VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

ONLINE OVLÁDÁNÍ ROBOTICKÉHO SCARA MANIPULÁTORU

ONLINE CONTROL OF SCARA ROBOTIC MANIPULATOR

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE DAVID PAŘÍK

AUTHOR

VEDOUCÍ PRÁCE Ing. ADAM CHROMÝ

SUPERVISOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student: David Pařík ***ID:*** 146923
Ročník: 3 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Online ovládání robotického SCARA manipulátoru

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je instalovat a oživit robotický SCARA manipulátor Epson, vytvořit pro něj elektromagnetický úchop, vybavit jej programovými prostředky umožňujícími jeho online ovládání a dokázat jejich funkčnost pomocí demonstrační aplikace.

1. Seznamte se s předloženým robotickým SCARA manipulátorem a se skriptovacím jazykem SPEL III pro ovládání robotu Epson.
2. Navrhněte uspořádání a zapojení hardwarových součástí a definujte požadavky na software s cílem ovládat online robotický manipulátor z počítače.
3. Proveďte instalaci a oživení manipulátoru.
4. Dle návrhu z bodu 2 vytvořte softwarový ovladač s rozhraním pro online ovládání robotu.
5. Navrhněte a sestavte magnetický úchop pro manipulátor.
6. Vytvořte aplikaci pro demonstraci schopností manipulátoru dle zadání vedoucího.

DOPORUČENÁ LITERATURA:

NOF, Shimon Y. Handbook of industrial robotics. 2nd ed. New York: John Wiley, c1999, xxii, 1348 p. ISBN 04-711-7783-0.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 25.5.2015

Vedoucí práce: Ing. Adam Chromý
Konzultanti bakalářské práce:

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

ABSTRAKT

Bakalářská práce se zabývá SCARA robotem značky Epson. Teoretická část popisuje části a funkce manipulátoru H554BN a jeho kontroléru SRC 310. Je zde taky zahrnuta kapitola o jazyku SPEL III, sloužícímu k programování robotu.
Praktická část se týká zprovozěnní robotu. Zaměřuje se na vytvoření magnetického úchopu a třídy v C# pro ovládání robotu pomocí PC. Závěrečný cíl je demonstrovat schopnosti manipulátoru na nějakém zajímavém úkolu.

KLÍČOVÁ SLOVA

Robot,EPSON,Manipulátor,SCARA,Magnetický úchop,SRC 310,H554BN

ABSTRACT

Bachelor's thesis deals with Epson SCARA robot. Theoretical part of this project describes parts and functions of manipulator H554BN and its contorller SRC 310. There is also included chapter about the SPEL III robot programming language.
Practical part is about putting the robot into operation. It's focused on the creation of magnetic grip and C# class for controlling robot via PC. The final aim is to demonstrate manipulators abilities in some interesting task.

KEYWORDS

Robot,EPSON,Manipulator,SCARA,Magnetic grip,SRC 310,H554BN

PAŘÍK, David *ONLINE OVLÁDÁNÍ ROBOTICKÉHO SCARA MANIPULÁTORU*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2014. 77 s. Vedoucí práce byl Ing. Adam Chromý,

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „ONLINE OVLÁDÁNÍ ROBOTICKÉHO SCARA MANIPULÁTORU“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Adamu Chromému za odborné vedení a trpělivost při konzultaci.

Brno . . . . . .

. . . . . .
(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Robotické manipulátory typu SCARA jsou v současné době velice oblíbené. Svým uspořádáním se velmi dobře hodí na přemisťování předmětů z běžícího pásu. V této práci je popsán čtyřosý manipulátor H554BN značky EPSON, řízený pomocí kontroléru SRC 310. Jedná se o starší, výrobcem již nepodporovaná zařízení jejichž aktuálními alternativami jsou roboty série G s řídící jednotkou RC 180 [19]. První kapitola je věnována teoretickému popisu robotu. Jsou v ní vysvětleny funkce jednotlivých částí manipulátoru a kontroléru, jehož prostřednictví se robot řídí. Dále je zde popsán programovací jazyk SPEL III, v němž jsou psány programy nahrávané do kontorléru. Následují praktické úseky práce, mezi kterými je kapitola popisující návrh a konstrukci magnetického úchopu. Další je kapitola, ve které se píše o zprovoznění robotu. Ta je následována popisem vytvořeného ovladače pro řízení robotu s využitím PC. Finální kapitola je věnována demonstrační úloze, prezentující schopnosti manipulátoru.

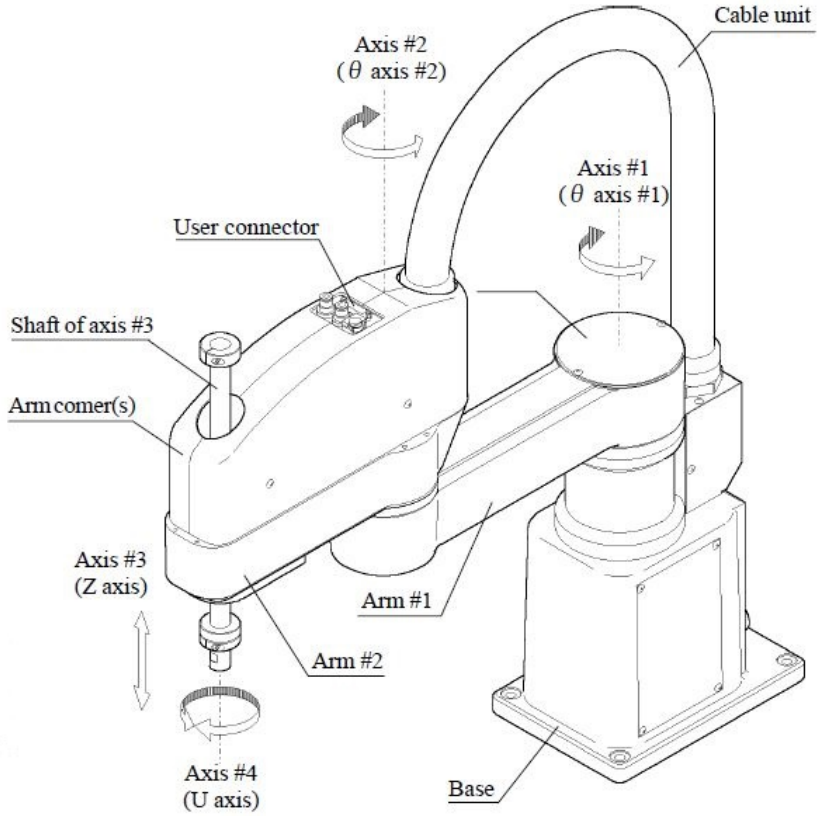
1 POPIS ROBOTU EPSON

1.1 Manipulátor H554BN

1.1.1 Uspořádání robotu

Manipulátor je uspořádán podle konceptu SCARA, jehož první prototyp sestavil roku 1978 profesor Hiroshi Makino. Tento typ robotu se vyznačuje svou jednoduchostí, kdy s minimem pohybu dokáže operovat s vysokou rychlostí a přesností. Ačkoliv byl představen již roku 1981, stále nabízí nejlepší poměr ceny a výkonu, čímž si zajištuje svou popularitu. [13]

