨NoveSouradnice X="..." Y="..." Z="..." A="..." B="..." C="..." /⟩
17       ; ⟨/Pozice⟩
18       ; ⟨Kleste⟩
19         ; ⟨Otevrit⟩0⟨/Otevrit⟩
20       ; ⟨/Kleste⟩
21       ; ⟨Status⟩
22         ; ⟨KinectPripojen⟩0⟨/KinectPripojen⟩
23       ; ⟨/Status⟩
24     ; ⟨/Sensor⟩
25   ;ENDFOLD (Prijimane od externiho systemu)
26
27   ;FOLD Odesilane externimu systemu
28     ; ⟨Robot⟩
29       ; ⟨Data⟩
30         ; ⟨PoziceTCPRobota A="..." B="..." C="..." X="..." Y="..." Z="..." /⟩
31       ; ⟨/Data⟩
32     ; ⟨/Robot⟩
33   ;ENDFOLD (Odesilane externimu systemu)
34 ;ENDFOLD (XML sablona dat pro komunikaci po Ethernetu)
35
36 ;FOLD INI
```

```
37    ;FOLD BASISTECH INI
38      BAS (#INITMOV,0 )
39    ;ENDFOLD (BASISTECH INI)
40    ;FOLD USER INI
41      ;Make your modifications here
42    ;ENDFOLD (USER INI)
43  ;ENDFOLD (INI)
44
45  ;FOLD Definice interruptu pro prichozi data
46    INTERRUPT DECL 89 WHEN $FLAG[998]==TRUE DO NactiData()
47    INTERRUPT ON 89
48  ;ENDFOLD (Definice interruptu pro prichozi data)
49
50  ;FOLD Inicializace
51    kinectFrame={X 1000.0,Y -340.0,Z 940.0,A 90.0,B 0.0,C 90.0} ;pro pre-
52      dani novych souradnic pro vykonani pohybu
53    kinectPripojen=FALSE      ;informace o funkcnim pripojenem senzoru
54      Kinect k externimu PC (informace prijde po Ethernetu)
55    spojeniNavazano = FALSE   ;informace pro PLC, zda bylo spojeni
56      navazano nebo ne
57    provadeniPohybu=FALSE     ;signalizuje beh hlavni smycky
58    otevritKleste=TRUE        ;od ext. systemu zda maji byt kleste
59      otevreny nebo zavreny (TRUE=otevreny, FALSE=zavreny)
60    klesteOtevreny=si_gr_open ;info od klesti pro PLC a rizeni behu
61      programu
62    KlesteZavreny=si_gr_close ;info od klesti pro PLC a rizeni behu
63      programu
64    $FLAG[1]=FALSE            ;true=data z Eth. spojeni jsou nactena
65    ;programKinectRun = spusteni hlavni smycky robotickeho programu pro
66      spolupraci s Kinectem, od PLC (vizualizace)
67    gr_opn()                  ;volani makra pro otevreni klesti
68  ;ENDFOLD (Inicializace)
69
70  ;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,
71      %P 1:PTP, 2:HOME, 3:, 5:100, 7:DEFAULT
72    $BWDSTART = FALSE
73    PDAT_ACT=PDEFAULT
74    FDAT_ACT=FHOME
75    BAS (#PTP_PARAMS,100 )
76    $H_POS=XHOME1
77    PTP  XHOME1
78  ;ENDFOLD
79
80  ;FOLD Navazani ethernetoveho spojeni
81    RET=EKI_Init("KinectAplikace")
82    RET=EKI_Open("KinectAplikace")
83    IF RET.Connected==FALSE THEN
84      spojeniNavazano=FALSE; info pro PLC - chyba spojeni
85      GOTO konec
86    ENDIF
87    spojeniNavazano=TRUE
88  ;ENDFOLD (Navazani ethernetoveho spojeni)
89
```

```
90  ;FOLD Hlavni smycka programu - vykonavani pohybu na zaklade dat ze
91      senzoru Kinect
92    ; nacteni prvni souradnice a stavu senzoru Kinect - inicializace
93    ZadostOData()
94
95    BAS(#VEL_PTP,100)  ;nastaveni rychlosti na 100%
96    BAS(#ACC_PTP,15)   ;nastaveni zrychleni na  15%
97
98    WAIT FOR programKinectRun
99    provadeniPohybu=TRUE
100   WHILE programKinectRun==TRUE  ;info od PLC
101     IF kinectPripojen == FALSE THEN  ;info od ext. sys.
102       EXIT
103     ENDIF
104     ; provedu pohyb PTP ze souradnic kinectFrame
105     PTP kinectFrame
106
107     ; ovladani klesti pomoci informace ziskane ze senzoru Kinect
108     IF ((otevritKleste==TRUE) AND (klesteZavreny==TRUE)) THEN
109       gr_opn()  ;volani makra pro otevreni klesti
110       klesteZavreny=FALSE
111       klesteOtevreny=TRUE
112     ENDIF
113     IF ((otevritKleste==FALSE) AND (klesteOtevreny==TRUE)) THEN
114       gr_cls()  ;volani makra pro zavreni klesti
115       klesteOtevreny=FALSE
116       klesteZavreny=TRUE
117     ENDIF
118
119     ZadostOData()  ;zadost o dalsi novou souradnici (data ze senzoru)
120   ENDWHILE
121   provadeniPohybu=FALSE
122 ;ENDFOLD (Hlavni smycka programu - vykonavani pohybu na zaklade dat ze
123   senzoru Kinect)
124
125 ;FOLD Ukonceni ethernetove komunikace
126   RET=EKI_Close("KinectAplikace")
127   RET=EKI_Clear("KinectAplikace")
128   INTERRUPT OFF 89
129 ;ENDFOLD (Ukonceni ethernetove komunikace)
130
131 ;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,
132     %P 1:PTP, 2:HOME, 3:, 5:100, 7:DEFAULT
133   $BWDSTART = FALSE
134   PDAT_ACT=PDEFAULT
135   FDAT_ACT=FHOME
136   BAS (#PTP_PARAMS,100 )
137   $H_POS=XHOME1
138   PTP  XHOME1
139 ;ENDFOLD
140
141 ;FOLD Ukonceni programu
142   konec:  ;navesti pro GOTO pri nenavazani spojeni
```

```
143     spojeniNavazano=FALSE
144     kinectPripojen=FALSE
145   ;ENDFOLD (Ukonceni programu)
146
147   END
148
149   DEF ZadostOData()
150
151     ;FOLD Deklarace
152       DECL EKI_STATUS RET
153     ;ENDFOLD (Deklarace)
154
155     ;FOLD Odeslani dat robota externimu systemu
156       ; zapsani dat robota do bufferu ethernetoveho spojeni - priprava
157         pred odeslanim
158       RET=EKI_SetFrame("KinectAplikace","Robot/Data/PoziceTCPRobota",
159         $POS_ACT)
160       ; odeslani dat robota externimu systemu
161       RET=EKI_Send("KinectAplikace","Robot")
162     ;ENDFOLD (Odeslani dat robota externimu systemu)
163
164     ;FOLD Cekani dokud nejsou vsechna prichozi data nactena
165       WAIT FOR $FLAG[1]
166       $FLAG[1] = FALSE
167     ;ENDFOLD (Cekani dokud nejsou vsechna prichozi data nactena)
168
169   END
170
171   DEF NactiData()
172
173     ;FOLD Deklarace
174       DECL EKI_STATUS RET
175     ;ENDFOLD (Deklarace)
176
177     ;FOLD Nacteni prijatych dat od externiho systemu
178       RET=EKI_GetFrame("KinectAplikace","Sensor/Pozice/NoveSouradnice",
179         kinectFrame)
180       RET=EKI_GetBool("KinectAplikace","Sensor/Kleste/Otevrit",
181         otevritKleste)
182       RET=EKI_GetBool("KinectAplikace","Sensor/Status/KinectPripojen",
183         kinectPripojen)
184     ;ENDFOLD (Nacteni prijatych dat od externiho systemu)
185
186     ;FOLD Nastaveni flagu pro dalsi beh
187       $FLAG[998]=FALSE
188       $FLAG[1]=TRUE
189     ;ENDFOLD (Nastaveni flagu pro dalsi beh)
190
191   END
```

Příloha C
KRL program demonstrační aplikace – offline varianta

Náplní této přílohy je zdrojový kód robotického programu v KRL jazyce pro *offline* variantu demonstrační aplikace „Robotické provádění pohybu snímaného senzorem Kinect V2“. Tento kód se nachází v souboru `KinectAplikaceOfline.SRC`, který je umístěn v řídicím systému robota v umístění `KRC:\R1\Program`, popř. nějaké podsložce (v době odevzdání se nachází v podsložce `\dastyluk`).

