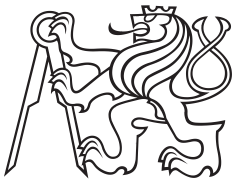
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Řízení trajektorie pohybu průmyslového robota

Pavel Král

Obor: Systémy a řízení
Studijní program: Kybernetika a robotika
Květen 2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Pavel Král**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Řízení trajektorie pohybu průmyslového robota**

Pokyny pro vypracování:

1. Seznamte se s principy programování průmyslového robota Kuka.
2. Navrhněte demonstrační aplikaci pohybu robota, při které bude trajektorie vypočítána na základě zadaných parametrů jako například přenos vybraného předmětu z místa na místo. Při počítání parametrů trajektorie zohledněte zadaná omezení na vstupu, která umožní nepřekročit zvolenou hodnotu zrychlení apod.
3. Vytvořte program v Matlabu pro plánování trajektorie robota.
4. Navrhněte způsob přenosu parametrů trajektorie do řídicího systému robota jako robotického programu. Implementujte řízení robota na připojeném PLC.
5. Vytvořte uživatelské rozhraní, které umožní systém ovládat a používat jako demonstrační model.

Seznam odborné literatury:

1. Kuka industrial robots programmming 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

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Pavlu Burgetovi, Ph.D. za jeho rady a čas, který mi věnoval při řešení bakalářské práce.

Prohlášení

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 .....

Podpis .....

Abstrakt

Tato práce se zaměřuje na řízení trajektorie robota KUKA KR5 arc pro světelnou tyč Visual Poi V3. Cílem práce je navrhnout způsob přichycení světelné tyče ke koncovému bodu robota a vypočítat výslednou trajektorii koncového bodu robota, kterou je poté možno přenést ve formě programu pro robota do jeho řídicího systému. Robot pak po zadané trajektorii vykoná pohyb. Přenos je možné ovládat z vizualizačního panelu.

Klíčová slova: KUKA, průmyslový robot, robot, trajektorie, Visual Poi V3

Abstract

This thesis aims to control of trajectory of robot KUKA KR5 for Visual Poi V3. The goal of the thesis is designing method to connect Visual Poi to end point of the robot and calculate trajectory of the robot end point, which can be transmitted to robot controller as robot program. Then robot end point follows trajectory. Transmission can be controlled through user interface.

Keywords: KUKA, industrial robot, robot, trajectory, Visual Poi V3

Title translation: Trajectory control of an industrial robot

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Cílem této práce je navržení demonstrační aplikace robota, kdy je k průmyslovému robotovi KR5 Arc značky KUKA přichycena světelná tyč Visual Poi V3, která je při pohybu, nejčastěji rotačnímu, schopna zobrazení obrazu pomocí dvou řad po 80ti LED diodách, které jsou umístěny na světelné tyči. Robot tedy na základě vypočítané trajektorie vykoná pohyb takový, aby světelná tyč dosáhla rychlosti, při které je schopna zobrazit obraz. Podrobnější popis světelné tyče je v kapitole 3

V prvé řadě je tedy potřeba navrhnout způsob přichycení světelné tyče k robotovi a poté vypočítat takovou trajektorii, aby světelná tyč byla schopná bezproblémového zobrazení obrazu. Pak navrhnout způsob, jak přenést parametry vypočtené trajektorie do řídicího systému robota, ke kterému je robot připojen, aby poté robot mohl pohyb po vypočítané trajektorii vykonat. Celý systém bude řízen pomocí PLC, které bude možné ovládat pomocí vizualizačního panelu, ke kterému je PLC připojeno.

1.1 Seznámení s pracovištěm

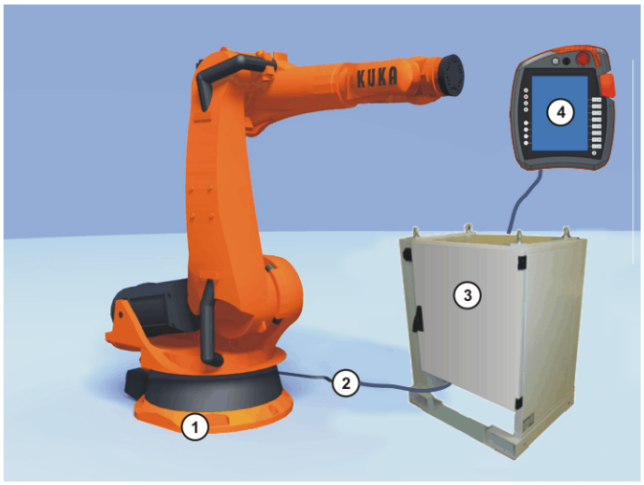
Celé pracoviště se skládá ze čtyř částí. Zobrazené jsou na obrázku níže.

Jednotlivé komponenty jsou pojmenovány v tabulce.

Číslo	Komponenta
1	Robot
2	Propojovací kabely
3	Řídicí systém
4	SmartPAD

Tabulka 1.1: Názvy komponent na obrázku 1.1

Robotem (1) je robot KR5 Arc, který má celkem šest os (6 rotačních kloubů). Ke koncovému bodu robota jsou připevněny kleště.



Obrázek 1.1: Propojení robota s řídicím systémem, převzaté z [2]

Řídicím systémem (3) robota je řídicí systém KR C4.

SmartPAD (4) je malé zařízení s dotykovou obrazovkou. K tomuto zařízení je možné připojit externí myš nebo klávesnici, kterými je možné ho pohodlně ovládat. K SmartPADu je možné připojit i flash disk souborového systému FAT. Podrobnější popis zařízení bude v další kapitole.

Součástí pracoviště je také skříň, ve které je umístěno PLC a vizualizační panel. Celé pracoviště je chráněno bezpečnostními prvky, kterými jsou bezpečnostní světelné závěsy SICK C4000 Standard a bezpečnostní laserový skener SICK S3000, který snímá, zda není narušeno pracoviště a jeho okolí.

Kapitola 2
Programování robota

V této kapitole jsem převážně vycházel z manuálu o programování robota [3].

Programovacím jazykem robota je KUKA Robot Language, zkráceně KRL. KRL je podobný běžným programovacím jazykům, jedná se o imperativní neobjektový programovací jazyk.

Programovat robota je možné ve dvou režimech. V uživatelském režimu a expertním režimu. Expertní režim nabízí více možností, umožňuje využívat řídicí struktury jako větvení, cykly, používat proměnné a další možnosti. V dalších sekcích uvedu programování jak v uživatelském, tak expertním režimu.

2.1 Robot

Robot se skládá ze šesti rotačních kloubů. Umístění a orientace os jsou patrné z obrázku 2.1.

2.2 SmartPAD

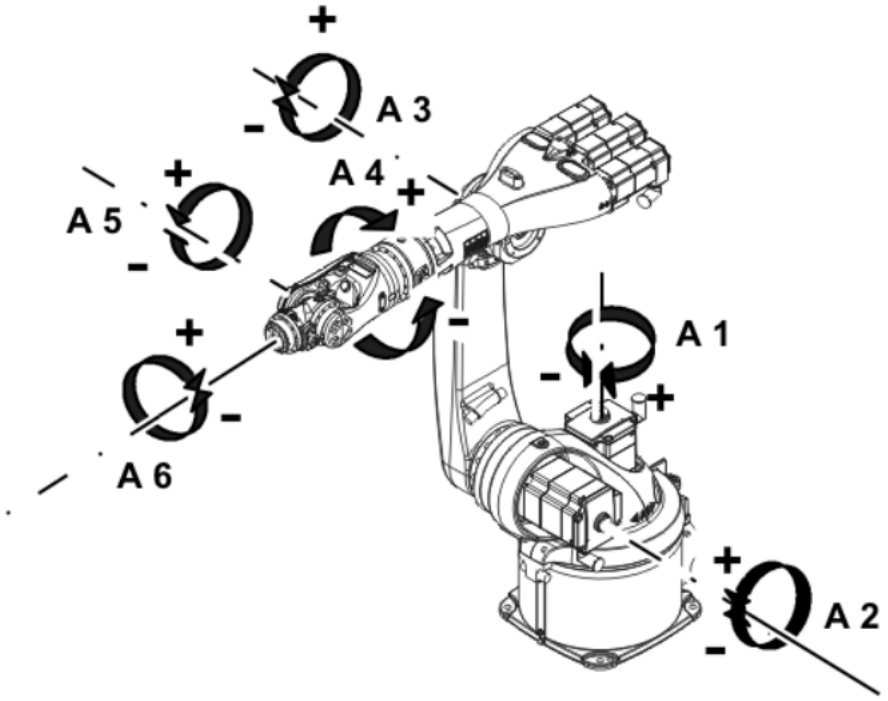
Robot se programuje pomocí SmartPADu, v dokumentaci je možné se v souvislosti se SmartPADem setkat také s pojmem KCP (KUKA Control Panel).