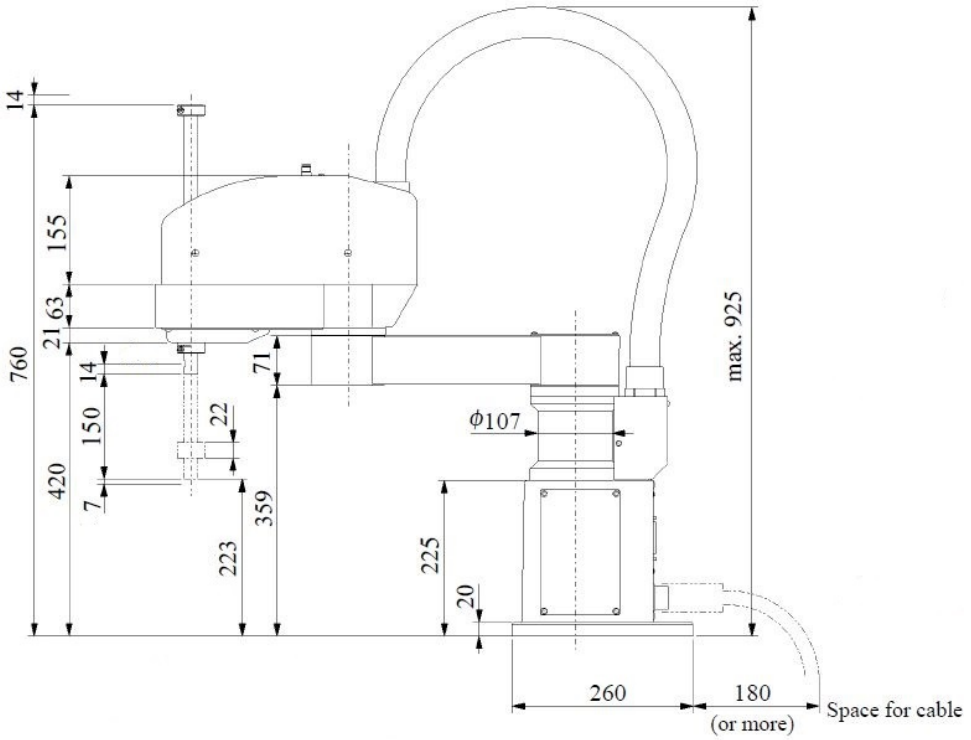
Robot má celkem 4 osy pohybu první (Axis#1) a druhá (Axis#2) osa obstarává horizontální pohyb, zatímco třetí osa (Axis#3) vertikální. Čtvrtá osa (Axis#4) pak umožnuje rotaci uchopeného předmětu. Tyto osy spolu s popisky jednotlivých částí manipulátoru vidíme na obrázku 1.1.



Obr. 1.1: Osy manipulátoru [10].

1.1.2 Rozměry a rozsah pohybu

Rozměry robotu jsou dány jeho typem, kdy hodnota 55 v pojmenování udává délku prvního ramene, která je 550 mm [10]. Dále se v našem případě jedná o typ s rozsahem třetí osy 150 mm. Rozměry jednotlivých částí jsou uvedeny v milimetrech na obrázku 1.2.

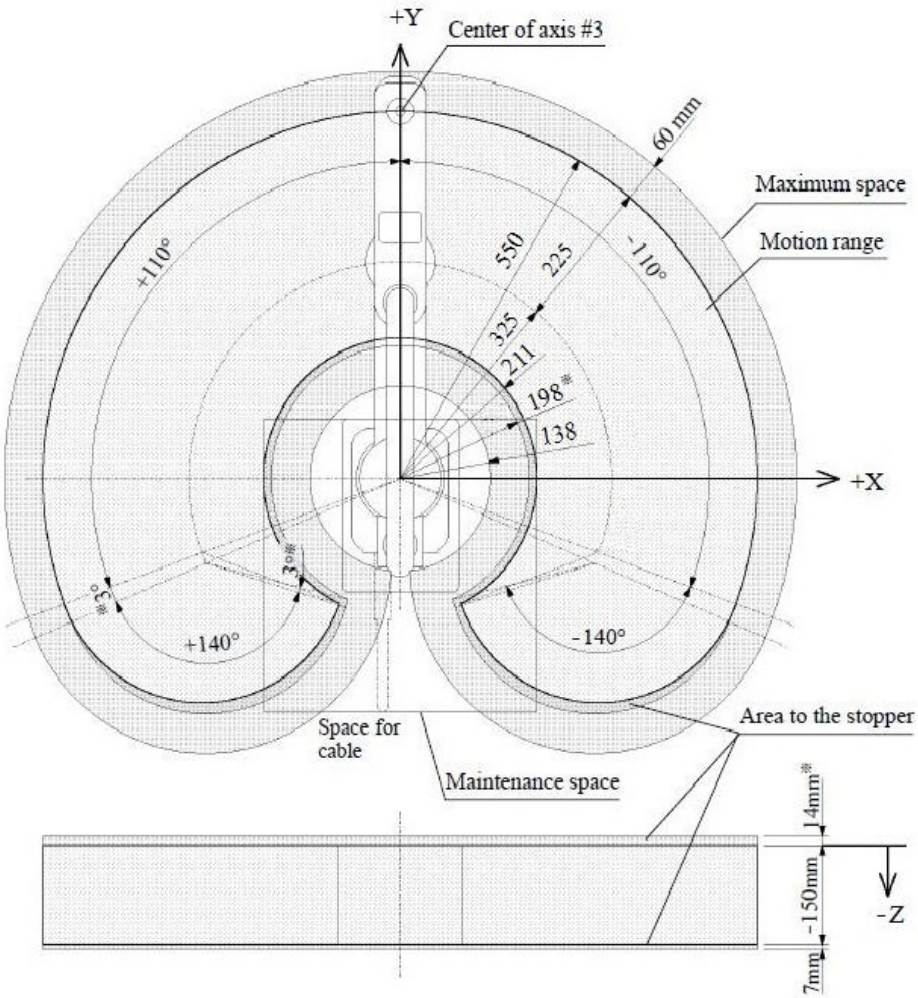


Obr. 1.2: Rozměry manipulátoru (mm) [11].

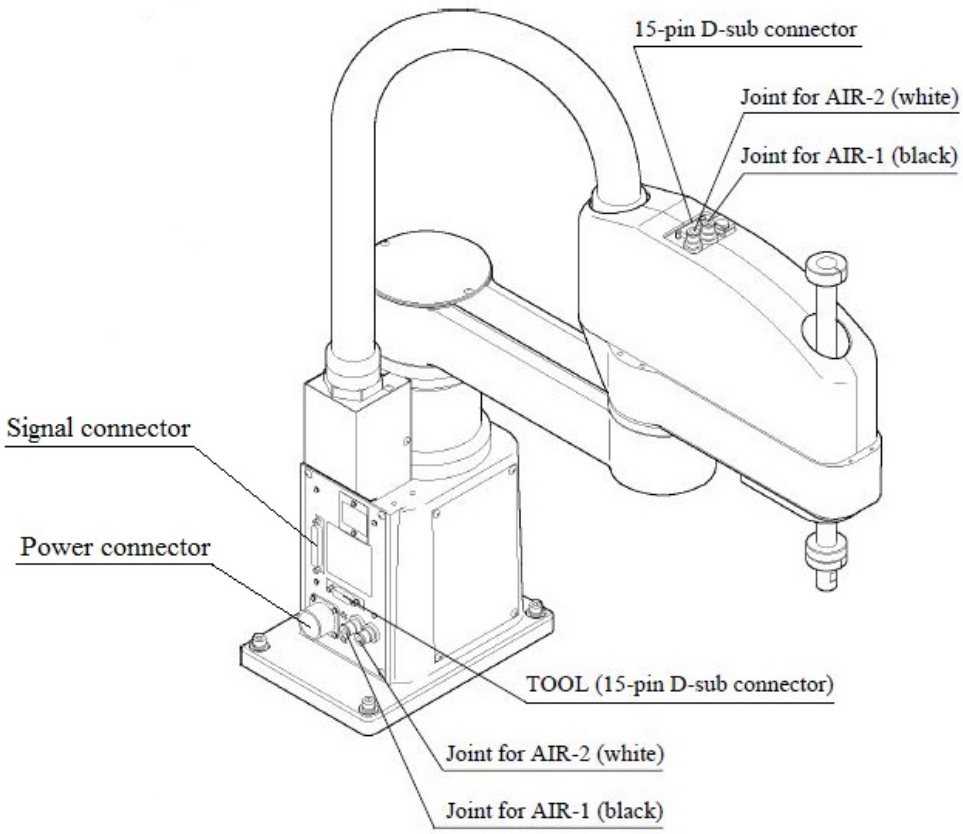
Na obrázku 1.3 vidíme rozsah pohybu robotu, přičemž hvězdička (∗) vyznačuje pozici mechanické zarážky.