```
&ACCESS RVO1
&REL 158
&PARAM SensorITMASK = *
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM EDITMASK = *
DEF KinectAplikaceOffline( )

;FOLD Deklarace
  DECL EKI_STATUS RET
;ENDFOLD (Deklarace)

;FOLD XML sablona dat pro komunikaci po Ethernetu
  ;FOLD Prijimane od externiho systemu
    ; ⟨Sensor⟩
      ; ⟨Pozice⟩
        ; ⟨NoveSouradnice X="..." Y="..." Z="..." A="..." B=".." C="..." /⟩
      ; ⟨/Pozice⟩
      ; ⟨Kleste⟩
        ; ⟨Otevrit⟩0⟨/Otevrit⟩
      ; ⟨/Kleste⟩
      ; ⟨Status⟩
        ; ⟨KinectPripojen⟩0⟨/KinectPripojen⟩
      ; ⟨/Status⟩
    ; ⟨/Sensor⟩
  ;ENDFOLD (Prijimane od externiho systemu)

  ;FOLD Odesilane externimu systemu
    ; ⟨Robot⟩
      ; ⟨Data⟩
        ; ⟨PoziceTCPRobota A="..." B="..." C="..." X="..." Y="..." Z="..." /⟩
      ; ⟨/Data⟩
    ; ⟨/Robot⟩
  ;ENDFOLD (Odesilane externimu systemu)
;ENDFOLD (XML sablona dat pro komunikaci po Ethernetu)

;FOLD INI
```

```
37      ;FOLD BASISTECH INI
38        BAS (#INITMOV,0 )
39      ;ENDFOLD (BASISTECH INI)
40      ;FOLD USER INI
41        ;Make your modifications here
42      ;ENDFOLD (USER INI)
43    ;ENDFOLD (INI)
44
45    ;FOLD Definice interruptu pro zastaveni vykonavani pohybu
46      INTERRUPT DECL 88 WHEN programKinectStop==TRUE DO
47        ZastaveniPohybuPreruseni()
48      INTERRUPT ON 88
49    ;ENDFOLD (Definice interruptu pro zastaveni vykonavani pohybu)
50
51    ;FOLD Definice interruptu pro prichozi data
52      INTERRUPT DECL 89 WHEN $FLAG[998]==TRUE DO NactiData()
53      INTERRUPT ON 89
54    ;ENDFOLD (Definice interruptu pro prichozi data)
55
56    ;FOLD Inicializace
57      tempFrame={X 1000.0,Y -340.0,Z 940.0,A 90.0,B 0.0,C 90.0} ;pro pre-
58        dani novych souradnic pro vykonani pohybu
59      kinectPripojen=TRUE       ;informace o funkcnim pripojenem senzoru
60        Kinect k externimu PC (informace prijde po Ethernetu)
61      pocetBodu=0               ;pocet nactenych bodu pres Ethernet
62      spojeniNavazano = FALSE   ;informace pro PLC, zda bylo spojeni
63        navazano nebo ne
64      provadeniPohybu=FALSE     ;signalizuje beh hlavni smycky
65      dataNactena=FALSE
66      opakovatPohyb=FALSE       ;dotaz k PLC zda se ma pohyb z nactenych
67        bodu opakovat
68      $FLAG[1]=FALSE            ;true=data z Eth. spojeni jsou nactena
69      ;programKinectRun = spusteni hlavni smycky robotickeho programu pro
70        spolupraci s Kinectem, od PLC (vizualizace)
71      kinectTemp=1              ;pracovni promenna typu integer (pro cykly)
72      FOR kinectTemp = 1 TO 1000  ;inicializace pole framu slouziciho pro
73          ulozeni prijatych souradnic bodu
74        poleKinectFrame[kinectTemp]={X 1000.0,Y -340.0,Z 940.0,
75                                     A 90.0,  B 0.0,   C 90.0}
76      ENDFOR
77    ;ENDFOLD (Inicializace)
78
79    ;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,
80       %P 1:PTP, 2:HOME, 3:, 5:100, 7:DEFAULT
81      $BWDSTART = FALSE
82      PDAT_ACT=PDEFAULT
83      FDAT_ACT=FHOME
84      BAS (#PTP_PARAMS,100 )
85      $H_POS=XHOME1
86      PTP  XHOME1
87    ;ENDFOLD
88
89    ;FOLD Navazani ethernetove spojeni
```

```
90    RET=EKI_Init("KinectAplikace")
91    RET=EKI_Open("KinectAplikace")
92    IF RET.Connected==FALSE THEN
93      spojeniNavazano=FALSE; info pro PLC - chyba spojeni
94      GOTO konec
95    ENDIF
96    spojeniNavazano=TRUE
97  ;ENDFOLD (Navazani ethernetove spojeni)
98
99  ;FOLD Nacteni vsech souradnic pro nasledne vykonani
100   NacteniVsechDat()
101 ;ENDFOLD (Nacteni vsech souradnic pro nasledne vykonani)
102
103 ;FOLD Hlavni smycka programu - vykonavani pohybu na zaklade dat ze
104     senzoru Kinect
105   vykonatZnovu: ;navesti pro zopakovani pohybu z nactenych souradnic
106   BAS(#VEL_PTP,30)   ;nastaveni rychlosti na  30%
107   BAS(#ACC_PTP,100)  ;nastaveni zrychleni na 100%
108
109   WAIT FOR programKinectRun
110   provadeniPohybu=TRUE
111   ; provedu pohyb PTP ze souradnic kinectFrame
112   kinectTemp=1
113   FOR kinectTemp=1 TO pocetBodu STEP 1
114     PTP poleKinectFrame[kinectTemp] C_PTP
115   ENDFOR
116   provadeniPohybu=FALSE
117 ;ENDFOLD (Hlavni smycka programu - vykonavani pohybu na zaklade dat ze
118   senzoru Kinect)
119
120 ;FOLD PTP HOME  Vel= 100 % DEFAULT;%{PE}%MKUKATPBASIS,%CMOVE,%VPTP,
121     %P 1:PTP, 2:HOME, 3:, 5:100, 7:DEFAULT
122   $BWDSTART = FALSE
123   PDAT_ACT=PDEFAULT
124   FDAT_ACT=FHOME
125   BAS (#PTP_PARAMS,100 )
126   $H_POS=XHOME1
127   PTP  XHOME1
128 ;ENDFOLD
129
130 ;FOLD Moznost opakovani pohybu z nactenych souradnic
131   opakovatPohyb=TRUE ;dotaz k PLC zda se ma pohyb z nactenych
132     bodu opakovat
133   WAIT FOR ((opakovatPohybANO==TRUE) OR (opakovatPohybNE==TRUE))
134   IF opakovatPohybANO==TRUE THEN
135     opakovatPohyb=FALSE
136     GOTO vykonatZnovu
137   ENDIF
138   opakovatPohyb=FALSE
139 ;ENDFOLD (Moznost opakovani pohybu z nactenych souradnic)
140
141 ;FOLD Ukonceni programu
142   konec:  ;navesti pro GOTO pri nenavazani spojeni
```