Klíčový spínač (2) nastavuje operační módy uvedeny v tabulce 2.2

Po zapnutí a inicializaci řídicího systému je spuštěn program SmartHMI v operátorském režimu.

2.3 Souřadnicové systémy

Souřadnicové systémy slouží k tomu, aby bylo možné popsat polohu koncového bodu robota. Souřadnicové systémy jsou jak osové, kdy je poloha bodu určena úhly natočení os robota, tak kartézskými souřadnicemi, kde

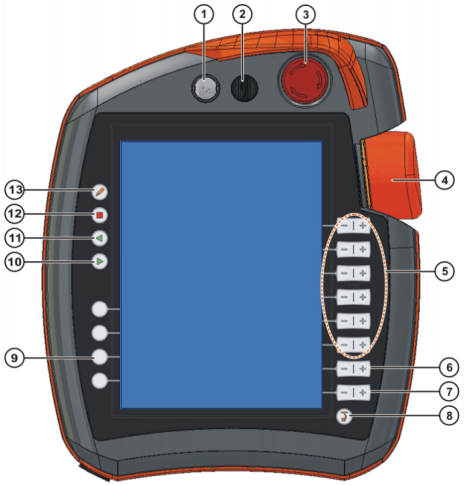


Obrázek 2.1: Osy robota[3]

Prvek	Popis prvku
1	Odpojení SmartPADu
2	Klíčový spínač pro změnu módu
3	Nouzové STOP tlačítko, po stisknutí se zablokuje
4	Prostorová myš pro manuální pohyb robota
5	Tlačítka pro manuální pohyb robota
6	Tlačítko pro nastavení rychlosti pohybu při programovém pohybu
7	Tlačítko pro nastavení rychlosti pohybu při manuálním pohybu
8	Tlačítko pro zobrazení hlavního menu v programu SmartHMI
9	Stavové tlačítko
10	Tlačítko pro spuštění programu
11	Tlačítko pro běh programu pozpátku (probíhá krok po kroku)
12	Tlačítko pro zastavení programu
13	Tlačítko pro zobrazení klávesnice

Tabulka 2.1: Popis tlačítek na přední části SmartPADu

řídicí systém přepočítává polohu koncového bodu z kartézských souřadnic do osových souřadnic a naopak. V obou případech je poloha bodu určena šesti hodnotami. V kartézském případě kromě souřadnic X, Y a Z jsou i souřadnice natočení.



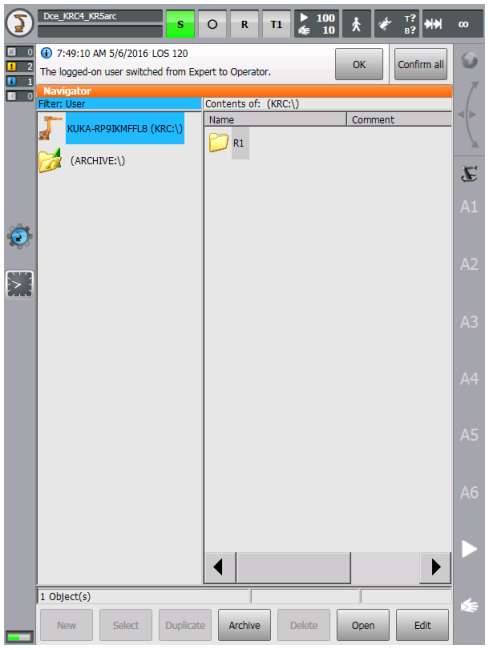
Obrázek 2.2: Přední část SmartPAD[3]

Mód	Popis omezení rychlosti Možnost manuálního módu
T1	Mód určený pro testování, programování a učení robota Maximální rychlost 250 mm/s Manuální ovládání robota je možné
T2	Mód určený pro testování Rychlost není omezená Manuální ovládání robota není možné
AUT	Mód určený pro průmyslové roboty bez ext. říd. systému Rychlost není omezená Manuální ovládání robota není možné
EXT AUT	Mód určený pro průmyslové roboty s ext. říd. systémem Rychlost není omezená Manuální ovládání robota není možné

Tabulka 2.2: Módy ovládání robota

Souřadnice	Popis
A	otočení kolem osy Z
B	otočení kolem osy Y
C	otočení kolem osy X

Tabulka 2.3: Otočení kolem os



Obrázek 2.3: Zobrazení hlavní obrazovky v roli operátor

2.4 Manuální pohyb

Robotem lze pohybovat manuálně. Možnost manuálního pohybu je pouze v módu T1, jak je uvedeno v tabulce 2.2.

Jsou možné dva druhy manuálního pohybu. Prvním druhem pohybu je pohyb kartézský, kde se robot ovládá změnou souřadnic X, Y, Z, A, B, C. Druhým druhem pohybu je pohyb osově specifický, kde je robot ovládán změnou úhlů os A1 až A6.

Robot se ovládá pomocí tlačítek pro manuální pohyb robota a nebo prostorovou myší pro manuální pohyb robota (2.1)

2.5 Druhy pohybů robota

Před programováním je důležité se seznámit s možnostmi pohybu robota. Ty jsou:

- PTP - Point to point
- LINE - lineární pohyb
- CIRC - pohyb po kružnici
- Spline

U pohybu PTP se robot pohybuje po nejrychlejší možné dráze, kterou si robot přepočítá. Robot má rotační klouby, dráha po přímce tedy nemusí být vždy ta nejrychlejší.

U pohybu LINE se robot pohybuje po úsečce. Úsečka je v euklidovském prostoru nejkratší spojnice dvou bodů.

U pohybu CIRC se robot pohybuje po kruhové dráze. Kruhová dráha je určena třemi body. Počátečním bodem, cílovým bodem a pomocným bodem P_{AUX}

2.6 Programování robota v uživatelském režimu

Celý program robota se skládá ze dvou souborů. Prvním souborem je soubor typu .src, kde je uložený zdrojový kód programu. Dalším souborem je datový soubor .dat, který je datovým listem. Může obsahovat např. souřadnice bodů.

V uživatelském režimu může programátor programu měnit, přidávat, mazat pohyby. Přístup má pouze k souboru se zdrojovým kódem (soubor .src). K datovému souboru přístup nemá, ale přidáváním bodů do programu se datový soubor sám generuje. Navigator obrazovka v programu SmartHMI pro programátora v roli Operator je zobrazena na obrázku 2.3

V uživatelském režimu jsou vidět pouze soubory s programem. Jak vypadá program v uživatelském režimu je ukázáno na obrázku 2.6

Ve složce KRC:\R1\Program , která se nachází v řídicím systému robota je možné vytvářet nové programy.

2.6.1 Naprogramování pohybu robota

Nově vytvořený program je zobrazen na obrázku 2.4

```
1  DEF pokus( )
2  INI
3
4  PTP HOME  Vel= 100 % DEFAULT
5
6  PTP HOME  Vel= 100 % DEFAULT
7
8  END
```

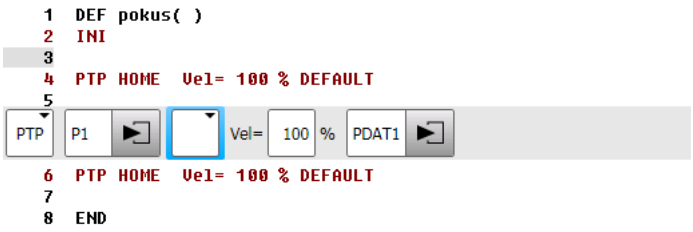
Obrázek 2.4: Nově vytvořený program

Nový pohyb se programuje pomocí Inline formuláře, kde je možnost volby například druhu pohybu, rychlosti, zrychlení. 2.5

Po vytvoření několika bodů může program vypadat podobně jako na obrázku 2.6

2.7 Programování robota v expertním režimu

Základní rozdíl v expertním režimu je ten, že v navigátoru je kromě souboru s programem viditelný i datový soubor. Navíc je možné v expertním režimu rozkliknout bloky programu a zobrazit podrobnější kód. Na obrázku 2.7 je



Obrázek 2.5: Inline formulář

```
1  INI
2
3  PTP HOME  Vel= 100 % DEFAULT
4
5  PTP P1 Vel=100 % PDAT1 Tool[1]:kleste7 Base[0]
6  CIRC P3 P4 Vel=2 m/s CPDAT1 Tool[1]:kleste7 Base[0]
7  PTP P5 Vel=100 % PDAT2 Tool[1]:kleste7 Base[0]
8  CIRC P6 P7 Vel=2 m/s CPDAT2 Tool[1]:kleste7 Base[0]
9  PTP HOME  Vel= 100 % DEFAULT
10
```