1.1.3 Konektory a uživatelská vedení

Jak je uvedeno v [2], Uživatelká zařízení umístěná na manipulátoru mohou pro přívod elektřiny využít 15-ti pinové konektory propojující spodní a vrchní část manipulátoru viz. 1.4 . Maximální napětí je 30 V AC/DC a povolený proud až 1 A. Dvě vzduchové trubice umístěné poblíž 15-ti pinového konektoru mohou přivádět vzduch pod tlakem až 0.59 MPa. Na zadní straně základny manipulátoru se dále nacházejí konektory pro přívod energie a řídícího signály, které jsou taktéž vyznačeny na obrázku 1.4.



Obr. 1.3: Rozsah pohybu manipulátoru (mm) [12].

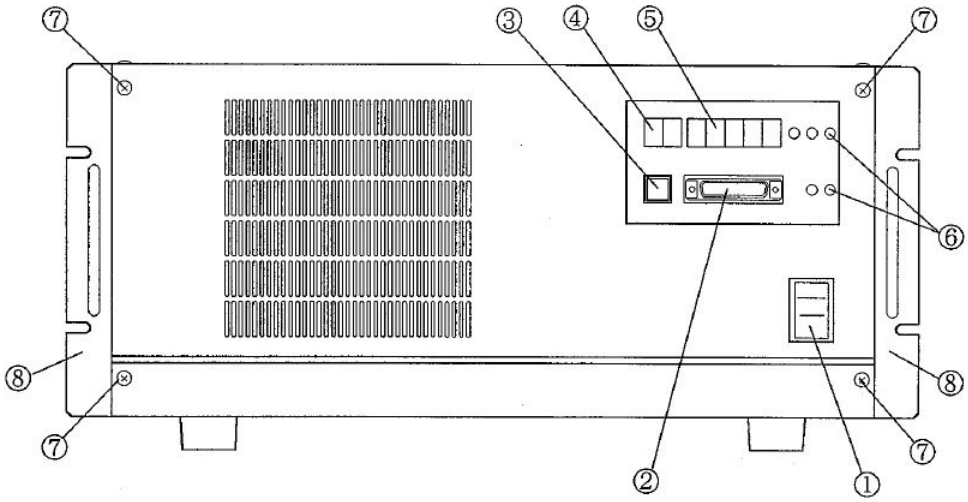


Obr. 1.4: Manipulátor - vedení elektřiny a vzduchu [8].

1.2 Kontrolér SRC 310

V následujících dvou podkapitolách je popsán čelní a zadní panel řídící jednotky dle [6].

1.2.1 Čelní panel



Obr. 1.5: Čelní panel kontroléru [6].

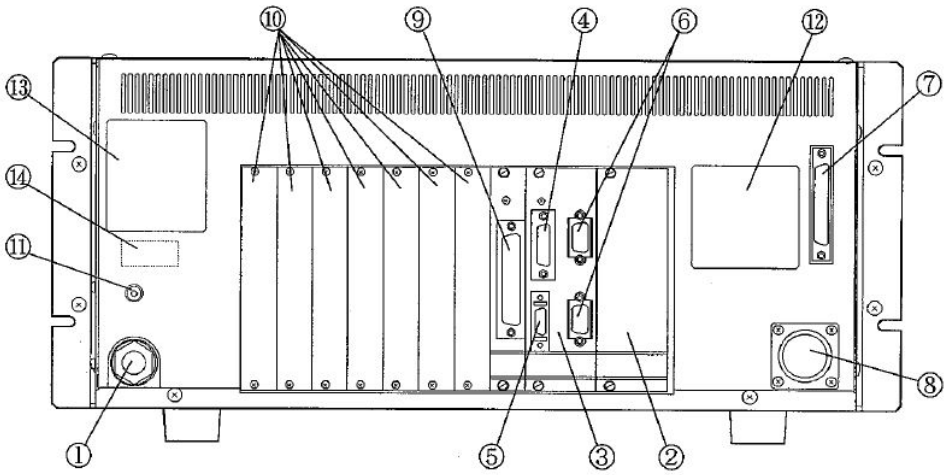
1. Vypínač napájení
2. TEACH port:
   Jedná se o RS-232C port sloužící k připojení programovací jednotky, kterou může být například PC nebo TP-320J [1]. Jeho jeho modulová rychlost je 9600 bitů za sekundu a při jeho odpojení je robot zastaven nouzovou brzdou.
3. E. STOP CANCEL přepínač:
   Slouží k vyřazení nouzového zastavení při odpojení TEACH konektoru.
4. PRG.No.LED:
   Segmentový displej zobrazující číslo programu. Pokud dojde k chybě, první číslice zobrazí číslo osy, které se chyba týká, nebo číslo chybného požadavku.
5. LINE NO./STATUS LED:
   Zobrazuje číslo řádku spuštěného programu. Dále zobrazuje číslo vyskytlé chyby.
6. LED indikátor:
   Diody, které svým rozsvícením signalizují následující informace:
   - E.STOP - je spuštěno nouzové zastavení.

[1] Malé zařízení pro provádění učících operací

- SAFE GUARD - byla otevřena pojistka.
- TEACH - zapnutí TEACH modu.
- AUTO - zapnutí AUTO modu.
- S.ERR - CPU je nefunkční.

7. Šrouby upevňující čelní panel.
8. Konzola k upevnění kontroléru

1.2.2 Zadní panel



Obr. 1.6: Zadní panel kontroléru [6].

1. Napájecí kabel
2. PSU deska:
   Deska plošných spojů pro napájení kontroléru manipulátoru a periferních jednotek.
3. REMOTE deska:
   Deska se čtyřmi konektory REMOTE1, REMOTE2, RS-232C #20 a #21.
4. REMOTE1: Konektor pro vstupy nouzového zastavení a pojistky. Nouzové zastavení způsobí zastavení chodu robotu, zatímco pojistka zastaví program a uvede robot do low power modu.
5. REMOTE2:
   Konektor určený k připojení operační jednotky OPU-300 nebo OPU-320, kterou je možno použít v AUTO modu. Pokud není jednotka připojena, je třeba ke konektoru připojit okruh vypínající nouzové zastavení.
6. RS-232C #20, #21:
   Konektor pro sériové rozhraní RS-232C. Využívá se pro komunikaci s roboty, periferními jednotkami nebo počítači.

7. M/C SIGNAL:
   Slouží k přenosu signálů z robotu, vysílaných např. kalibračním senzorem, nebo enkodérem motoru.
8. M/C POWER:
   Konektor pro napájení manipulátoru.
9. I/O-1:
   Padesátipinový D-sub konektor s 16-ti vstupy a 16-ti výstupy.
10. Volitelné slot:
    Sedm slotů, ke kterým je možno připojit dodatečné desky se vstupy a výstupy, nebo sériovými porty.
11. PE(F-GND) terminál:
    Terminál pro přivedení přídavného zemnícího vodiče.
12. M.CODE:
    Štítek s kódem odpovídajícím danému manipulátoru.
13. Sériové číslo.
14. MT štítek:
    Štítek se specifikačním číslem, které mají roboty se speciálními požadavky na údržbu.