```
143     spojeniNavazano=FALSE
144     kinectPripojen=FALSE
145     dataNactena=FALSE
146   ;ENDFOLD (Ukonceni programu)
147
148   END
149
150   DEF NacteniVsechDat()
151
152     ;FOLD Deklarace
153       DECL EKI_STATUS RET
154     ;ENDFOLD (Deklarace)
155
156     ;FOLD Nacteni dat od externiho systemu
157       pocetBodu=0
158       WHILE kinectPripojen==TRUE
159         ZadostOData()
160         pocetBodu=pocetBodu+1
161         poleKinectFrame[pocetBodu]=tempFrame
162       ENDWHILE
163     ;ENDFOLD (Nacteni dat od externiho systemu)
164
165     ;FOLD Ukonceni ethernetove komunikace
166       RET=EKI_Close("KinectAplikace")
167       RET=EKI_Clear("KinectAplikace")
168       INTERRUPT OFF 89
169       dataNactena=TRUE
170     ;ENDFOLD (Ukonceni ethernetove komunikace)
171
172   END
173
174   DEF ZadostOData()
175
176     ;FOLD Deklarace
177       DECL EKI_STATUS RET
178     ;ENDFOLD (Deklarace)
179
180     ;FOLD Odeslani dat robota externimu systemu
181       ; zapsani dat robota do bufferu ethernetoveho spojeni - priprava
182         pred odeslanim
183       RET=EKI_SetFrame("KinectAplikace","Robot/Data/PoziceTCPRobota",
184         $POS_ACT)
185       ; odeslani dat robota externimu systemu
186       RET=EKI_Send("KinectAplikace","Robot")
187     ;ENDFOLD (Odeslani dat robota externimu systemu)
188
189     ;FOLD Cekani dokud nejsou vsechna prichozi data nactena
190       WAIT FOR $FLAG[1]
191       $FLAG[1] = FALSE
192     ;ENDFOLD (Cekani dokud nejsou vsechna prichozi data nactena)
193
194   END
195
```

```
196 DEF NactiData()
197
198   ;FOLD Deklarace
199     DECL EKI_STATUS RET
200   ;ENDFOLD (Deklarace)
201
202   ;FOLD Nacteni prijatych dat od externiho systemu
203     RET=EKI_GetFrame("KinectAplikace","Sensor/Pozice/NoveSouradnice",
204       tempFrame)
205     RET=EKI_GetBool("KinectAplikace","Sensor/Status/KinectPripojen",
206       kinectPripojen)
207     RET=EKI_GetBool("KinectAplikace","Sensor/Kleste/Otevrit",
208       otevritKleste) ;vycteni hodnoty pouze pro vyprazdneni
209         bufferu robota
210   ;ENDFOLD (Nacteni prijatych dat od externiho systemu)
211
212   ;FOLD Nastaveni flagu pro dalsi beh
213     $FLAG[998]=FALSE
214     $FLAG[1]=TRUE
215   ;ENDFOLD (Nastaveni flagu pro dalsi beh)
216
217 END
218
219 DEF ZastaveniPohybuPreruseni()
220
221   ;FOLD Zastaveni provadeneho pohybu pres PLC
222     kinectTemp=pocetBodu
223   ;ENDFOLD (Zastaveni provadeneho pohybu pres PLC)
224
225 END
```

Příloha D
Datový soubor ke KRL programu demonstrační aplikace

Náplní této přílohy je zdrojový kód datového souboru ke KRL programu demonstrační aplikace „Robotické provádění pohybu snímaného senzorem Kinect V2“. Tento kód je stejný pro *online* i *offline* variantu demonstrační aplikace. Pro *online* variantu se nachází v souboru `KinectAplikaceOnline.dat` a pro *offline* variantu se nachází v souboru `KinectAplikaceOffline.dat`. Oba tyto soubory jsou umístěny v řídicím systému robota v umístění `KRC:\R1\Program`, ve stejné podsložce jako soubor `KinectAplikaceOnline.SRC` a `KinectAplikaceOffline.SRC` (v době odevzdání se nachází v podsložce `\dastyluk`).

```
1  &ACCESS RVO1
2  &REL 123
3  &PARAM SensorITMASK = *
4  &PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
5  &PARAM EDITMASK = *
6  DEFDAT  KinectAplikace PUBLIC
7    ;FOLD EXTERNAL DECLARATIONS;%{PE}%MKUKATPBASIS,%CEXT,%VCOMMON,%P
8      ;FOLD BASISTECH EXT;%{PE}%MKUKATPBASIS,%CEXT,%VEXT,%P
9        EXT  BAS (BAS_COMMAND  :IN,REAL  :IN )
10       DECL INT SUCCESS
11     ;ENDFOLD (BASISTECH EXT)
12     ;FOLD USER EXT;%{E}%MKUKATPUSER,%CEXT,%VEXT,%P
13       ;Make your modifications here
14     ;ENDFOLD (USER EXT)
15   ;ENDFOLD (EXTERNAL DECLARATIONS)
16 ENDDAT
```

Příloha E
Konfigurační XML soubor ethernetového spojení

Náplní této přílohy je konfigurační XML soubor ethernetového spojení. Tento kód se nachází v souboru `KinectAplikace.XML`, který je umístěn v řídicím systému robota v umístění `C:\KRC\ROBOTER\Config\user\Common\EthernetKRL`.

```
1  ⟨ETHERNETKRL⟩
2    ⟨CONFIGURATION⟩
3      ⟨EXTERNAL⟩
4        ⟨TYPE⟩Server⟨/TYPE⟩
5        ⟨IP⟩192.168.136.123⟨/IP⟩
6        ⟨PORT⟩6008⟨/PORT⟩
7      ⟨/EXTERNAL⟩
8      ⟨INTERNAL⟩
9        ⟨ALIVE Set˙Out="249"/⟩
10       ⟨PROTOCOL⟩TCP⟨/PROTOCOL⟩
11     ⟨/INTERNAL⟩
12   ⟨/CONFIGURATION⟩
13   ⟨RECEIVE⟩
14     ⟨XML⟩
15       ⟨ELEMENT Tag="Sensor/Pozice/NoveSouradnice" Type="FRAME" /⟩
16       ⟨ELEMENT Tag="Sensor/Kleste/Otevrit" Type="BOOL" /⟩
17       ⟨ELEMENT Tag="Sensor/Status/KinectPripojen" Type="BOOL" /⟩
18       ⟨ELEMENT Tag="Sensor" Set˙Flag="998" /⟩
19     ⟨/XML⟩
20   ⟨/RECEIVE⟩
21   ⟨SEND⟩
22     ⟨XML⟩
23       ⟨ELEMENT Tag="Robot/Data/PoziceTCPRobota/@X" /⟩
24       ⟨ELEMENT Tag="Robot/Data/PoziceTCPRobota/@Y" /⟩
25       ⟨ELEMENT Tag="Robot/Data/PoziceTCPRobota/@Z" /⟩
26       ⟨ELEMENT Tag="Robot/Data/PoziceTCPRobota/@A" /⟩
27       ⟨ELEMENT Tag="Robot/Data/PoziceTCPRobota/@B" /⟩
28       ⟨ELEMENT Tag="Robot/Data/PoziceTCPRobota/@C" /⟩
29     ⟨/XML⟩
30   ⟨/SEND⟩
31 ⟨/ETHERNETKRL⟩
```

Příloha F
Uživatelsky definovaná část souboru $config.DAT

Náplní této přílohy je část souboru `$config.DAT`, pomocí které je možné deklarovat globální proměnné. Tento soubor se nachází v řídicím systému robota v umístění `KRC:\R1\System`. Tento soubor je braný jako systémový s oblastí (na konci souboru), která je určena pro uživatelskou deklaraci globálních proměnných s platností na všechny programy a podprogramy umístěné v `KRC:\R1` a všech podsložkách.