Obrázek 2.6: Program v uživatelském režimu

program z obrázku 2.6 zobrazený v expertním režimu. Základním rozdílem je, že v expertním režimu jsou bloky k pohybům viditelné.

```
 1  INI
 2  BASISTECH INI
 3  USER INI
 4
 5  PTP HOME  Vel= 100 % DEFAULT
 6  $BWDSTART = FALSE
 7  PDAT_ACT=PDEFAULT
 8  FDAT_ACT=FHOME
 9  BAS (#PTP_PARAMS,100 )
10  $H_POS=XHOME
11  PTP  XHOME
12
13  PTP P1 Vel=100 % PDAT1 Tool[1]:kleste7 Base[0]
14  $BWDSTART=FALSE
15  PDAT_ACT=PPDAT1
16  FDAT_ACT=FP1
17  BAS(#PTP_PARAMS,100)
18  PTP XP1
19
20  CIRC P3 P4 Vel=2 m/s CPDAT1 Tool[1]:kleste7 Base[0]
21  $BWDSTART=FALSE
22  LDAT_ACT=LCPDAT1
23  FDAT_ACT=FP4
24  BAS(#CP_PARAMS,2)
25  CIRC XP3, XP4
26
27  PTP P5 Vel=100 % PDAT2 Tool[1]:kleste7 Base[0]
28
29  CIRC P6 P7 Vel=2 m/s CPDAT2 Tool[1]:kleste7 Base[0]
30
31  PTP HOME  Vel= 100 % DEFAULT
32  $BWDSTART = FALSE
33  PDAT_ACT=PDEFAULT
34  FDAT_ACT=FHOME
35  BAS (#PTP_PARAMS,100 )
36  $H_POS=XHOME
37  PTP  XHOME
38
```

Obrázek 2.7: Program v expertním režimu

2.7.1 Datové typy

V programu KRL je možné používat proměnné. Proměnná reprezentuje určité místo paměti, kde je uložena její hodnota.

Datový typ určuje, kolik paměti proměnná zabírá a jak má být interpretována její hodnota, která je v paměti uložena v binární podobě. Jazyk KUKA KRL obsahuje 4 základní datové typy:

Datový typ	Popis
Boolean	Jednobitový, TRUE nebo FALSE
Int	Celá čísla
Real	Reálná čísla
Char	Znaky, například 'a', '.'

Tabulka 2.4: Základní datové typy

Dále existují i strukturové datové typy. Struktura je datový typ, který v sobě zahrnuje skupinu proměnných, které spolu souvisí nebo jsou součástí nějakého většího celku.

Struktura	Prvky struktury
POS	X, Y, Z, A, B, C, S, T
E6POS	X, Y, Z, A, B, C, E1, E2, E3, E4, E5, E6
FRAME	X, Y, Z, A, B, C
AXIS	A1, A2, A3, A4, A5, A6
E6AXIS	A1, A2, A3, A4, A5, A6, E1, E2, E3, E4, E5, E6

Tabulka 2.5: Souřadnicové struktury

Struktura FRAME uchovává prostorové souřadnice, v této práci je proto důležitá. Tato struktura se od struktur POS nebo E6POS liší tím, že POS obsahuje prvky S a T, které určují kromě polohy i natočení os robota (ne souřadných os) a E6POS obsahuje prvky externích os. Prvky struktury FRAME jsou vysvětleny v tabulce 2.6.

Prvek	Popis
X	souřadnice x
Y	souřadnice y
Z	souřadnice z
A	úhel natočení na ose z
B	úhel natočení na ose y
C	úhel natočení na ose x

Tabulka 2.6: Prvky struktury FRAME

Dále jazyk KRL umožňuje i vytváření polí. Pole je speciální datový typ, který umožňuje uložit více hodnot stejného typu. Při deklaraci pole se uvádí i jeho velikost – tedy, kolik prvků daného typu chci do pole uložit. Lze tvořit i tzv. pole polí, kdy každý prvek pole obsahuje své pole. Takovým polím se říká dvourozměrné pole.

2.7.2 Řídicí struktury

Základní řídicí struktury jsou větvení a cykly. Větvení probíhá podle určité podmínky. Základními příkazy větvení je IF-THEN, IF-THEN-ELSE, SWITCH-CASE

Nejjednodušším cyklem v jazyce KRL je smyčka LOOP, kterou je možno ukončit příkazem EXIT. KRL zná i další cykly, které jsou časté i v jiných programovacích jazycích, například FOR, WHILE nebo REPEAT-UNTIL.

Dalšími řídicími strukturami jsou příkazy WAIT FOR a WAIT SEC. WAIT FOR pozastaví program do doby, dokud není splněna podmínka. WAIT SEC

pozastaví program na určitou dobu.

2.8 Shrnutí

V této kapitole jsem uvedl základní princip programování, kde programovat lze ve dvou režimech. Uvedl jsem příklad programování v uživatelském režimu a příklad programování v operátorském režimu. Dále jsem představil základní datové typy a datový typ struktury, který obsahuje skupinu proměnných. Na závěr jsem uvedl základní řídicí struktury, které využiji v dalších kapitolách.

Kapitola 3

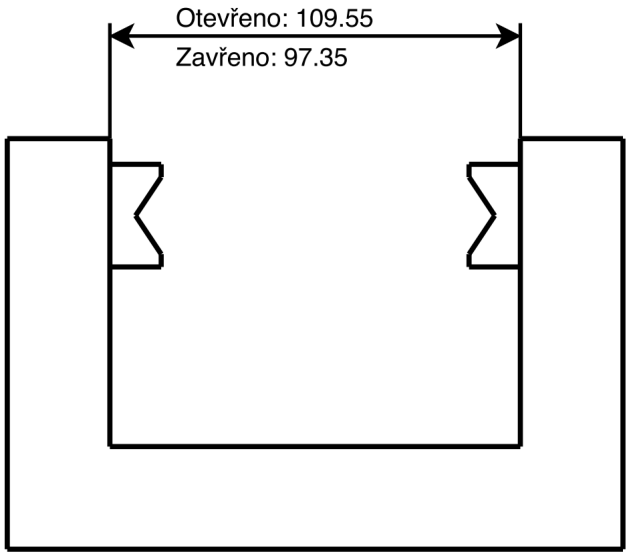
Pohyb světelnou tyčí robotem

Cílem bylo navrhnout způsob uchycení světelné tyče robotem. Světelná tyč Visual Poi V3 (obrázek) je 49 centimetrů[1] dlouhá s průměrem 2.49 centimetrů. Hmotnost tyče je 170 gramů[1]. Tyč obsahuje jedno tlačítko, kterým se tyč zapíná a nastavuje se program.

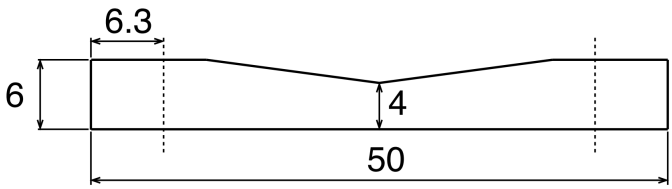
Obrázek 3.1: Světelné tyče Visual Poi V3, obrázek převzatý z [6]

Poté jsem navrhl dvě možné varianty přichycení robota. První variantou bylo přichycení světelné tyče přímo ke kleštím. Tuto variantu jsem zvolil, protože realizace tohoto řešení je jednoduchá, stačí pouze vyměnit vložky uvnitř kleští. Problémem může ale být možná malá rychlost. V prvé řadě jsem proto ověřil, zda robot je schopen dosáhnout pohybem dostatečně velké rychlosti, aby světelná tyč byla schopna zobrazení obrazu. Požadované rychlosti dosahoval pouze pohyb PTP. Požadovanou rychlost jsem určil experimentováním se světelnou tyčí, kdy jsem rukou roztáčel tyč.

Kleště se skládají ze dvou čelistí. K čelistem kleští už byly zevnitř připevněny vložky, ovšem nevyhovující k tomu, aby pomocí nich mohla být připevněna tyč. Rozměry rozpětí ve stavu otevřeno, zavřeno a rozměry vložek udávají obrázky. 3.2 a 3.3. U nákresu vložky čárkované čáry značí díru se šroubem o průměru 58 mm.