1.2.3 Bezpečnostní prvky

Robot má řadu bezpečnostních prvků, které zajišťují ochranu uživatelů. Jednotlivé prvky jsou popsány v následujících podkapitolách, čerpajících z [15].

Low Power a High Power módy

Motory robotu mají nízkosilový a vysokosilový mód. V nízkosilovém (Low Power) módu je omezena rychlost pohybu robotu a také jeho kroutící moment. V High Power módu je poté možné operovat s robotem se zvolenou rychlostí a kroutivou silou.

Robot je automaticky přepnut do Low Power módu kdykoliv je kontrolér resetován, nebo je přepnut TEACH a AUTO mód, popsaný v následující kapitole. Do Low Power módu je robot uveden také při otevření pojistky, o které je psáno v následující podkapitole.

Pojistka

Vnitřní okruh s přepínačem pojistky je napojen na piny 5 a 12 konektoru REMOTE1 [3]. Spuštění bezpečnostní funkce, jakou je třeba vstoupení do Low Power módu, se uskuteční při otevření pojistky.

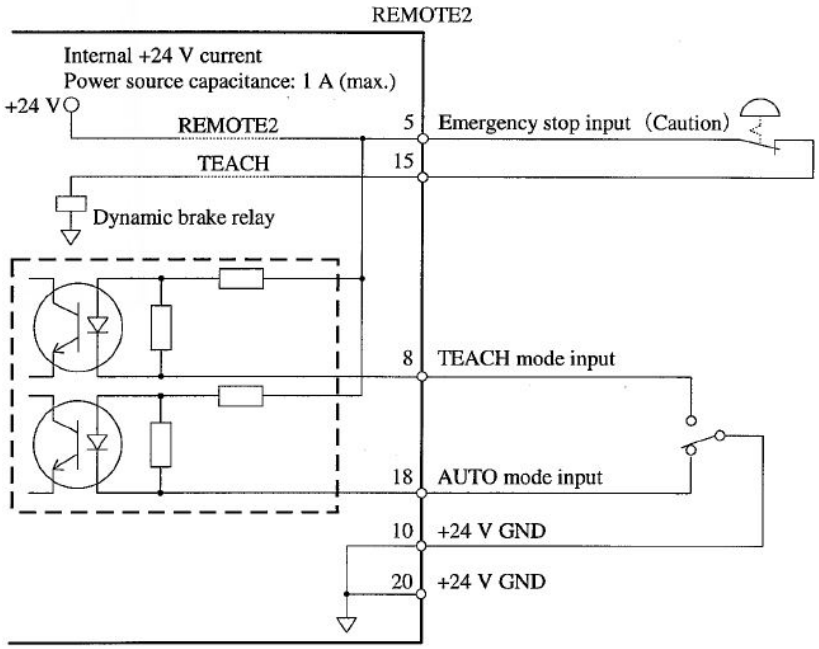
Funkce pojistky se liší pro jednotlivé operační módy (TEACH/AUTO). To jak pojistka v daných módech funguje je popsáno v kapitole: Operační módy kontroléru.

Nouzová brzda

Na kontroléru se nacházejí vstupní terminály pro okruh s nouzovým tlačítkem. Jeho stisknutím dojde k rozpojení, které vyvolá odpojení napájení motorů a okamžité zastavení robotu pomocí dynamické brzdy. Vstupy pro nouzovou brzdu se nacházejí na konektorech REMOTE1, REMOTE2 a TEACH [4].

1.2.4 Operační módy kontroléru

Jak je uvedeno v [14], SRC 310 je možné přepínat do dvou módů, na základě operací které potřebujeme vykonávat. Přepínání mezi módy je možné dvěma způsoby. Jedním je přepnutí pomocí operační jednoty OPU-300. Pokud ji nevyužíváme, je možné módy přepínat posláním signálu na příslušný pin REMOTE2 konektoru. V tomto případě je dále potřeba připojit na konektor okruh s tlačítkem nouzového zastavení. Příklad možného zapojení je vyobrazen na obr. 1.7.



Obr. 1.7: Příklad zapojení pro přepínání módů [5].

TEACH

Tento mód je využíván pro učení programování a ladění robotu. Instrukce jsou robotu posílány pomocí PC, nebo TP-320J skrze TEACH konektor. Modulová rychlost

musí být v PC nastavena na 9600 bitů za sekundu, počet datových bitů 8, sudá parita a 2 stop bity.

Z bezpečnostních důvodů je robot při přepnutí na TEACH mód uveden do low power režimu, v němž lze s robotem pracovat i při otevřené pojistce. Na high power je možné přepnout pouze při zavřené pojistce, posláním příkazu POWER HIGH (LP OFF).

AUTO

AUTO mód je určen pro provoz robotu v továrně. Příkazy pro spuštění a zastavení programu je robotu posílán buďto pomocí operační jednotky, skrze REMOTE3 konektor a nebo RS-232C port. Zvolení, která z uvedených možností bude sloužit jako konzole, je možné odesláním příkazu *CONSOLE* v monitorovacím okně aplikace SPEL for windows.

Pokud je robot v AUTO módu, není s ním možno operovat při otevřené pojistce. Po pokusu o operaci při otevřené pojistce robot vstoupí do quick pause módu, z něhož vystoupí až po otevření pojistky a odeslání spouštěcího signálu. V quick pause módu není možné s robotem operovat.

1.2.5 Vstupy a výstupy kontroléru

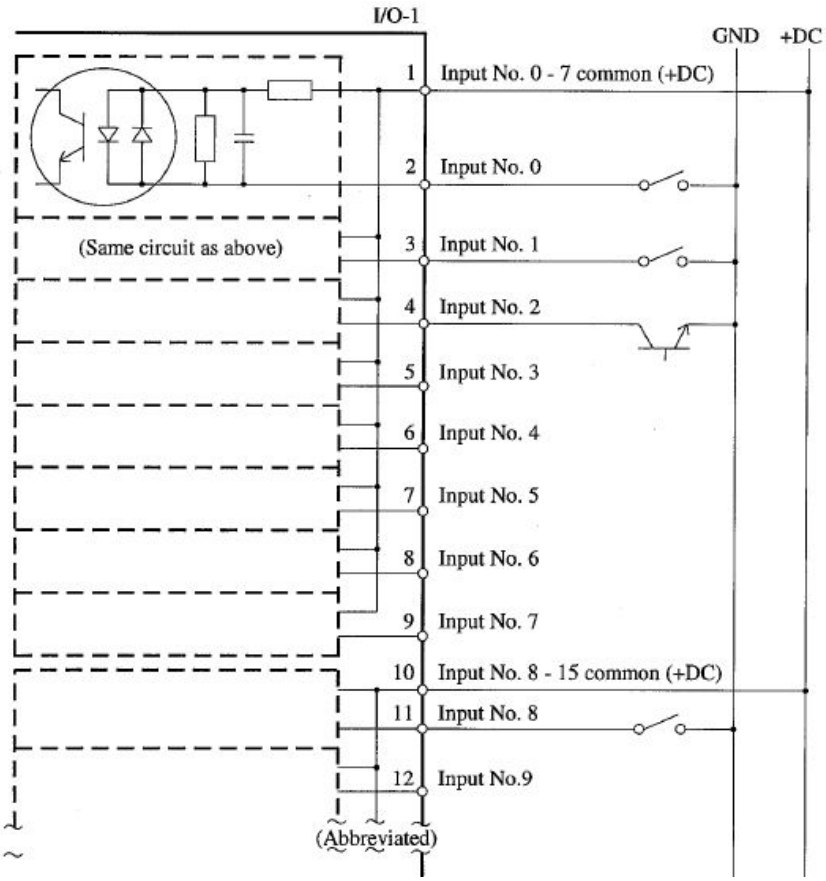
Tato podkapitola čerpá z [2]. Na zadní straně kontroléru SRC 300 se standardně nachází jeden I/O port s binárními vstupy a výstupy. Ten nábízí k použití celkem 16 vstupů a 16 výstupů. Tento počet může být dále navýšen až na hodnotu 126 vstupů a stejného počtu výstupů, přidáním dalších I/O desek.