```
1  &ACCESS  RV
2  &PARAM DISKPATH = SYSTEM
3  &REL 44
4  DEFDAT  $CONFIG
5  ...
6  ;FOLD USER GLOBALS
7  ;================================
8  ; Userdefined Externals
9  ;================================
10
11 ; Signals for gripper device
12 SIGNAL si_gr_open $in[301]     ;if TRUE, gripper is opened, if FALSE,
13                                  gripper is neither open nor closed
14 SIGNAL si_gr_close $in[302]    ;if TRUE, gripper is closed, if FALSE,
15                                  gripper is neither open nor closed
16 SIGNAL so_gr_enable $out[301]  ;in macro is still TRUE
17 SIGNAL so_gr_open $out[302]    ;if TRUE, it is requirement for open,
18                                  if FALSE, it is requirement for closing
19
20 ; Signals for runnig programs via PLC
21 SIGNAL so_konec_prace $out[14] ;robot requires confirm end of the
22   program
23 SIGNAL si_konec_prace $in[14]  ;PLC confirm end of the program
24
25 ;================================
26 ; Userdefined Variables
27 ;================================
28
29 ; Global variables for Kinect programs which communicates via
30   Ethernet KRL
31 FRAME kinectFrame={X 1000.0,Y -340.0,Z 940.0,A 90.0,B 0.0,C 90.0}
32 FRAME poleKinectFrame[1000] ;pole framu pro offline verzi aplikace
33 FRAME tempFrame ;pomocny frame pro prijem souradnic v offline verzi
34 INT pocetBodu ;pocet nactenych bodu pro offline verzi
35 INT kinectTemp=62
36 BOOL otevritKleste=TRUE ;od ext. systemu zda maji byt kleste otevreny
```

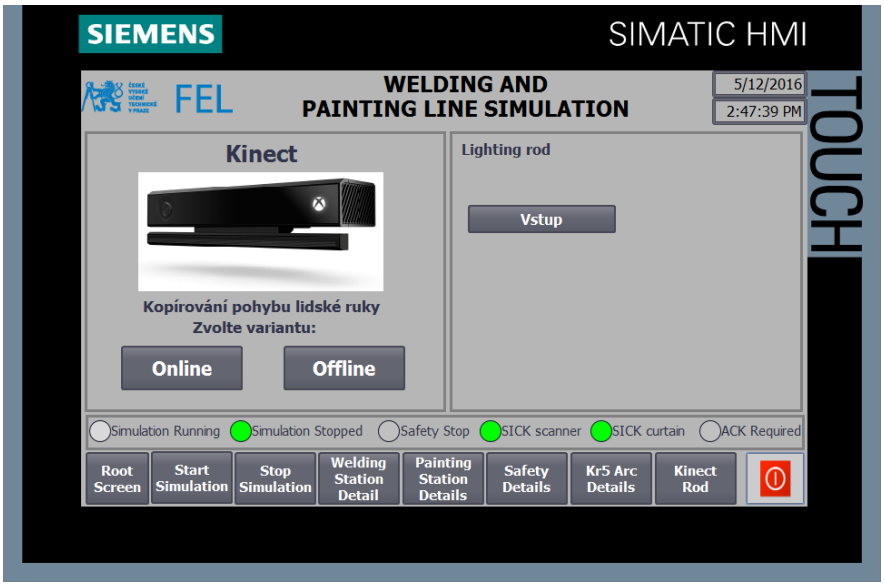
```
37    nebo zavreny (TRUE=otevreny, FALSE=zavreny)
38  SIGNAL programKinectRun $IN[249] ;od PLC (vizualizace) - spusteni
39    provadeni pohybu
40  SIGNAL programKinectStop $IN[250] ;od PLC (vizualizace) - zastaveni
41    provadeni pohybu
42  SIGNAL opakovatPohybANO $IN[251] ;od PLC - pohyb z nactenych bodu
43    opakovat
44  SIGNAL opakovatPohybNE $IN[252] ;od PLC - pohyb z nactenych bodu
45    neopakovat
46  SIGNAL spojeniNavazano $OUT[249] ;informace pro PLC, zda bylo spojeni
47    navazano nebo ne
48  SIGNAL kinectPripojen $OUT[250] ;informace o funkcnim pripojenem senzoru
49    Kinect k externimu PC (informace prijde po Ethernetu)
50  SIGNAL provadeniPohybu $OUT[251] ;signalizuje beh hlavni smycky
51  SIGNAL klesteOtevreny $OUT[252] ;info od klesti pro PLC a rizeni behu
52    programu
53  SIGNAL KlesteZavreny $OUT[253] ;info od klesti pro PLC a rizeni behu
54    programu
55  SIGNAL dataNactena $OUT[254] ;info pro vizualizaci, ze jsou vsechna data
56    nactena
57  SIGNAL opakovatPohyb $OUT[255] ;dotaz k PLC zda se ma pohyb z nactenych
58    bodu opakovat
59  ;$FLAG[1]  ;true=data z Eth. spojeni jsou nactena
60
61  ;ENDFOLD (USER GLOBALS)
62  ENDDAT
```

Příloha G
XML šablona pro odesílání dat serverovým programem

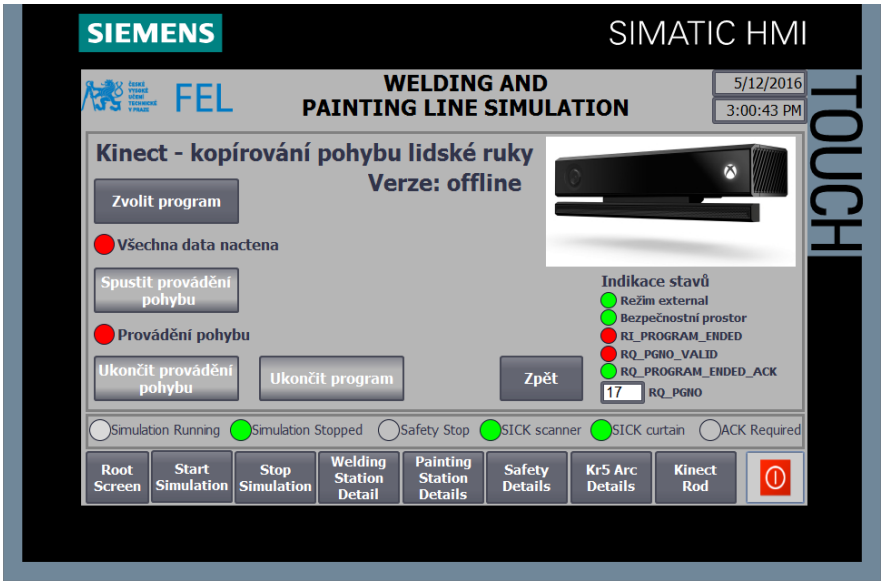
Náplní této přílohy je XML šablona pro odesílaná data serverovým programem řídicímu systému robota. Tento kód se nachází v souboru `ExternalDataXML.XML`, který je umístěn ve složce `XML`. Tato složka musí být umístěna ve stejné složce jako soubor spouštějící serverovou aplikaci (`Kinect_KUKA_server.EXE`).