Obrázek 3.2: Naměřené rozměry kleští



Obrázek 3.3: Naměřené rozměry vložky

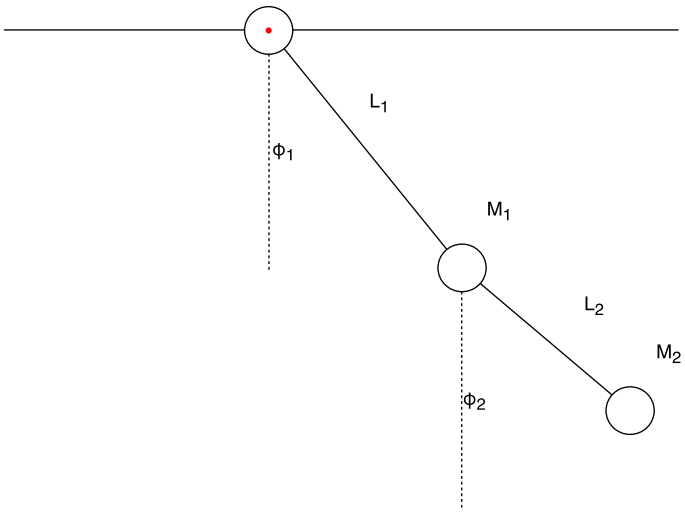
Dále jsem měřil průměr světelné tyče, který byl 24.9 mm. Ke kleštím je tedy potřeba přidělat takovou vložku, aby rozpětí kleští v momentě kdy jsou zavřené, bylo menší průměr tyče (< 24.9 mm) a zároveň, aby rozpětí kleští v momentě, kdy jsou kleště otevřené, bylo větší než průměr tyče. Pokud by se tedy první vložka nechala a vyráběla pouze druhá, její délka by musela být v rozmezí $66.45 < x < 78.65$.

Bohužel se vložky nestihly vyrobit, tudíž nebylo možné pohyb s tyčí připevněné ke kleštím vyzkoušet.

3.1 Návrh modelu

Druhou variantou bylo zavěšení tyče k robotovi, kterému by se odstranily kleště a ke koncovému bodu robota by se uchytila světelná tyč. Protože základním pohybem pro roztočení tyče je pohyb po kružnici a protože každý pohyb je možné přibližně popsat několika pohyby po kružnici (např. přímka může být uvažována jako kružnice s nekonečným poloměrem), tak jsem zavěšení světelné tyče uvažoval jako dvojité kyvadlo, kde prvním kyvadlem

byl poloměr kružnice, po které se koncový bod robota pohyboval a druhým kyvadlem byla světelná tyč. Takto uvažované dvojité kyvadlo se od původního dvojitého kyvadla liší tím, že koncový bod robota není ovlivněn pohybem tyče.



Obrázek 3.4: Dvojité kyvadlo

V tabulce jsou uvedeny parametry dvojitého kyvadla

Značka	Název	Význam parametru
l_1	délka prvního kyvadla	poloměr kruhové trajektorie robota
M_1	hmotnost koncového bodu prvního kyvadla	–
l_2	délka druhého kyvadla	délka tyče
M_2	hmotnost koncového bodu druhého kyvadla	hmotnost tyče
ϕ_1	výchylka prvního kyvadla	poloha koncového bodu
ϕ_2	výchylka druhého kyvadla	výchylka tyče

Tabulka 3.1: Parametry dvojitého kyvadla

Souřadnice koncového bodu prvního kyvadla označím jako $[x_1, y_1]$.
Souřadnice koncového bodu druhého kyvadla označím jako $[x_2, y_2]$.
Pohybová rovnice[[5]] druhého kyvadla je 3.1

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\phi}} - \frac{\partial L}{\partial \phi} = 0 \tag{3.1}$$

kde L je Lagrangián, který je roven

$$L = T - V \tag{3.2}$$

což je rozdíl kinetické energie T a potenciální energie V.

Kinetická energie dvojitého kyvadla je

$$T = \frac{1}{2}m_1 v_1^2 + \frac{1}{2}m_2 v_2^2 \tag{3.3}$$

$$V = m_1 g y_1 + m_2 g y_2 \tag{3.4}$$

Rychlosti uvedené v 3.3 jsou

$$v_1^2 = \dot{x}_1^2 + \dot{y}_1^2 \tag{3.5}$$

$$v_2^2 = \dot{x}_2^2 + \dot{y}_2^2 \tag{3.6}$$

Souřadnice x_1, x_2, y_1 a y_2 jsou rovny

$$x_1 = l_1 \sin(\phi_1) \tag{3.7}$$

$$y_1 = -l_1 \cos(\phi_1) \tag{3.8}$$

$$x_2 = x_1 + l_2 \sin(\phi_2) \tag{3.9}$$

$$y_2 = y_1 - l_2 \cos(\phi_2) \tag{3.10}$$

Po zderivování rovností 3.7 až 3.10

$$\dot{x}_1 = l_1 \dot{\phi}_1 \cos(\phi_1) \tag{3.11}$$

$$\dot{y}_1 = l_1 \dot{\phi}_1 \sin(\phi_1) \tag{3.12}$$

$$\dot{x}_2 = l_1 \dot{\phi}_1 \cos(\phi_1) + l_2 \dot{\phi}_2 \cos(\phi_2) \tag{3.13}$$

$$\dot{y}_2 = l_1 \dot{\phi}_1 \sin(\phi_1) + l_2 \dot{\phi}_2 \sin(\phi_2) \tag{3.14}$$

Rovnosti 3.11 až 3.14 dosadím do vztahů 3.3 a 3.4, které poté mohu dosadit do Lagrangeových rovnic

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\phi}_1} - \frac{\partial L}{\partial \phi_1} = 0 \tag{3.15}$$

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\phi}_2} - \frac{\partial L}{\partial \phi_2} = 0 \tag{3.16}$$

Protože v mém případě první část kyvadla budu řídit externím působením a druhá část ji nijak nebude ovlivňovat, uvažuji pouze rovnici 3.16, která po zderivování a úpravách je níže

$$\ddot{\phi}_2 = -\frac{l_1}{l_2}\ddot{\phi}_1 \cos(\phi_1 - \phi 2) + \frac{l_1}{l_2}\dot{\phi}_1^2 \sin(\phi_1 - \phi_2) - \frac{g}{l_2}\sin(\phi_2) \tag{3.17}$$

Rovnice 3.17 platí pro netlumený pohyb. Protože uvažuji tlumené kyvadlo, je nutné uvažovat i disipativní sílu

$$F_{Dis} = -d_2(\dot{\phi}_2 - \dot{\phi}_1) \tag{3.18}$$

Podle 2. Newtonova zákona

$$m_2\ddot{a}_2 = -d_2(\dot{\phi}_2 - \dot{\phi}_1) \tag{3.19}$$

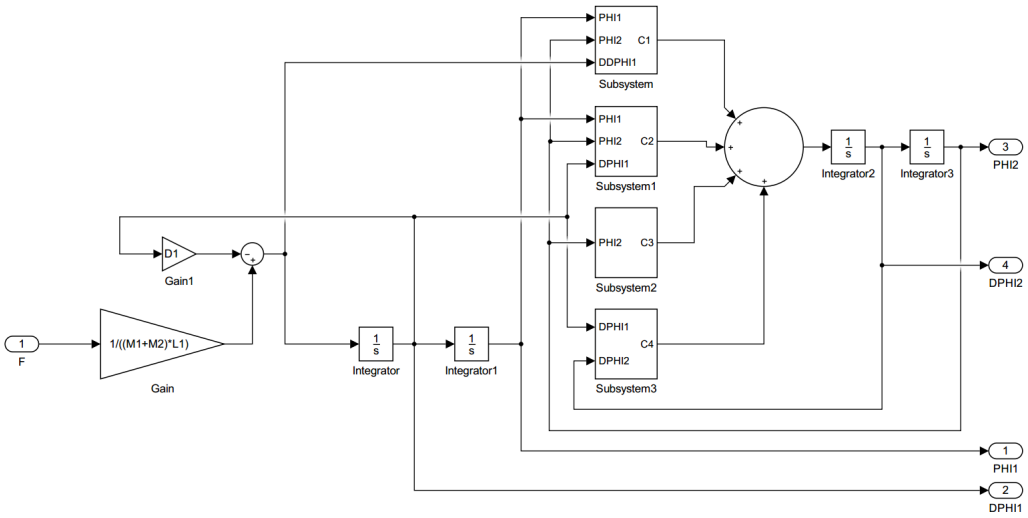
$$m_2\ddot{\phi}_2 l_2 = -d_2(\dot{\phi}_2 - \dot{\phi}_1) \tag{3.20}$$

$$\ddot{\phi}_2 = -\frac{d_2(\dot{\phi}_2 - \dot{\phi}_1)}{m_2 l_2} \tag{3.21}$$

Konečná rovnice je tedy

$$\ddot{\phi}_2 = -\frac{l_1}{l_2}\ddot{\phi}_1 \cos(\phi_1 - \phi 2) + \frac{l_1}{l_2}\dot{phi}_1^2 \sin(\phi_1 - \phi_2) - \frac{g}{l_2}\sin(\phi_2) - \frac{d_2(\dot{\phi}_2 - \dot{\phi}_1)}{m_2 l_2} \tag{3.22}$$

Schéma v Simulinku je na obrázku 3.5.



Obrázek 3.5: Blokové schéma

Subsystémy na obrázku 3.5 představují jednotlivé členy rovnice.