Hodnoty výstupů jsou po stisknutí nouzového tlačítka přepnuty na vypnuto.

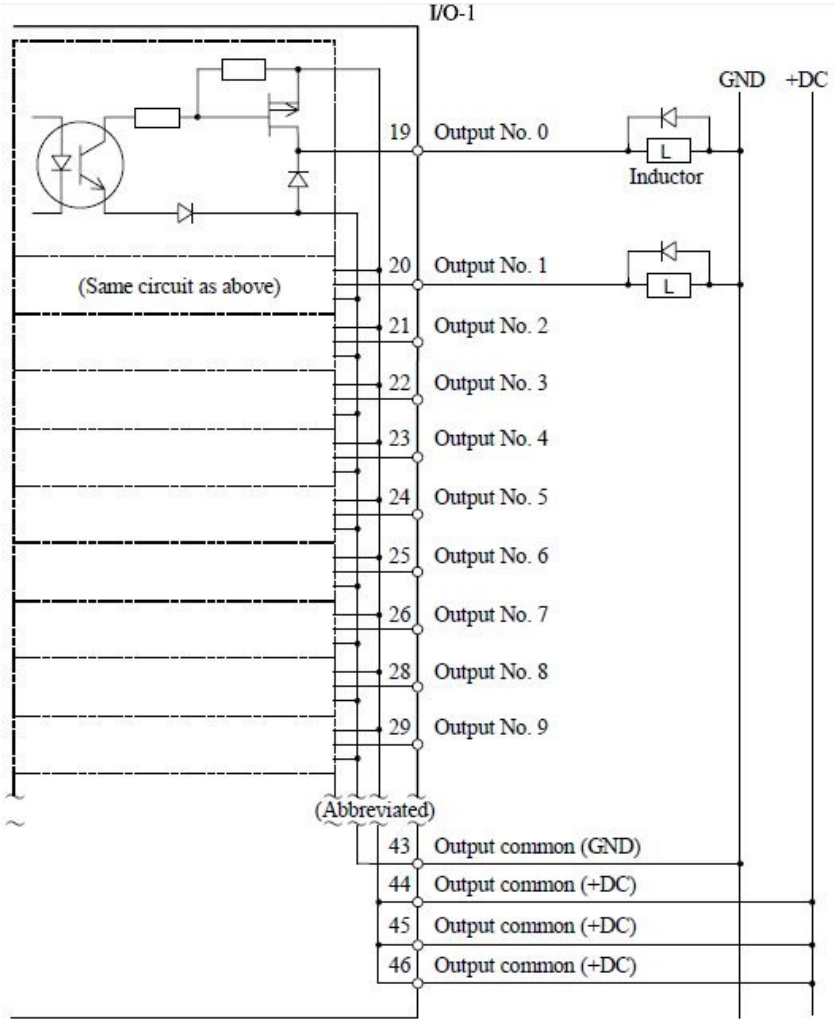
Napětí na vstupech může být v rozmezí 12 až 24 voltů s odchylkou ± 10%, přičemž typický proud pro vstupní napětí 24 V DC je 5 mA. Minimální napětí pro hodnotu zapnuto je 10,8 V a maximální napětí pro hodnotu vypnuto 4 V. Obvod se vstupy s příkladem zapojení je možné vidět na obrázku 1.8

Napětí na výstupech se může pohybovat, obdobně jako na vstupech, v rozmezí 12 až 24 voltů s odchylkou ± 10%. Maximální výstupní proud je 250mA pro každý výstup. Zapojení s výstupy je znázorněno na obrázku 1.9.

K napájení uživatelských zařízení je možné využít piny číslo 47 až 50, na kterémkoliv I/O konektoru, včetně REMOTE1 a REMOTE2. Na pinech 49 a 50 se nachází zem. Výstupní napětí těchto zdrojů je 24 V DC a maximální odebíraný proud 1 A.



Obr. 1.8: Obvod se vstupy [2].



Obr. 1.9: Obvod s výstupy [2].

1.3 Programování robotu

Programovacím jazykem pro kontroléry SRC 300 až 320 je SPEL III, který se svou syntaxí podobá jazyku BASIC [16].

Pro usnadnění programování poskytuje výrobce program SPEL for Windovs, který nabízí přehledné grafické prostředí. To umožňuje uživateli snadno nahrávat programy do kontroléru, ukládat pozice manipulátoru, nebo například sledovat vstupy a výstupy. Některé příkazy je možné provést okamžitě. Ve SPEL for Windows K tomu slouží monitorovací okno.

Následující oddíl popisující jazyk SPEL III čerpá z [16].

1.3.1 SPEL III

Struktura programu

Program obsahuje jednu nebo více funkcí začínajících klíčovým slovem FUNCTION a končících FEND. Veškerý kód se musí nacházet uvnitř funkce a každý řádek programu musí být očíslovaný. Následující ukázka kódu demonstruje formát zápisu funkcí, přičemž ve funkci Main je volána funkce f1 a f2. Funkce f2 se nachází v jiném programu a proto je třeba použít klíčová slova EXTERN FUNCTION:

```
10 FUNCTION Main
20  CALL f1
30  EXTERN FUNCTION f2
40  CALL f2
50 FEND
60
70 FUNCTION f2
80  JUMP P1
90 FEND
```

Názvy proměnných a funkcí

Názvy mohou obsahovat nanejvýš osm alfanumerických znaků a také symbol podtržítka. Controller nerozlišuje velká a malá písmena. Pojmenování dále nesmí začínat číslem nebo písmenem P, které je určeno pro pro názvy uložených bodů. Proměnné typu string musí mít jako poslední znak v názvu symbol dolaru $.

Datové typy

V jazyce SPEL III mohou být deklarovány různé datové typy. Pokud není proměnná deklarována, pak je jí automaticky přidělen datový typ REAL. Všechny datové typy jsou vypsány v následující tabulece:

Datový typ	**Velikost**	**Rozsah**
BYTE	1 bajt	+/-127
INTEGRER	2 bajty	+/-32767
LONG	4 bajty	+/-2147483549
REAL	4 bajty	7 cifer
DOUBLE	8 bajtů	14 cifer
STRING	1 až 80 bajtů	Všechny ASCII znaky

Tab. 1.1: Datové typy jazyka SPEL [16].

Příkazy pro pohyb

Při prvním spuštění kontroléru je třeba robot zkalibrovat pomocí příkazu *MCAL*.

K nastavení rychlosti přesunu slouží příkaz *SPEED* a pro nastavení zrychlení či zpomalení příkaz *ACCEL*.

Příkazy pro pohyb, který nemusí být přímočarý jsou popsány níže:

- GO - přímy přesun na bod.
- JUMP - přeskočení na bod.
- PASS - přiblížení se k bodu.

Pro přímočarý pohyb slouží příkazy:

- MOVE - přesun přímou čarou na bod.
- CMOVE - přesun na bod přímou čarou, bez zpomalení.