```
1  ⟨Sensor⟩
2    ⟨Pozice⟩
3      ⟨NoveSouradnice X="" Y="" Z="" A="" B="" C="" /⟩
4    ⟨/Pozice⟩
5    ⟨Kleste⟩
6      ⟨Otevrit⟩⟨/Otevrit⟩
7    ⟨/Kleste⟩
8    ⟨Status⟩
9      ⟨KinectPripojen⟩⟨/KinectPripojen⟩
10   ⟨/Status⟩
11 ⟨/Sensor⟩
```

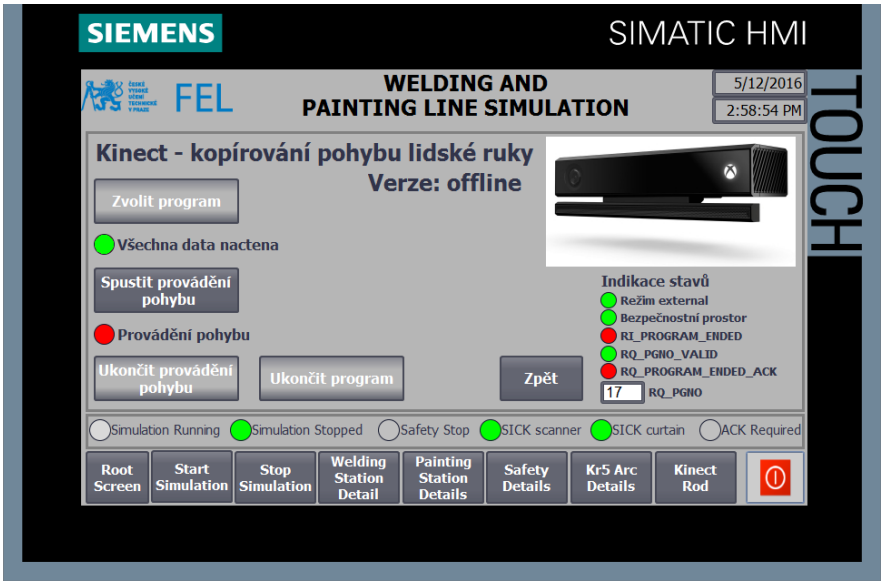
Příloha H
Snímky vizualizace na operátorském panelu – varianta offline demonstrační aplikace



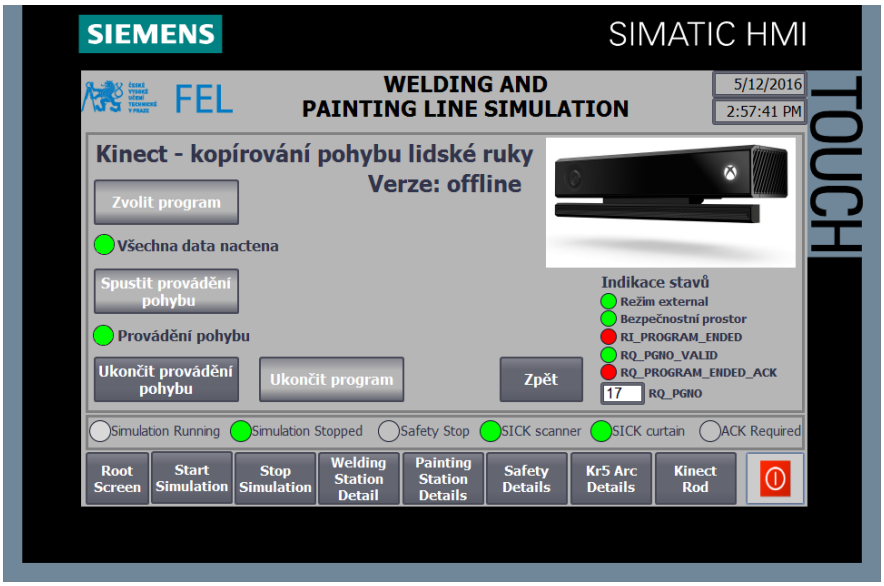
Obrázek H.1. Snímek 1 vizualizačního panelu demonstrační aplikace – výběr varianty



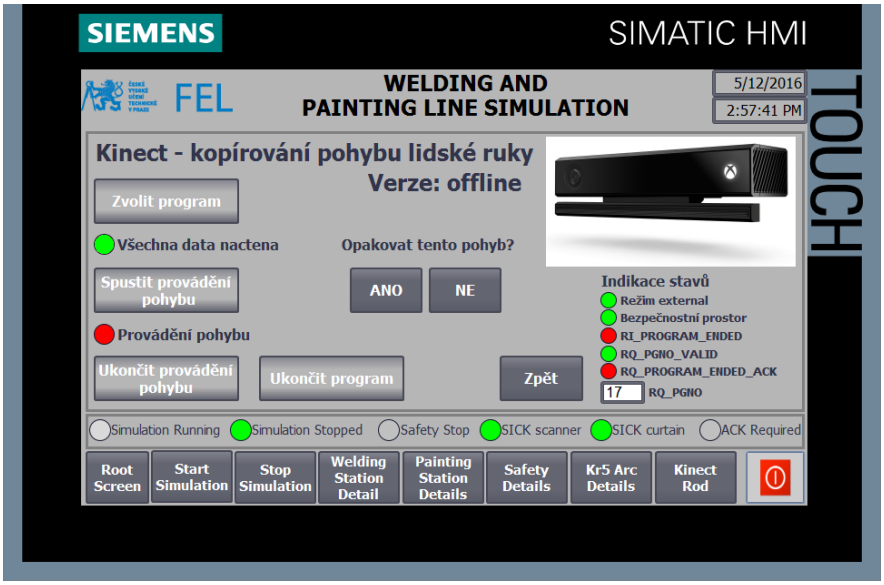
Obrázek H.2. Snímek 2 vizualizačního panelu demonstrační aplikace varianty offline – navolení programu



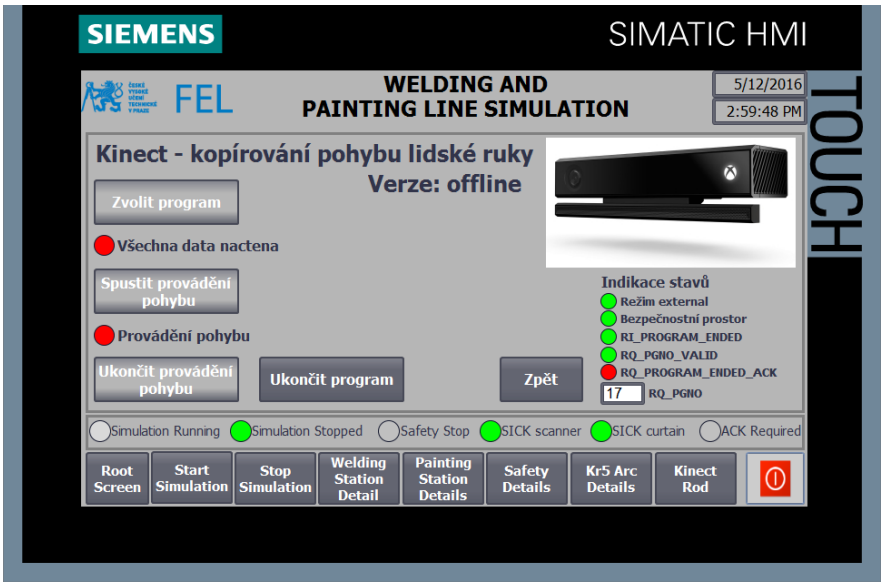
Obrázek H.3. Snímek 3 vizualizačního panelu demonstrační aplikace varianty offline – spuštění provádění pohybu



Obrázek H.4. Snímek 4 vizualizačního panelu demonstrační aplikace varianty offline – možnost předčasně ukončit provádění pohybu

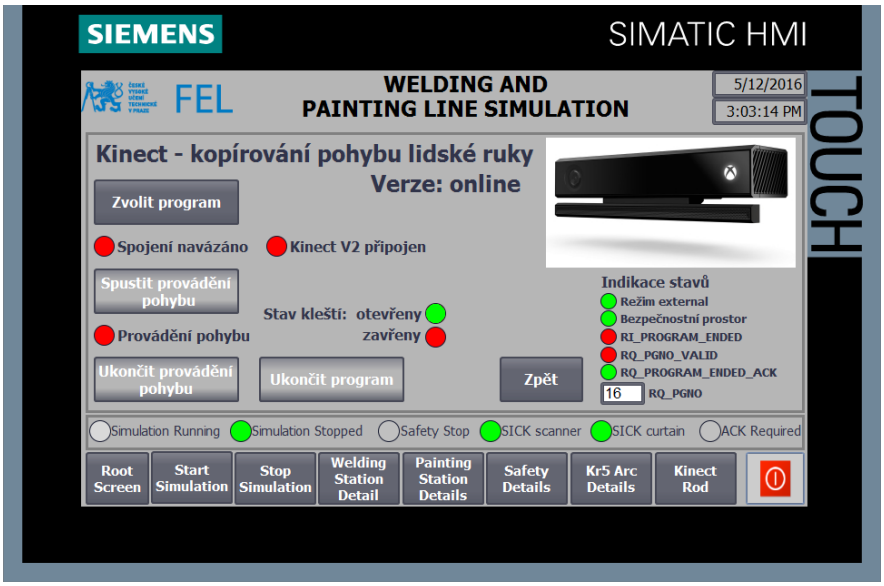


Obrázek H.5. Snímek 5 vizualizačního panelu demonstrační aplikace varianty offline – volba opakování provedeného pohybu

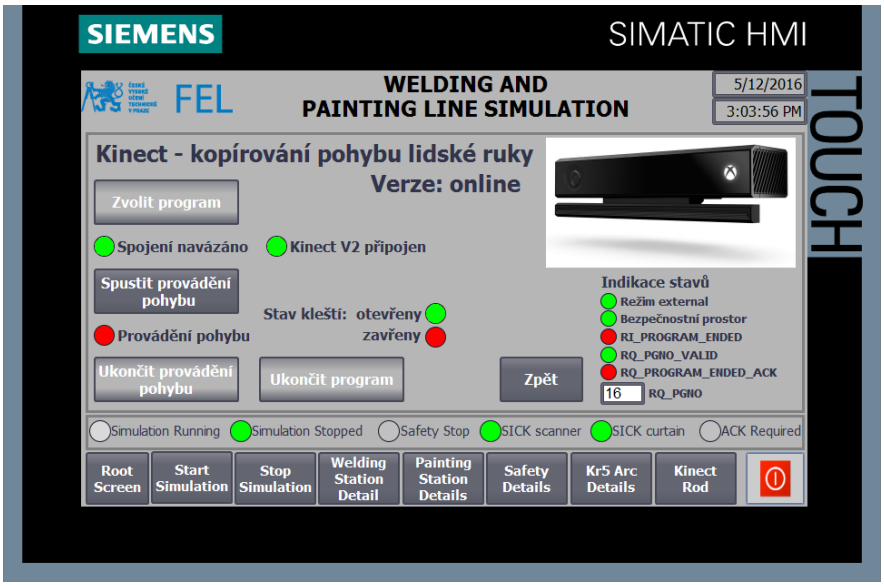


Obrázek H.6. Snímek 6 vizualizačního panelu demonstrační aplikace varianty offline – ukončení zvoleného programu

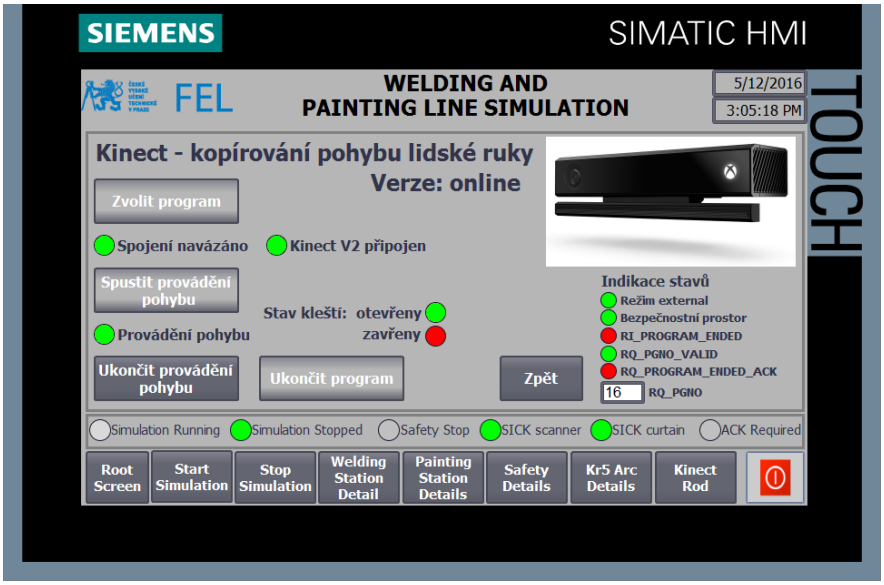
Příloha I
Snímky vizualizace na operátorském panelu – varianta online demonstrační aplikace



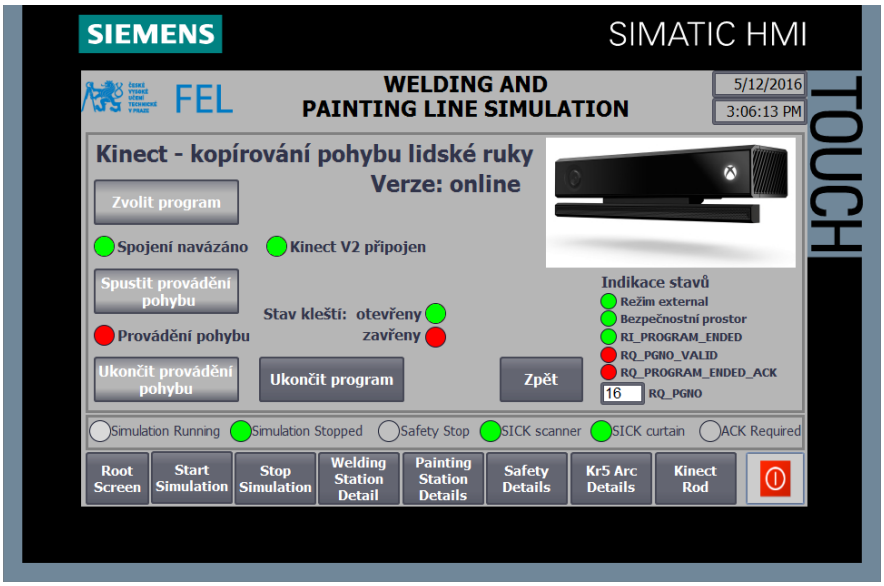
Obrázek I.7. Snímek 7 vizualizačního panelu demonstrační aplikace varianty offline – navolení programu



Obrázek I.8. Snímek 8 vizualizačního panelu demonstrační aplikace varianty offline – spuštění provádění pohybu



Obrázek I.9. Snímek 9 vizualizačního panelu demonstrační aplikace varianty offline – ukončení provádění pohybu



Obrázek I.10. Snímek 10 vizualizačního panelu demonstrační aplikace varianty offline – ukončení zvoleného programu

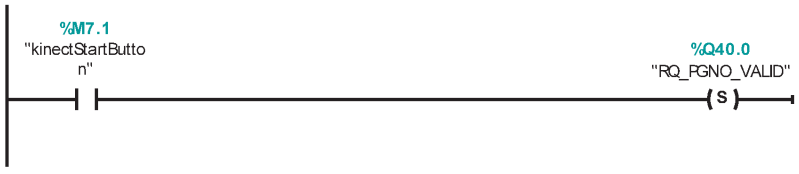
Příloha J
Tagy dodefinované v PLC a LAD diagramy funkčního bloku Kinect [FC5]

Náplní této přílohy je seznam „Tagů“ dodefinovaných v PLC pro tuto demonstrační aplikaci (ostatní používané „Tagy“ již byly součástí převzatého projektu). Dále jsou uvedeny LAD diagramy funkčního bloku `Kinect [FC5]`.

PLC tags

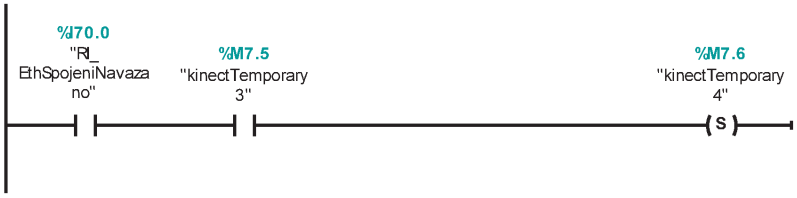
	Name	Data type	Address	Visible in HMI	Accessible from HMI
	kinectStartButton	Bool	%M7.1	True	True
	kinectTemporary1	Bool	%M7.0	True	True
	kinectStartButton2	Bool	%M7.2	True	True
	kinectStopButton	Bool	%M7.3	True	True
	RI_EthSpojeniNavazano	Bool	%I70.0	True	True
	RI_KinectPripojen	Bool	%I70.1	True	True
	RQ_ProgramKinectRun	Bool	%Q70.0	True	True
	RI_provadeniPohybu	Bool	%I70.2	True	True
	kinectTemporary2	Bool	%M7.4	True	True
	kinectTemporary3	Bool	%M7.5	True	True
	kinectTemporary4	Bool	%M7.6	True	True
	kinectTemporary5	Bool	%M7.7	True	True
	RI_klesteOtevreny	Bool	%I70.3	True	True
	RI_klesteZavreny	Bool	%I70.4	True	True
	RI_dataNactena	Bool	%I70.5	True	True
	RI_opakovatPohyb	Bool	%I70.6	True	True
	RQ_programKinectStop	Bool	%Q70.1	True	True
	RQ_opakovatPohybANO	Bool	%Q70.2	True	True
	RQ_opakovatPohybNE	Bool	%Q70.3	True	True
	kinectTemporary6	Bool	%M8.0	True	True
	kinectTemporary7	Bool	%M8.1	True	True
	kinectTemporary8	Bool	%M8.2	True	True

Obrázek J.11. Tagy dodefinované v PLC pro tuto demonstrační aplikaci

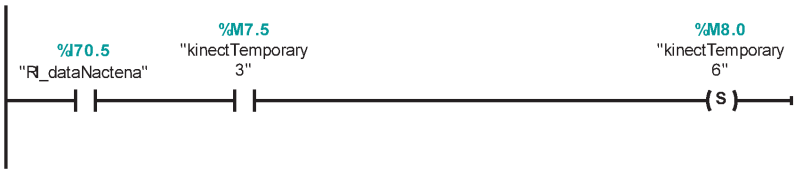


Obrázek J.12. 1. LAD diagram funkčního bloku Kinect [FC5]

Obrázek J.13. 2. LAD diagram funkčního bloku Kinect [FC5]



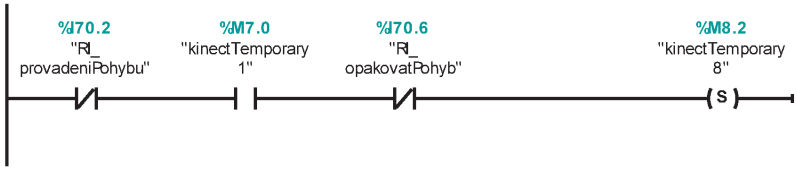
Obrázek J.14. 3. LAD diagram funkčního bloku Kinect [FC5]



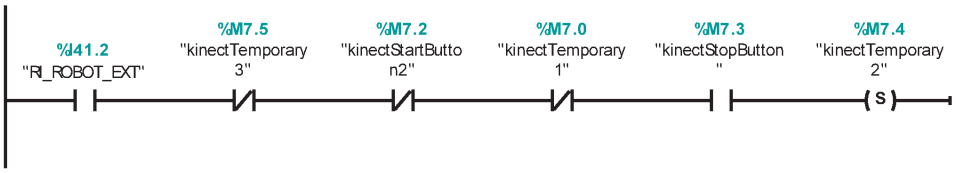
Obrázek J.15. 4. LAD diagram funkčního bloku Kinect [FC5]

Obrázek J.16. 5. LAD diagram funkčního bloku Kinect [FC5]

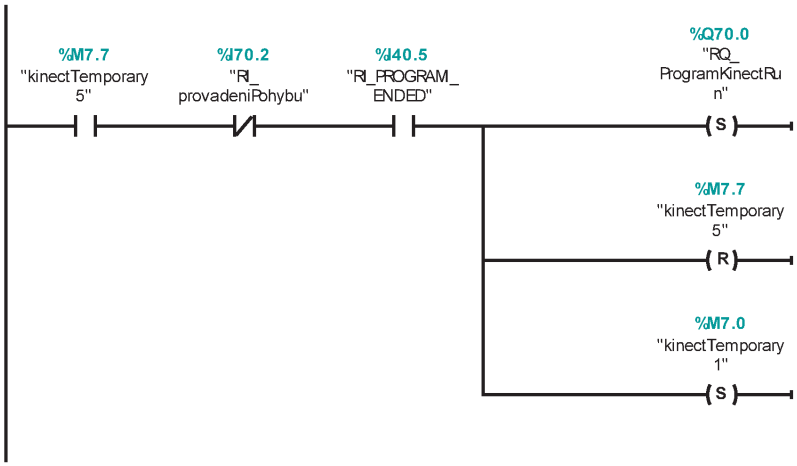
Obrázek J.17. 6. LAD diagram funkčního bloku Kinect [FC5]



Obrázek J.18. 7. LAD diagram funkčního bloku Kinect [FC5]