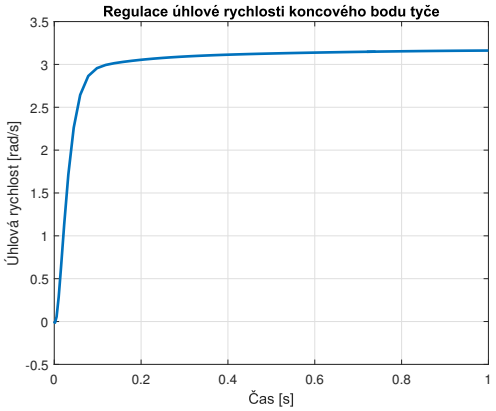
Parametr	Hodnota	Jednotka
l_1	0.1	m
l_2	0.49	m
m_2	0.170	kg
g	9.8	ms^{-2}

Tabulka 3.2: Známé parametry modelu

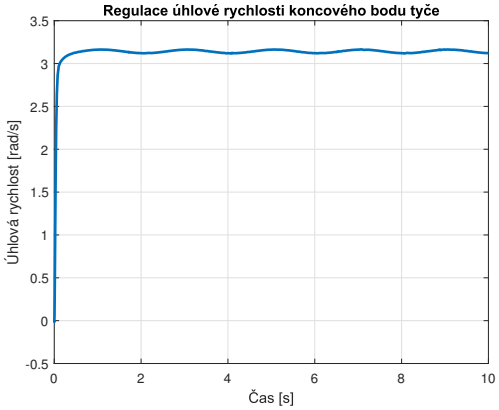
3.2 Rychlost koncového bodu tyče na modelu

Světelná tyč, aby dosáhla požadovaného efektu, potřebuje určitou rychlost. Na základě pokusů s tyčí jsem určil požadovanou úhlovou rychlost $\pi rad/s$. Protože se zatím nevyrobilo uchycení tyče k robotovi, jsou parametry d_1 a d_2 neznámé.

Budu tedy předpokládat $d_1 = 10$ a $d_2 = 10$. K udržení požadované rychlosti použiji PID regulátor. Experimentováním v modelu jsem došel ke konstantám regulátoru PID = [0.5, 4, 0]. Výsledek regulace je zobrazen na grafech níže



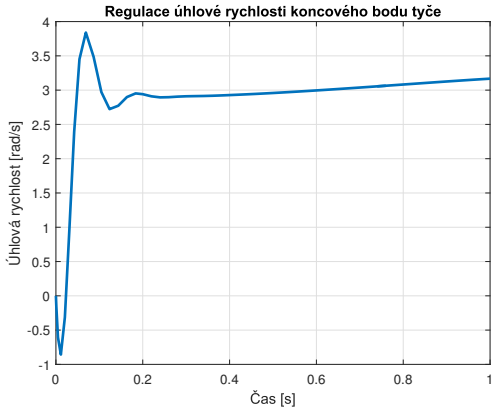
Obrázek 3.6: Regulace úhlové rychlost za 1 sekundu



Obrázek 3.7: Regulace úhlové rychlost za 10 sekund

Model dosáhl relativně rychle požadované hodnoty, kolem které mírně kmitá. Kmitání je způsobeno tíhovým zrychlením, kdy při klesání je koncový bod zrychlován a při stoupání tíhovým zrychlením zpomalován.

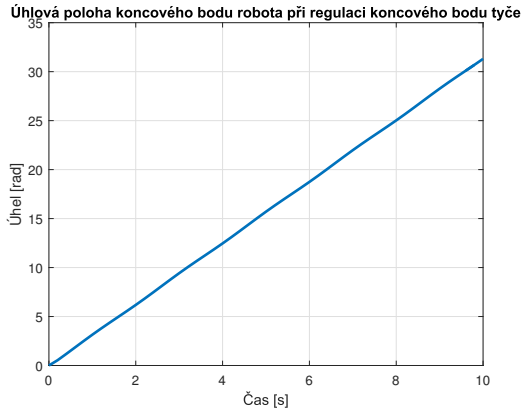
Vygeneroval jsem průběh úhlové rychlosti i pro $d_1 = 1$ a $d_2 = 1$, konstanty PID jsem nastavil opět experimentováním na hodnoty PID = [0.5, 0.05, 0.04]. Zde se mi nepodařilo nastavit konstanty přesně a došlo k výraznému překmitu.



Obrázek 3.8: Regulace úhlové rychlost za 1 sekundu pro konstanty $d_1 = 1$ a $d_2 = 1$

3.3 Vygenerování kartézských souřadnic koncového bodu robota

V první řadě jsem v kartézských souřadnicích robota potřeboval určit střed. Jako střed jsem zvolil [1000, 0, 800]. Tento bod jsem zvolil proto, že v něm nedochází k přetáčení os ani v okolí tohoto bodu není hranice pracovního prostoru.
Souřadnice koncového bodu robota jsem generoval z hodnot úhlu ϕ_1, který vyjadřuje úhlovou polohu koncového bodu robota. Abych bodů nebylo zbytečně moc, je pohyb je vzorkován po 0.2 sekundách. Takovou hodnotu jsem zvolil experimentálně.



Obrázek 3.9: Úhlová poloha koncového bodu robota

Úhel ϕ_1 jsem dále převedl do kartézských souřadnic pomocí Matlabovské funkce `pol2cart` do roviny $y - -z$.

V závěru jsem body uložil do matice, kde každý řádek představuje bod, který je v matici ve tvaru [x y z a b c]. Souřadnice x je konstantní. Souřadnice

a, b, c jsou nastaveny a = 0, b = 90, c = 0, tedy stejně jako v HOME pozici robota. Matice [x y z a b c] je velmi důležitá pro programy uvedené v dalších kapitolách.

Kapitola 4

Program pro plánování trajektorie robota

4.1 Zadání

Dalším úkolem bylo vytvoření programu v Matlabu. Program vygeneruje soubor se zdrojovým kódem (.src) a datový soubor (.dat) pro robota na základě zadaných parametrů, kterými jsou body (např. vygenerované z modelu), do kterých má robot najet a dále rychlosti, kterými má najet do bodů. Pohyb robota je aproximovaný PTP, robot tedy nenajede přímo do určených bodů, pouze se k nim přiblíží. Toto řešení jsem zvolil, protože při neaproximovaném pohybu by pohyb nebyl plynulý, což by nebylo dobré, pokud by body byly blízko u sebe. Program je tedy využitelný v případě, že uživateli jde o plynulost a rychlost pohybu, nikoliv o přesné najetí do požadovaných bodů. Rychlost je ale nakonec ovlivněna hlavně polohou bodů. Pokud body budou nahuštěny blízko u sebe a navíc bude robot muset konat protipohyby, tak výsledný pohyb bude pomalý. Program je vytvořen jako funkce, kde vstupními parametry programu je název programu a dvě matice:

- Matice bodů, kde jsou zadány body, kam má robot najet
- Matice rychlostí, kde je uvedeno, jakou rychlostí má robot najet do bodů stanovených maticí bodů

Počet řádků každé matice je roven počtu bodů, kolem kterých má robot najet.

Matice obsahující body má 6 sloupců, které představují kartézské souřadnice (X, Y, Z, A, B, C)

Matice rychlosti má pouze jeden sloupec. Řádek může mít buď jeden, pak uvedená rychlost bude platit pro všechny body a nebo může mít řádků stejně jako počet bodů. Pak do každého bodu bude robot najíždět s rychlostí uvedenou v matici V.

Program vrací 0, pokud proběhl v pořádku. V případě chyby vrací nenulové číslo. Zvažoval jsem, zda by měl program vracet číslo nebo řetězec obsahující popis chyby. Rozhodl jsem se pro celočíselnou návratovou hodnotu proto, aby

bylo možné tento program využít i v jiném programu, který může jednoduše reagovat na návratové hodnoty hlavní funkce.

4.2 Struktura programu

Na začátku program ověřuje vstupní parametry. Vstupními parametry jsou:

Parametr	Typ	Význam parametru
name	String	název programu
P	$n \times 6$	matice s n body
V	$n \times 1$	matice s n rychlostmi

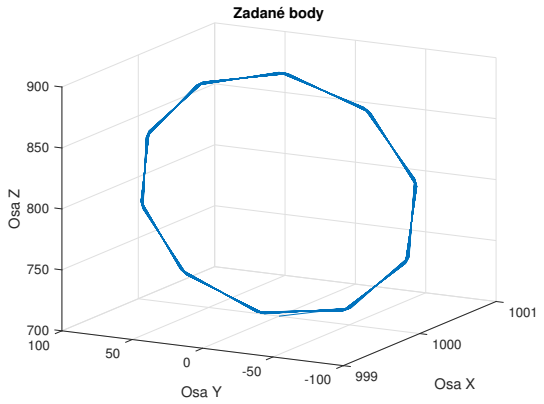
Tabulka 4.1: Vstupní parametry programu pro generování programu

kde n je počet bodů, které programu předávám.