Příkaz *PULSE* slouží k pohybu jednoho ze čtyř kloubů manipulátoru.

Pro nastavení přesnosti pozice cíleného bodu je zde příkaz *FINE*.

Následující příkazy slouží k pohybu po křivce:

- ARC - přesune robot skrze jeden bod k druhému, s použitím kruhové interpolace.
- CARC - obdobný výraz jako *ARC*, ovšem bez zpomalení.
- CURVE - definuje pohyb po volné křivce skrze definované body.
- CVMOV - provede pohyb definovaný příkazem *CURVE*.

2 KONSTRUKCE MAGNETICKÉHO ÚCHOPU

Návrh úchopu probíhal se záměrem zvedat drobné kovové předměty s hladkým povrchem (např. mince). Maximální váha břemene zdvihaného manipulátorem je 5 kg. Pokud zvedáme předmět lehčí než 2 kg, není třeba upravovat nastavení manipulátoru příkazem *WEIGHT* [18]. V našem případě tedy není žádné nastavování nutné.

2.1 Výběr elektromagnetu

Elektromagnet byl získán z relé FINDER 40.52.9.012.0000, jenž obsahuje cívku o odporu 220 Ω dimenzovanou pro napájecí napětí 12 V [7]. Při něm je proud protékající cívkou 55 mA. Jádro cívky je tvarované tak, že tvoří s přiloženým kovovým předmětem uzavřený magnetický obvod. Díky své malé energetické náročnosti je možné získaný elektromagnet napájet rovnou z výstupu kontroléru. Je jej ale třeba opatřit ochranným obvodem s diodou, kvůli indukovanému napětí, které by mohlo poškodit tranzistor v obvodu výstupu. Tento obvod můžeme vidět na schématu 1.9, které odpovídá výslednému zapojení magnetu.

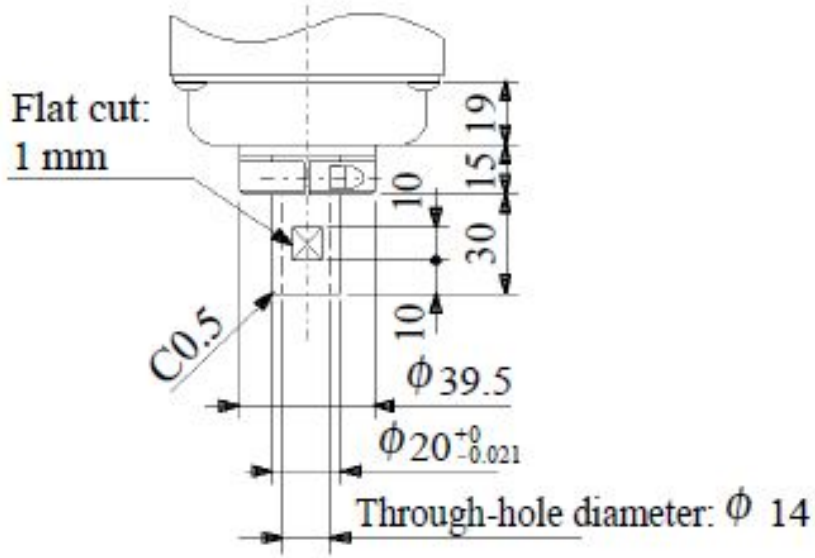
2.2 Návrh krytu

Kryt byl navržen tak, aby jej bylo možné připevnit na držadlo s hladkým řezem s parametry viz. 2.1. Pro vytvoření 3D modelu byl využit program SkechUp, který se vyznačuje svým snadným a intuitivním ovládáním. Model byl tvořen se zvětšením 1:1000, z důvodu zachování drobných detailů. Při vytváření otvorů a drobných struktur bylo počítáno s odchylkou od skutečných rozměrů vytištěné součástky, a proto byla k původním hodnotám přidána rezerva 1 mm.

Výsledný model s vyznačenými rozměry je vyobrazen na obrázku 2.2. Přivedení kabelů až k úchopu je možné skrze dutinu v části manipulátoru, která obstarává vertikální a rotační pohyb.

Na plášti válcovitého modelu je vidět otvor pro šroub sloužící k upevnění úchopu přitlačením na hladký řez držáku. Vnitřní pohled na otvor a výsek pro připevnění matice je dále zobrazen na obr. 2.3.

Ve spodní podstavě je umístěn otvor pro vsunutí elektromagnetu viz. 2.4. K jeho upevnění dále slouží tenký plíšek procházející mezi cívkou a jádrem. Prostor uvnitř krytu jsou dostatečné pro umístění gumy chránící jeho stěnu a zároveň zabraňující sklouznutí.



Obr. 2.1: Rozměry držadla [9].