Obrázek J.19. 8. LAD diagram funkčního bloku Kinect [FC5]



Obrázek J.20. 9. LAD diagram funkčního bloku Kinect [FC5]

Příloha K
Návod k obsluze demonstrační aplikace – varianta online

1. Zapněte externí PC a řídicí systém robota. Připojte senzor *Kinect V2* pomocí adaptéru do portu USB 3.0 externího PC a napájecí zdroj do el. sítě.

2. Na ovládacím panelu řídicího systému robota nalezněte soubor `CELL.SRC` v adresářové struktuře `KRC:\R1`. Klikněte na tento soubor a zvolte ho pomocí tlačítka `Select` v dolní liště. Přepněte klíčový přepínač do vodorovné polohy, klikněte na `AUT` a přepněte klíčový přepínač zpět do svislé polohy. Stiskněte a držte tlačítko `Start` (se zelenou šipkou doprava, nebo zelené tlačítko zespoda ovládacího panelu) na ovládacím panelu, než robot najede do výchozí pozice (na ovládacím panelu se zobrazí `Programmed path reached (BCO)`). Znovu stiskněte tlačítko `Start`, než se zobrazí chybová hláška `Incorrect operating mode`. Přepněte klíčový přepínač do vodorovné polohy, klikněte na `EXT` a přepněte klíčový přepínač zpět do svislé polohy.

3. Na externím PC spusťte serverovou aplikaci. Tato aplikace musí být povolena v bráně Windows Firewall. Pokud tomu tak není, ukončete serverovou aplikaci a povolte ji (viz kap. 3.2).

4. Ve spuštěné serverové aplikaci stiskněte číslo síťové karty (zobrazené IP adresy), ke které je připojen řídicí systém robota (na tomto pracovišti v době odevzdání práce – číslo `3`). Dále stiskněte `1` pro zvolení *online* varianty. Serverová aplikace je v tuto chvíli připravena k provozu.

5. Přejděte k operátorskému panelu a v dolní liště vizualizačního panelu klikněte na záložku `Kinect, Rod`. Na obrazovce, která se otevře, klikněte na `Online`.

6. Na obrazovce, která se otevře, klikněte na `Zvolit program`. Jakmile se rozsvítí zelená kontrolka `Spojení navázáno` a `Kinect V2 připojen` klikněte na `Spustit provádění pohybu`. Rozsvítí se zelená kontrolka `Provádění pohybu`.

7. Přejděte k senzoru *Kinect V2* a postavte se před něj ve vzdálenosti cca 1,5 m. Můžete začít předvádět pohyb pravou rukou a ovládat kleště polohou levé ruky

(levá ruka nad úrovní ramen – kleště otevřeny, levá ruka pod úrovní ramen – kleště zavřeny). Tento předváděný pohyb bude ihned vykonáván robotem.

8. Pro ukončení předvádění pohybu klikněte na operátorském panelu na `Ukončit provádění pohybu` a poté na `Ukončit program`.

9. Pokud chcete znovu začít převádění pohybu opakujte kroky od bodu 6.

10. Pokud chcete demonstrační aplikaci ve variantě *online* ukončit, přejděte k externímu PC a stiskněte klávesu `'q'` pro ukončení serverové aplikace. Celý systém je připraven k jakékoli další práci.

Příloha L
Návod k obsluze demonstrační aplikace – varianta offline s předváděním pohybu

1. Zapněte externí PC a řídicí systém robota. Připojte senzor *Kinect V2* pomocí adaptéru do portu USB 3.0 externího PC a napájecí zdroj do el. sítě.

2. Na ovládacím panelu řídicího systému robota nalezněte soubor `CELL.SRC` v adresářové struktuře `KRC:\R1`. Klikněte na tento soubor a zvolte ho pomocí tlačítka `Select` v dolní liště. Přepněte klíčový přepínač do vodorovné polohy, klikněte na `AUT` a přepněte klíčový přepínač zpět do svislé polohy. Stiskněte a držte tlačítko `Start` (se zelenou šipkou doprava, nebo zelené tlačítko zespoda ovládacího panelu) na ovládacím panelu, než robot najede do výchozí pozice (na ovládacím panelu se zobrazí `Programmed path reached (BCO)`). Znovu stiskněte tlačítko `Start`, než se zobrazí chybová hláška `Incorrect operating mode`. Přepněte klíčový přepínač do vodorovné polohy, klikněte na `EXT` a přepněte klíčový přepínač zpět do svislé polohy.

3. Na externím PC spusťte serverovou aplikaci. Tato aplikace musí být povolena v bráně Windows Firewall. Pokud tomu tak není, ukončete serverovou aplikaci a povolte ji (viz kap. 3.2).