Pokud jsou vstupní proměnné v pořádku, program může začít generovat kód. Program generuje zdrojový kód způsobem, že převezme hlavičku, která je stejná pro všechny programy. Pod ní pak přidává PTP pohyby. Před každý PTP pohyb je uvedena rychlost, kterou má robot najíždět do bodu.

4.3 Vygenerování programu vypočítané trajektorie modelu

Vygenerované kartézské souřadnice koncového bodu robota z předchozí kapitoly jsem vložil do tohoto programu. Program vygeneroval robotický program – vygenerované soubory jsou na CD (více o CD k této práci je v příloze). Body, do kterých robot podle programu najede, jsou zobrazeny na obrázku 4.1. Pro lepší přehled o návaznosti bodů jsou body propojeny čárou.



Obrázek 4.1: Zadané body, do kterých robot najede

Kapitola 5

Přenos parametrů trajektorie do řídicího systému

5.1 Způsob přenosu

K přenosu parametrů jsem se rozhodl využít doplňku Ethernet KRL[4], který umožňuje komunikaci mezi počítačem a řídicím systémem pomocí Ethernetu, kde řídicí systém je klientem a externí systém je serverem.

Komunikace probíhá přes komunikační protokol TCP/IP, kde se externí systém jako server pokusí navázat spojení a čeká na odezvu od řídicího systému. V případě připojení poté server řídicímu systému posílá data. řídicí systém má deklarované dvě globální proměnné. Proměnnou typu Integer, která ukládá počet uložených bodů a datové pole typů FRAME, do kterého se ukládají body. Ke globálním proměnným jsem se rozhodl proto, aby řídicí systém byl schopen si do restartu pamatovat vytvořenou trajektorii. Opětovné spuštění programu nebo spuštění programu, který využívá tuto proměnnou však může způsobit přepsání proměnné.

5.2 Způsob provedení

Zpracoval jsem celkem dvě varianty. V první variantě uživatel předá serveru všechny body najednou a postupně všechny body pošle řídicímu systému, který si body uloží a vykoná spojitý PTP pohyb.

Druhá varianta spočívá v tom, že uživatel zadává postupně body a robot je vykonává. Po dokončení zadávání robot projede celou trajektorii znovu. Program navíc vrátí matici zadaných bodů, uživatel může proto příště zadat tuto matici programu v první variantě a nebo si nechat vygenerovat program generujícím programem, o kterém jsem psal v předchozí kapitole.

5.3 Konfigurace připojení

Ethernetové připojení se konfiguruje pomocí XML souboru v řídicím systému. Podle XML souboru je řídicí systém schopen detekovat příchozí data ze serveru. Velikost vstupního bufferu je nastavena na maximální možnou velikost, je tedy možné z externího systému do řídicího systému poslat až 512 bodů. Server řídicímu systému posílá souřadnice bodu a informaci o tom, jak se má klient zachovat. Pro každý program používám jiný XML soubor. Soubory se liší pouze číslem portu, aby nedošlo ke komunikaci, kde by například server z prvního programu komunikoval s klientem z druhého programu. Takové spojení by jistě vedlo k chybě.

5.4 Přenos všech bodů

V této podkapitole rozeberu podrobně přenos bodů trajektorie do řídicího systému. Výhoda tohoto programu oproti nadcházejícímu je ta, že se přímo předává body. Program tedy lze použít společně s jiným programem, který generuje kartézské souřadnice souřadnice.

Přenos probíhá tak, že jsou serveru předány souřadnice bodů, které tvoří trajektorii a serverový program je poté odešle ve smyčce řídicímu systému. Celý přenos netrvá dlouho.

řídicí systém poté přijme 2 proměnné, první je struktura Frame a druhá je celočíselná hodnota (Integer), která udává, zda daný bod je poslední. Server s klientem se připojují k portu 6009.

5.4.1 Serverová část

Serverová část je naprogramována v MATLABu. Podmínkou správné funkčnosti je matlabovský toolbox Instrument Control Toolbox, který umožňuje TCP připojení. Na začátku programu pomocí funkce tcpip server naslouchá na zadané IP a portu, na který se klient pokusí připojit.

Před připojením server ověří vstupní parametr. V případě chybně zadaného parametru server vrátí chybu. V případě, že spojení bude úspěšné a program odešle všechny body. návratovou hodnotou bude opět matice P.

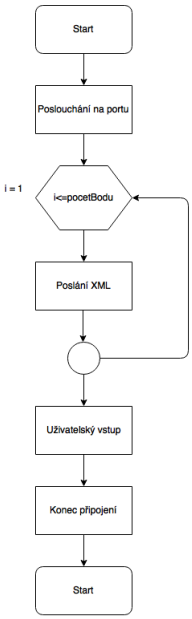
V případě úspěšného připojení server odešle řídicímu systému (klientovi) data ve formátu XML, který odešle ve smyčce souřadnice bodů, které klient přijímá jako FRAME, a celočíselnou hodnotu, která se rozdílná pouze při odeslání posledního FRAME. Díky tomu klient pozná, že FRAME je poslední a další už nedostane.

Po odeslání dat server počká na uživatelský vstup, kdy pak ukončí spojení. Ten je tu z důvodu, aby server neukončil spojení dříve než klient. Jinak by

Návratová hodnota	Popis
10	Vstupní parametr neexistuje
11	Vstupní parametr není číselný
12	Vstupní parametr nemá 6 sloupců
13	Vstupní parametr má více než 512 řádků
14	Prvky vstupního parametru jsou mimo rozsah

Tabulka 5.1: Návratové hodnoty programu

program na straně klienta skončil chybou.



Obrázek 5.1: Vývojový diagram serveru

5.4.2 Klientská část

Klientská část je robotický program, který využívá globální proměnné, kterými je pole FRAMEů uchovávající souřadnice bodů a celočíselná proměnná, do které se ukládá počet bodů.

Na začátku programu je deklarovaná proměnná typu EKI STATUS, což je struktura. Elementy této struktury jsou zobrazeny níže.

Robot najede do domovského bodu a pokusí se připojit k serveru. V případě úspěšného připojení ve smyčce začne přijímat data od serveru a ukládat do FRAME pole. Po načtení posledního bodu začne vykonávat aproximovaný PTP pohyb, kdy postupně projede body uložené ve FRAME poli. V případě neúspěšného připojení program rovnou přejde k vykonávání pohybů, které má uložené ve FRAME poli, například z minulého běhu, které si FRAME pole pamatuje díky tomu, že je globální.

Název	Význam
Buff	Počet prvků v bufferu
Read	Počet prvků přečtených z bufferu
Msg No	Číslo chyby. Pokud nenastane vrací nulu
Connected	Informace, zda bylo navázáno spojení

Tabulka 5.2: Elementy struktury EKI STATUS

Na konci programu robot najede zpět do HOME pozice a ukončí se.

5.5 Přenos bodů postupně

V této variantě uživatel zadává body postupně. Uživatel zadá bod a robot do bodu najede. Uživatel může kdykoliv zadávání ukončit nebo se vrátit do předposledního bodu a poslední bod tak změnit. Pokud se uživatel bude chtít vrátit po zadání jen jednoho bodu, vrátí se do HOME pozice. Tím je ošetřena možnost, aby se program nedostal před začátek pole, což by pravděpodobně skončilo chybou.

5.5.1 Serverová část

Server je podobný serveru v první variantě. Důležitý rozdíl je ten, že ve smyčce server čeká na uživatelský vstup, kterým je řetězec. Tabulka 5.3 zobrazuje přípustné řetězce.

Vstup	Akce
6-tice bodů	Vykoná pohyb
"vratit"	Robot najede do předchozího bodu
"konec"	Zadávání se ukončí

Tabulka 5.3: Možné uživatelské vstupy

Návratovou hodnotou serveru je matice zadaných bodů, která je ve stejném formátu jako vstupní matice u předchozí varianty. Pracné zadávání bod po bodu tak může být jen jednou a při příštím odeslání lze využít prvního programu, který zadané body odešle řídicímu systému naráz.

5.5.2 Klientská část

Klientská část je program robota. Stejně jako v předchozí variantě, program využívá globální proměnné a deklaruje proměnnou typu EKI STATUS. Rozdíl je až ve smyčce programu, kde se na základě uživatelského vstupu na straně serveru klientský program rozhoduje, kterou část kódu vykoná.

Oproti předchozí variantě je v klientovi návrat do předchozího bodu. Návrat do předchozího bodu je řešen snížením hodnoty čítací proměnné, čímž se v poli načte předchozí bod, do kterého robot najede. V případě, že by se tato proměnná snížila na hodnotu menší než 1, robot najede do HOME pozice.

5.6 Shrnutí

Formát bodů je u obou variant stejný, je tedy možné matici vygenerovanou druhým programem předat jako vstupní parametr prvnímu programu.