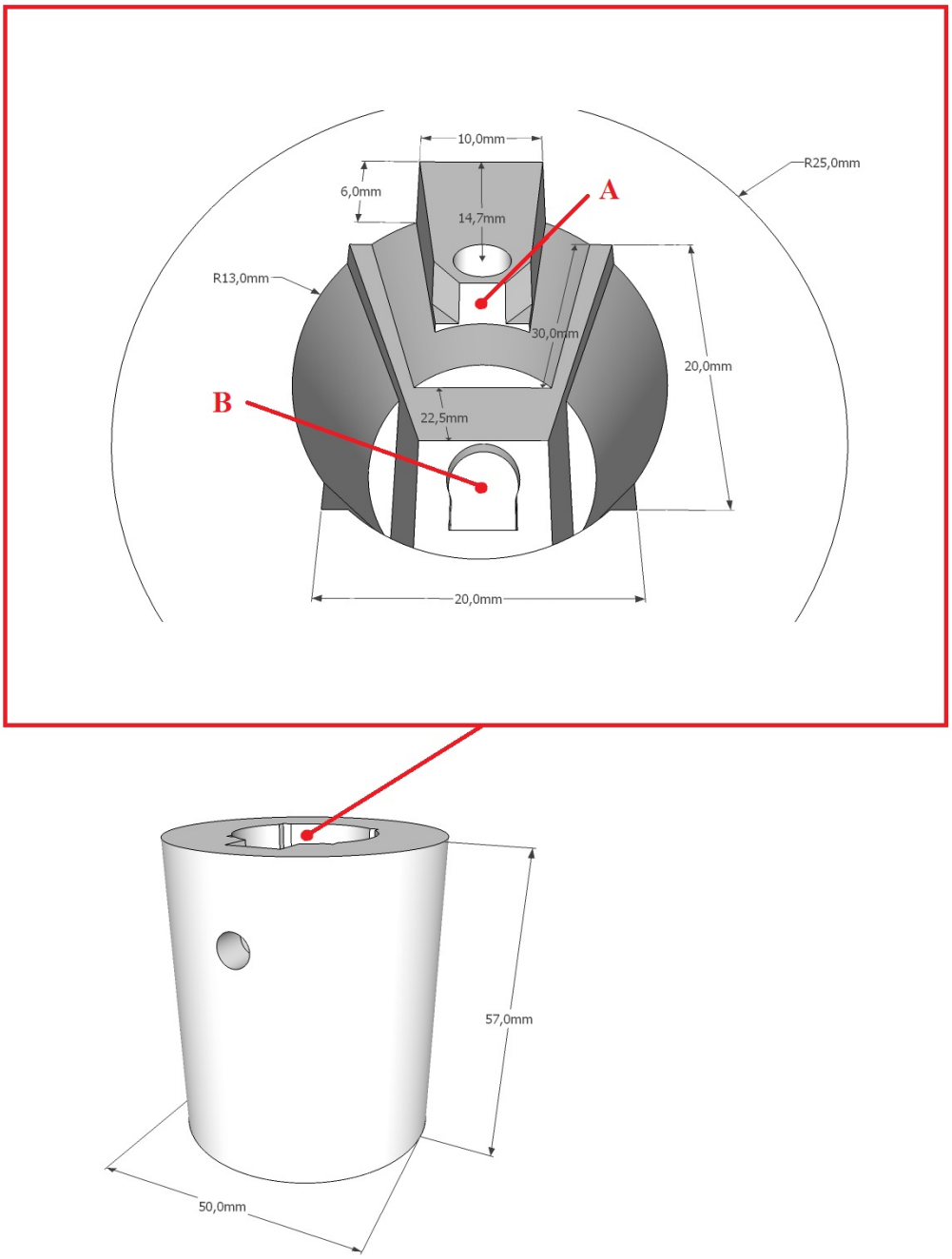
Tisk úchopu

Tisk proběhl na 3D tiskárně typu REBELIX navrženou Martinem Nerudou. Tiskárna vychází z modelů Rebel II a Průša i3 a vyznačuje se svou pevností a jednoduchostí [1]. Pro tisk byl zvolen běžný materiál ABS, jehož přednostmi jsou zdravotní nezávadnost, odolnost proti vysokým i nízkým teplotám a také proti machanickému poškození. Pro tyto vlastností je hojně využíván při výrobě hraček, domácích potřeb či hudebních nástrojů [17].

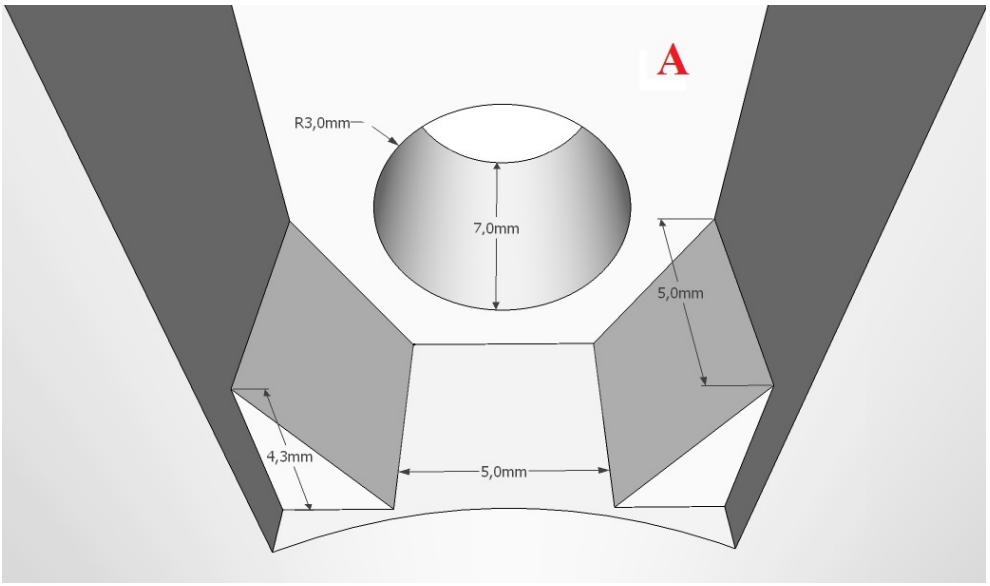
K vystužení vnitřních prostor krytu byla zvolena hexagonální výplň, zaručující potřebnou pevnost. Tyto tvary jsou patrné na obrázku 2.5, na kterém je zachycen tisk úchopu.

Sestavení úchopu

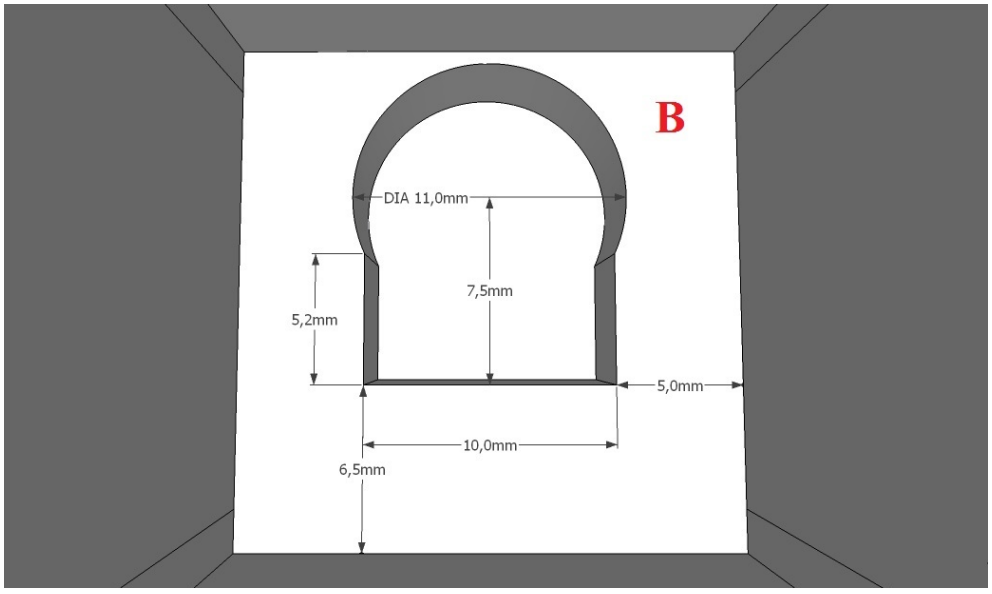
K připevnění elektromagnetu a matice upevňovacího šroubu byla využita tavná pistole. Na vnitřní stěnu, protilehlou k šroubu, byla dále přitisknuta ochranná guma. Na cívku byly připájeny přívodní vodiče a dioda chránící výstup kontroléru. Výsledný elektromagnet je ukázán na obrázku 2.6.



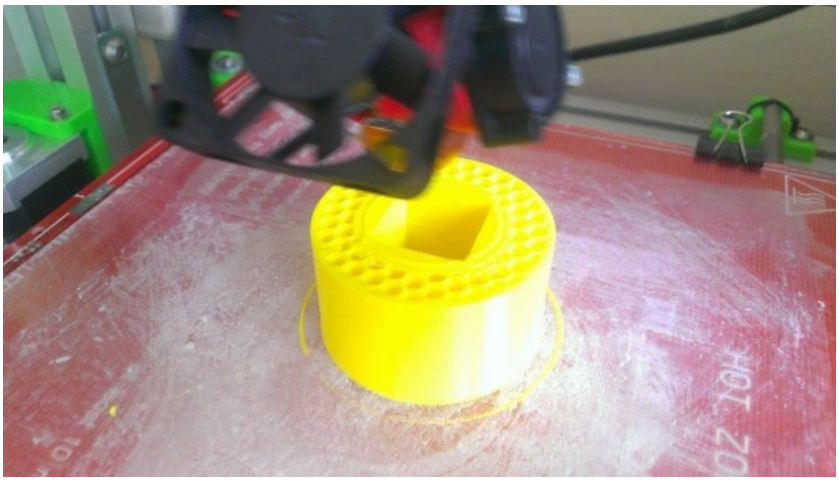
Obr. 2.2: Rozměry modelu úchopu.