4. Ve spuštěné serverové aplikaci stiskněte číslo síťové karty (zobrazené IP adresy), ke které je připojen řídicí systém robota (na tomto pracovišti v době odevzdání práce – číslo `3`). Dále stiskněte `2` pro zvolení *offline* varianty a `1` pro zvolení předvádění nového pohybu před senzorem *Kinect V2*.

5. Postavte se před senzor *Kinect V2* a předveďte pohyb, který chcete, aby robot zopakoval. Zaznamenávání pohybu započnete zvednutím levé ruky nad úroveň ramen. Během zaznamenávání pohybu mějte levou ruku stále zdviženou. Pro ukončení zaznamenávání pohybu dejte levou ruku pod úroveň ramen. Serverová aplikace zobrazí počet načtených bodů a čeká na spojení od řídicího systému.

6. Přejděte k operátorskému panelu a v dolní liště vizualizačního panelu klikněte na záložku `Kinect, Rod`. Na obrazovce, která se otevře, klikněte na `Offline`.

7. Na obrazovce, která se otevře, klikněte na `Zvolit program`. Jakmile se rozsvítí zelená kontrolka `Všechna data načtena` klikněte na `Spustit provádění pohybu`. Rozsvítí se zelená kontrolka `Provádění pohybu` a robot začne provádět předvedený pohyb.

8. Toto provádění pohybu můžete předčasně ukončit kliknutím na `Ukončit provádění pohybu`.

9. Po dokončení (nebo předčasném ukončení) prováděného pohybu se na operátorském panelu zobrazí dotaz na opakování předvedeného pohybu. Pokud chcete pohyb opakovat klikněte na `ANO`, pohyb se začne opakovat (přesuňte se k bodu 8). Pokud nechcete pohyb opakovat klikněte na `NE`.

10. Následně demonstrační aplikaci ukončete kliknutím na `Ukončit program`. Přejděte k externímu PC a stiskněte klávesu `‘q‘` pro ukončení serverové aplikace. Celý systém je připraven k jakékoli další práci.

Příloha M
Návod k obsluze demonstrační aplikace – varianta offline s načtením dříve předvedeného pohybu ze souboru

1. Zapněte externí PC a řídicí systém robota.

2. Na ovládacím panelu řídicího systému robota nalezněte soubor `CELL.SRC` v adresářové struktuře `KRC:\R1`. Klikněte na tento soubor a zvolte ho pomocí tlačítka `Select` v dolní liště. Přepněte klíčový přepínač do vodorovné polohy, klikněte na `AUT` a přepněte klíčový přepínač zpět do svislé polohy. Stiskněte a držte tlačítko `Start` (se zelenou šipkou doprava, nebo zelené tlačítko zespoda ovládacího panelu) na ovládacím panelu, než robot najede do výchozí pozice (na ovládacím panelu se zobrazí `Programmed path reached (BCO)`). Znovu stiskněte tlačítko `Start`, než se zobrazí chybová hláška `Incorrect operating mode`. Přepněte klíčový přepínač do vodorovné polohy, klikněte na `EXT` a přepněte klíčový přepínač zpět do svislé polohy.

3. Na externím PC spusťte serverovou aplikaci. Tato aplikace musí být povolena v bráně Windows Firewall. Pokud tomu tak není, ukončete serverovou aplikaci a povolte ji (viz kap. 3.2).

4. Ve spuštěné serverové aplikaci stiskněte číslo síťové karty (zobrazené IP adresy), ke které je připojen řídicí systém robota (na tomto pracovišti v době odevzdání práce – číslo `3`). Dále stiskněte `2` pro zvolení *offline* varianty a `2` pro načtení uloženého dříve předvedeného pohybu.

5. Z dostupných souborů vyberte soubor, který chcete načíst (zadejte číslo souboru a stiskněte Enter). Serverová aplikace zobrazí počet načtených bodů ze zadaného souboru a čeká na spojení od řídicího systému.

6. Přejděte k operátorskému panelu a v dolní liště vizualizačního panelu klikněte na záložku `Kinect, Rod`. Na obrazovce, která se otevře, klikněte na `Offline`.

7. Na obrazovce, která se otevře, klikněte na `Zvolit program`. Jakmile se rozsvítí zelená kontrolka `Všechna data načtena` klikněte na `Spustit provádění pohybu`. Rozsvítí se zelená kontrolka `Provádění pohybu` a robot začne provádět předvedený pohyb.

8. Toto provádění pohybu můžete předčasně ukončit kliknutím na `Ukončit provádění pohybu`.

9. Po dokončení (nebo předčasném ukončení) prováděného pohybu se na operátorském panelu zobrazí dotaz na opakování předvedeného pohybu. Pokud chcete pohyb opakovat klikněte na `ANO`, pohyb se začne opakovat (přesuňte se k bodu 8). Pokud nechcete pohyb opakovat klikněte na `NE`.