První program přijímá body najednou, v běhu nevyžaduje akci uživatele, což je výhodné v případě, že jiný program už vygeneroval trajektorii. V mém případě ji vygeneroval model světelné tyče zavěšené ke koncovému bodu robota. Druhý program je vhodný v situaci, kdy chce uživatel zadat body postupně sám.

Kapitola 6

Propojení řízení robota s PLC

Součástí pracoviště je i vizualizační panel. Součástí panelu je i PLC.

PLC je propojený s robotem. Řízení přes PLC je možné v režimu EXT AUT, kde je potřeba spustit program cell.src, což je program, který spouští ostatní programy robota, který je v programu cell.src přiřazeno číslo. Číslo programu posílá řídícímu systému PLC. Ke spuštění programu je potřeba nastavit tagy na určité hodnoty.

Tag	Hodnota
RQ_PROGRAM_ENDED_ACK	FALSE
RQ_PGNO_VALID	FALSE
RQ_EXT_START	TRUE
RQ_DRIVES_ON	TRUE
RQ_PGNO	číslo programu (bajt)
RQ_PGNO_VALID	TRUE

Tabulka 6.1: Nastavení tagů ke spuštění programu

Po ukončení programu se ukončení potvrdí nastavením RQ_PROGRAM_ENDED_ACK na TRUE.

Pomocí vizualizačního panelu uživatel nemusí tyto tagy nastavovat. O nastavení tagů se stará program v PLC, který při vyvolání určité události, například stisknutí tlačítka, vyvolá potřebné funkce, kterými jsou tagy nastaveny na potřebné hodnoty. Podrobnější popis tvorby uživatelského rozhraní a nastavení událostí je uvedeno v příloze.

Kapitola 7

Závěr

V této práci jsem se seznámil s principy programování robota. Na základě těchto poznatků jsem vytvořil programy, pro generování trajektorie a její přenos do řídicího systému robota.

Vytvořil jsem model světelné tyče Visual Poi V3 zavěšené ke koncovému bodu robota. Z vytvořeného modelu je možné generovat kartézské souřadnice, které pak lze předat vytvořeným programům, které pak pošlou souřadnice řídicímu systému robota nebo přímo vygenerují program pro robota, který stačí přenést do řídicího systému a poté spustit.

Nakonec jsem se seznámil se systémem PLC a vytvořil jsem uživatelské rozhraní, z kterého je možné spouštět programy z řídicího systému.

Na závěr nutno dodat, že uspořádání pracoviště je dle mého názoru nevhodné, což se projeví v případě, kdy je narušen skenovaný prostor a je nutné robota zprovoznit potvrzujícím tlačítkem, které se nachází na druhé straně pracoviště. Navíc je průchozí ulička mezi panelem s tlačítkem a ovládacím zařízením robota (SmartPADem) skenována, tudíž po stisknutí potvrzujícího tlačítka se pracoviště musí obcházet. Tento problém se objevuje i v případě, když chci ovládat robota pomocí PLC.

Příloha A

Literatura

[1] Visual poi v3. `http://lighttoys.cz/store/led-products/visual-poi-staff/`, [cit. 2016-05-24]. délka a hmotnost tyče.

[2] KUKA Roboter GmbH. *KR C4*. 2012.

[3] KUKA Roboter GmbH. *KUKA System Software 8.2 Operating and Programming Instructions for System Integrators*. 2012.

[4] KUKA Roboter GmbH. *KUKA.Ethernet KRL 2.1 For KUKA System Software 8.2*. 2012.

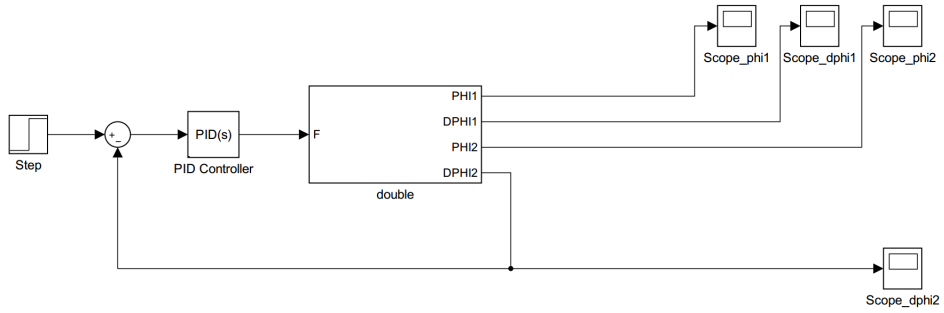
[5] Petr Kulhánek. Teoretická mechanika: Studijní text pro doktorské studium. `http://www.aldebaran.cz/studium/mechanika.pdf`, 2011 [cit. 2016-05-10].

[6] LightToys. *Visual Poi V3 User manual*.

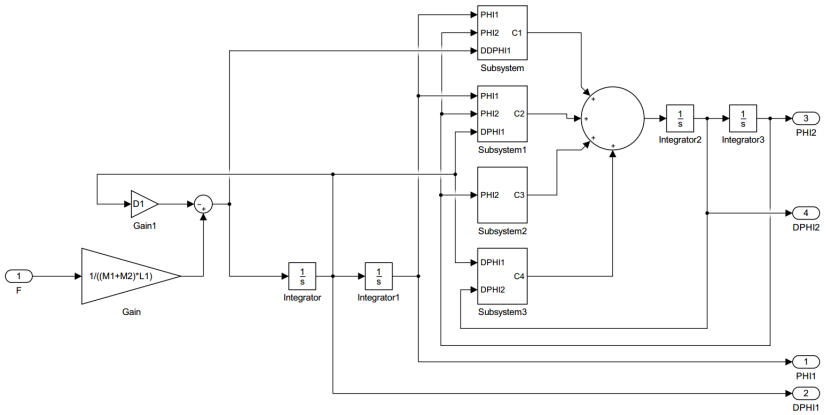
Příloha B

Simulinkové schéma

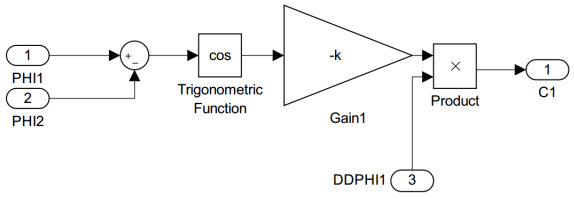
V této příloze je kompletní schéma ke kapitole 3



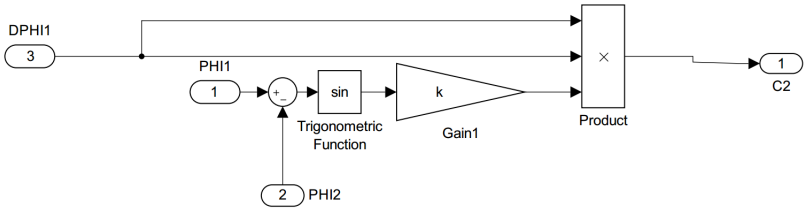
Obrázek B.1: Model světelné tyče s PID regulátorem



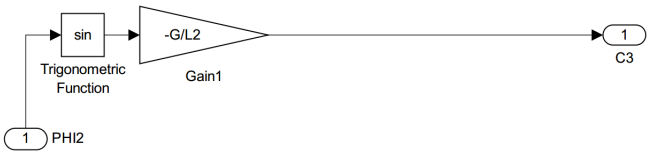
Obrázek B.2: Model světelné tyče



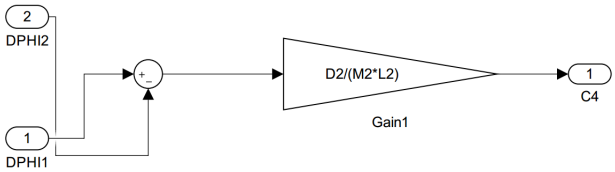
Obrázek B.3: Podsystém C1



Obrázek B.4: Podsystém C2



Obrázek B.5: Podsystém C3

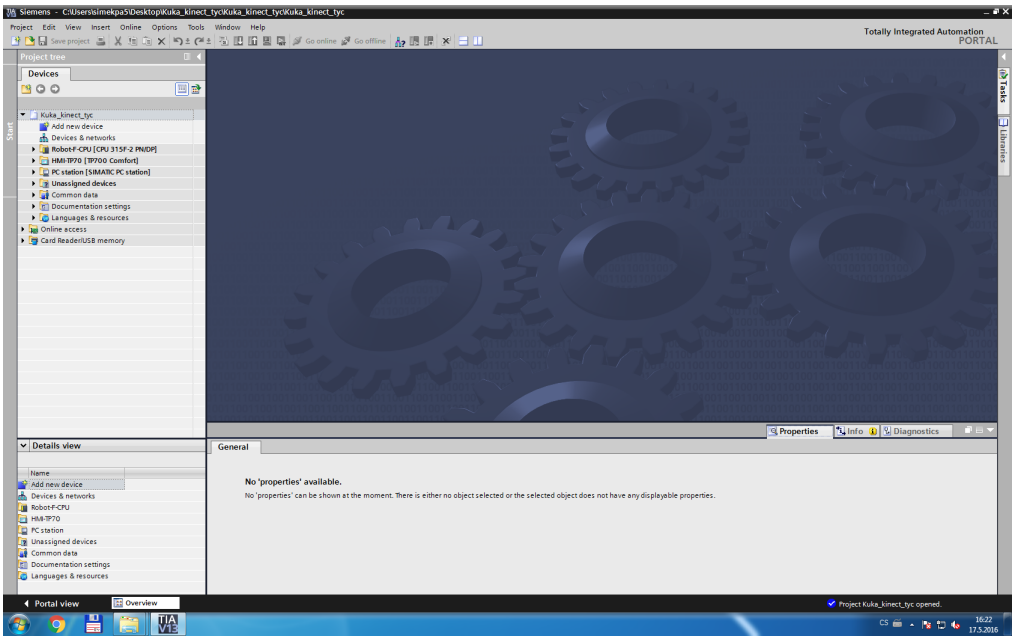


Obrázek B.6: Podsystém C4

Příloha C

Vytvoření uživatelského rozhraní

Uživatelské rozhraní se tvoří v programu Totally Integrated Automation Portal (zkráceně TIA Portal). Po spuštění programu a výběru projektu vypadá obrazovka takto:



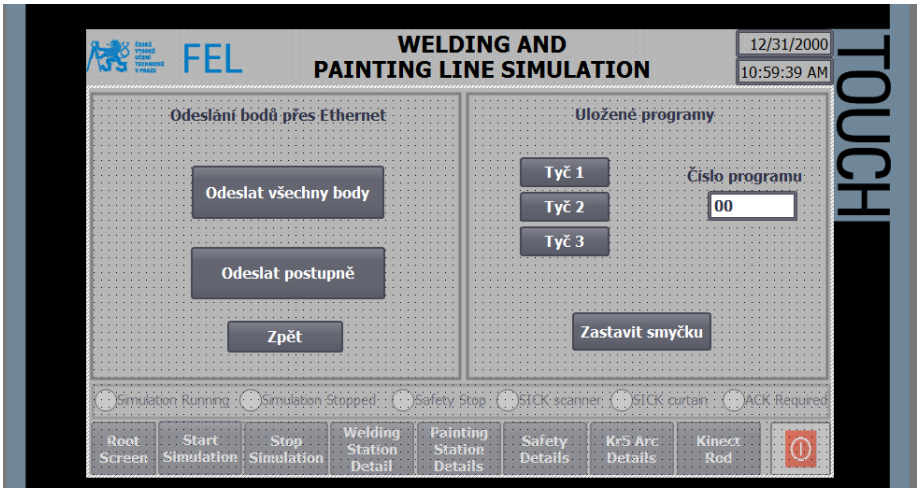
Obrázek C.1: Hlavní obrazovka TIA Portal

Vlevo je výběr položek, kde je možné programovat PLC nebo vytvářet uživatelské rozhraní.

Uživatelské rozhraní je uloženo pod položkami HMI-TP700 → Screens → Tyc. Po kliknutí se zobrazí editor grafického rozhraní s možností editací obrazovky.

V editoru obrazovky se nastavují a vytvářejí různé komponenty jako tlačítka, popisky, textová pole, rámečky a jiné.

Umístění komponent ovšem nestačí, dále je potřeba i nastavení různých

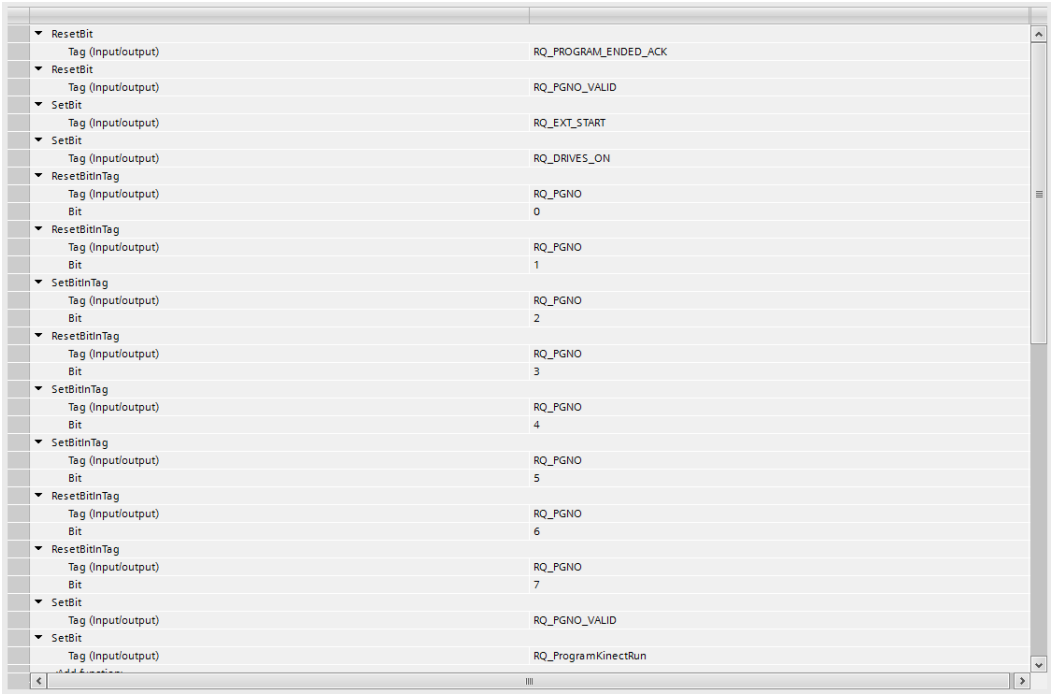


Obrázek C.2: Editor obrazovky

událostí, které se vyvolají např. při stisknutí tlačítka. Taková událost se nazývá Click. V události se pak nastavují funkce, které jsou volány, pokud je na tlačítko kliknuto. Základními funkcemi je SetBit, který Tagu nastaví logickou 1 a ResetBit, který Tagu nastaví logickou 0. Tag je adresa v paměti, kde je uložená nějaká hodnota. Tagem také může být vstup nebo výstup PLC.

Dalšími funkcemi jsou SetBitInTag a ResetBitInTag, kde se kromě Tagu stanoví i pozice bitu. Tyto funkce se využívají u vícebitových Tagů (například u bajtových).

Konečné uživatelské rozhraní vypadá stejně jako na obrázku C.3. Po levé straně se spouští programy komunikující přes Ethernet. Na pravé straně jsou další programy, které mají pevně daný pohyb, který probíhá ve smyčce. Opustit smyčku lze tlačítkem Zastavit smyčku.



Obrázek C.3: Nastavení událostí

Příloha D

Soubory na přiloženém CD

```
MAPA
|   model tyče zavěšené k robotovi.pdf  Simulinkové schéma modelu v .pdf (kapitola 2)
|   Mapa.txt                           Textový soubor zobrazující adresářovou strukturu
|
+---programy pro robota                Robotické programy k přenosu trajektorie přes Ethernet (Kapitola 5)
|   |   prenosNajednou.dat             Klient pro přenos trajektorie přes Ethernet (Kapitola 5.4.2)
|   |   prenosNajednou.src
|   |   prenosPostupne.dat             Klient pro přenos trajektorie přes Ethernet (Kapitola 5.5.2)
|   |   prenosPostupne.src
|   |
|   \---Vygenerovaný program           Vygenerovaný program podle vypočtené trajektorie (Kapitola 4.3)
|           model.dat
|           model.src
|
\---programy v Matlabu
    |   gen_model_output.m             Matlab skript, který převede úhel phi1 na kartézské souřadnice pro robota
    |   gen_program_model.m            Funkce, která propojuje skript gen_model_output.m s program_generator (níže)
    |   MODEL.slx                      Simulinkové schéma modelu v .slx (Kapitola 2)
    |   PARAMS.m                       Parametry modelu (Kapitola 2)
    |   prenosNajednou.m               Server pro přenos trajektorie přes Ethernet (Kapitola 5.4.1)
    |   prenosPostupne.m               Server pro přenos trajektorie přes Ethernet (Kapitola 5.5.1)
    |
    \---program_generator              Program generující robotické programy (Kapitola 4)
        |   blabla.txt
        |   end.txt
        |   gen_dat.m
        |   gen_program.m
        |   gen_program_dat.m
        |   gen_src.m
        |   head.txt
        |   headdat.txt
        |   main.m
        |   read_file.m
        |   write_file.m
        |
        \---program                    Sem jsou ukládány vygenerované programy
```

Obrázek D.1: Adresářová struktura přiloženého CD