10. Následně demonstrační aplikaci ukončete kliknutím na `Ukončit program`. Přejděte k externímu PC a stiskněte klávesu `‘q‘` pro ukončení serverové aplikace. Celý systém je připraven k jakékoli další práci.

Příloha N
Soupis některých chyb a jejich řešení

N.1 Serverová aplikace, externí PC, Kinect V2

- *Senzor Kinect V2 nebyl nalezen!* – Připojte senzor Kinect V2 pomocí adaptéru k portu USB 3.0 počítače a napájecí zdroj do el. sítě.

- *Komunikace se senzorem Kinect V2 nebyla správně navázána!* – Odpojte a znovu připojte senzor Kinect V2 pomocí adaptéru k portu USB 3.0 počítače a restartujte serverovou aplikaci.

- *Byl dosazen maximální počet bodů předváděného pohybu, záznam pohybu byl ukončen.* – Maximální počet bodů zaznamenávaného pohybu je 1000, což odpovídá přibližně 5-ti minutám a 33 sekundám záznamu předváděného pohybu. Pokud by tento čas pro předváděný pohyb nestačil, je nutné pohyb rozdělit na 2 a více jednotlivých pohybů, které nejsou delší než přibližný uvedený čas záznamu.

N.2 Řídicí systém robota

- *Během provádění pohybu se robot zastaví a na ovládacím panelu se zobrazí hláška* `"Workspace error."`. – Předváděný (resp. předvedený) pohyb není možné provést. Pohyb zasahuje mimo pracovní prostor robota. Restartujte program `CELL.SRC` v řídicím systému robota a znovu ho spusťte podle bodu 2 návodu k obsluze viz příloha K. Stiskněte potvrzovací (bílé) tlačítko na operátorském panelu. Klikněte na tlačítko `Zpět`, poté klikněte na Vámi požadovanou variantu a dále znovu spusťte program postupem uvedeným v daném návodu.

- Na ovládacím panelu se zobrazí hláška `"External EMERGENCY STOP."`. – Došlo k narušení bezpečnostního prostoru robota, nebo je stále narušen. Pokud svítí červený majáček, uvolněte bezpečnostní prostor. Pokud svítí oranžový a zároveň zelený majáček, stiskněte potvrzovací (bílé) tlačítko na operátorském panelu. Restartujte program `CELL.SRC` v řídicím systému robota a znovu ho spusťte podle bodu 2 návodu k obsluze viz příloha K.

N.3 Externí spouštění programů, vizualizace

- *Po stisknutí tlačítka* `Zvolit program` *nedojde k navázání spojení nebo přenesení dat (kontrolka je stále červená).* – Problém 1: Nedošlo ke správnému spuštění programu. Pokuste se restartovat program `CELL.SRC` v řídicím systému robota a znovu ho spustit podle bodu 2 návodu k obsluze viz příloha K. Stiskněte potvrzovací (bílé) tlačítko na operátorském panelu. Klikněte na tlačítko `Zpět`, poté klikněte na Vámi požadovanou variantu a dále znovu spusťte program postupem uvedeným v daném návodu.
  – Problém 2: Nebylo navázáno spojení řídicího systému robota s externím systémem. Na externím PC ověřte pomocí příkazu `ping` dostupnost řídicího systému v síti. Pokud je řidicí systém robota dostupný, pravděpodobně nemáte správně povolenou serverovou aplikaci v bráně Windows Firewall, povolte ji podle návodu viz kap. 3.2.

- *Po zvolení varianty demonstrační aplikace, není možné zvolit program pomocí tlačítka* `Zvolit program` *(je neaktivní).* – Pravděpodobně není řídicí systém robota v režimu `EXT AUT`. Postupujte podle bodu 2 návodu k obsluze viz příloha K.

Příloha O
Obsah elektronické přílohy (CD)

```
Bakalarska_prace-Dastych_Lukas
|---Demonstracni_aplikace_Kinect_KUKA
|   |---Konfiguracni_soubor_doplnku_KUKA_Ethernet_KRL_v_KRC
|   |---Konfiguracni_soubor_v_KRC
|   |---Konfiguracni_XML_soubor_ethernetoveho_spojeni
|   |---Makra_pro_ovladani_klesti_v_KRC
|   |---Program_pro_volbu_externe_spoustenych_programu_v_KRC
|   |---Projekt_v_TIA Portal_V13
|   |   |---Kuka_kinect
|   |---Roboticke_programy_v_KRC
|   |---Serverova_aplikace
|   |   |---generovane_soubory
|   |   |---Projekt_ve_Visual_Studiu_2015
|   |   |   |---Kinect_KUKA_server
|   |   |---XML
|   |---Soubor_pro_nastaveni_laseroveho_scaneru_SICK_S3000
|---Foto_a_video
|---Literatura
|---Propojení_senzoru_Kinect_na_průmyslového_robota.pdf
```