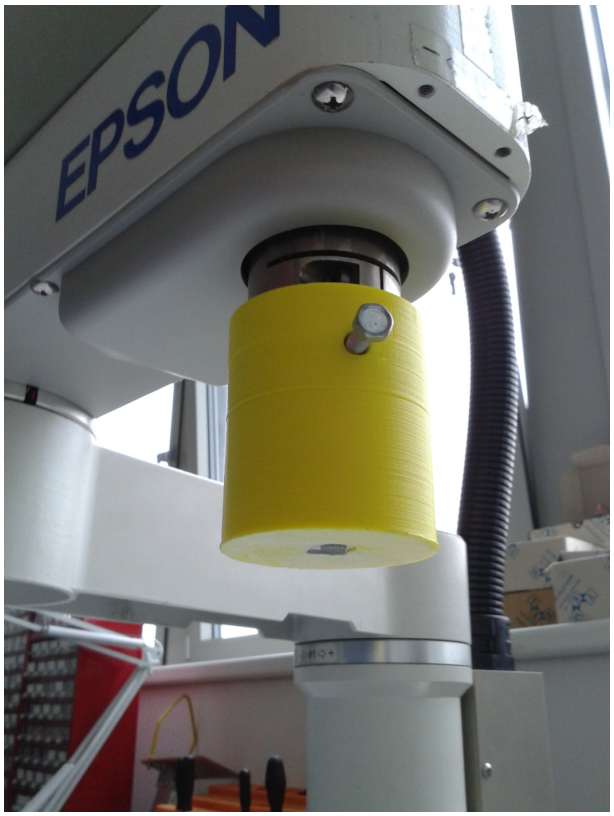
Obr. 2.3: Rozměry modelu úchopu.



Obr. 2.4: Rozměry modelu úchopu.



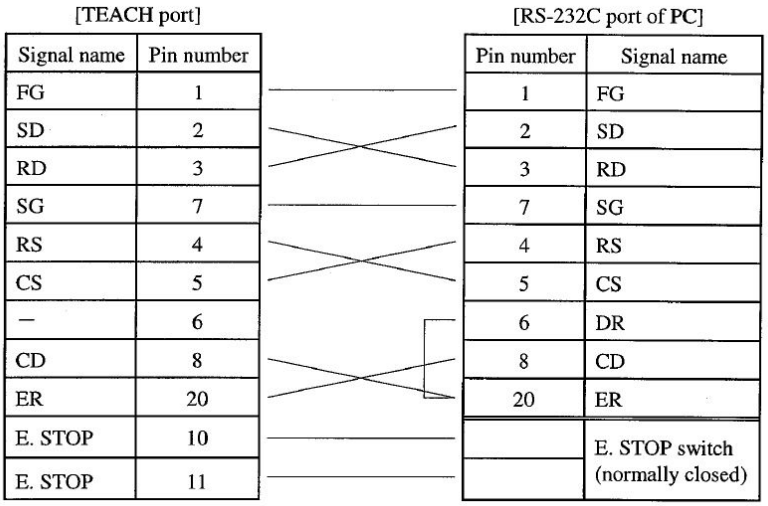
Obr. 2.5: Tisk krytu pro magnetický úchop.



Obr. 2.6: Výsledný vzhled úchopu.

3 ZPROVOZNĚNÍ ROBOTU A PŘÍPRAVA PRACOVIŠTĚ

Jak je uvedeno v [20], robot využívá jednofázové napájení, kdy hodnota střídavého napětí musí dosahovat 200 až 220 V s odchylkou ± 10%. Napájecí kabel není opatřen zástrčkou, proto na něj byla s použitím svorkovnice (čokolády) připojena flexo šňůra s třemi vodiči o průřezu 1,5 mm^2 Ke konektoru REMOTE1 byl připojen spínač pro ovládání pojistky a taktéž byl vybrán pro spínání nouzové brzdy na pinech 12 a 5 [3]. REMOTE1 byl upřednostněn, jelikož s ním není třeba v průběhu práce s robotem manipulovat a typ konektoru je snadno dostupný na rozdíl od REMOTE2. Ten je typu SCSI MDR a vzhledem k jeho vyšší ceně, byla zvolena možnost připájet přímo na desku přepínač módu TEACH/AUTO a propojit okruh nouzové brzdy viz. 1.7. Vycházejíce z [21], byl k programování vytvořen teach kabel propojující TEACH port a RS-232 konektor PC. Na TEACH portu byly propojeny piny 10 a 11 vypínající nouzovou brzdu a dále bylo provedeno zapojení dle obr. 1.7. Dále byl vytvořen konektor rušící nouzovou brzdu, který bude připojen k TEACH portu během ovládání robotu přes RS-232C #20.



Obr. 3.1: Propojení teach kabelu [21].

4 DRIVER PRO ONLINE ŘÍZENÍ ROBOTU

Příkazy v jazyce SPEL, které byly použity v ovladači a jsou popisovány v této kapitole, byly nastudovány z [22]. Pro ovladání robotu přes sériový port bylo nejprve potřeba v TEACH módu zvolit jako konzoli pro AUTO mód RS-232C port #20. Toho bylo docíleno odesláním příkazu *CONSOLE #20* a následnou konfigurací pomocí *CONFIG 5, 3, 5, 7*. Tímto byly ihned po restartování kontroléru nastaveny parametry komunikace podle tabulky 4.1.

Parametr	**Hodnota**
Modulová rychlost	38400 Bd
Parita	žádná
Protokol	TTY
Data bity	8
Stop bity	1
Handshaking	XON/XOFF
Ukončovací znak	CR
Time-out	5 s

Tab. 4.1: Parametry komunikace.

Zvolený TTY protokol nepoužívá žádnou větší režii. Jedná se pouze o odesílání dat zakončených ukončovacím znakem CR (&H0D) [23]. Kontrolér signalizuje svou připravenost přijmout nový příkaz odesláním speciálního znaku (>). Robotem je možné pohybovat a ovládat jej jednoduchým posíláním příkazu v jazyce SPEL přes sériový port. Má to ovšem tu nevýhodu, že další požadavek je možné vyslat až po vykonání předešlého příkazu. Tento způsob ovládání je tedy nepoužitelný, pokud máme během pohybu robotu sledovat informace o jeho poloze nebo kontrolovat stav připojení. Pro potřeby ovladače byla zvolena možnost kombinující odesílání příkazů v jazyce SPEL a měnění hodnot proměnných ovládajících chod programu nahraného přímo do kontroléru. Komunikace tedy probíhá mezi programem napsaným v jazyce SPEL a třídou RobotDriver v jazyce C# odesílající povely přez RS-232.

4.1 Program v kontroléru

Program který je v kontroléru spouštěn po připojení používá a je řízen pomocí následujících proměnných:

- BYTE s - spouští pohyb manipulátoru.
- BYTE r - slouží ke kontrole připojení.

- BYTE d - signalizuje dokončení pohybu.
- BYTE e - signalizuje dokončení série pohybů (PTP).
- BYTE k - oznamuje, že je třída srozuměna se změnou stavu *d* a *e*.
- BYTE b - umožňuje ukončení provádění série pohybů (PTP).
- INTEGRER t - volí číslo vstupu kontroléru pro ovládání pohybu.
- BYTE z - spouští opakování série pohybů (PTP).
- BYTE v - řídí pozastavení provádění série pohybů (PTP).
- BYTE w - signalizuje zaneprázdněnost kontroléru.
- BYTE i - slouží k volbě typu pohybu.
- DOUBLE value1 až 4 - pomocné proměnné pro uložení aktuální pozice.
- INTEGRER sum - pomocná proměnná pro volbu indexu pozice.
- INTEGRER var - pomocná proměnná pro volbu indexů pozic.
- LONG C - slouží k uložení hodnot pulzů jednotlivých os.
- INTEGRER n - určuje počet pohybů v PTP.
- INTEGRER q - nastavuje trajektorii zdvihání a snižování u pohybu JUMP.
- DOUBLE l - volí výšku zdvihu při pohybu JUMP.
- BYTE m - určuje jestli se jedná o PTP pohyb.

Program je rozdělen na čtyři samostatné funkce jejichž účel je popsán níže:

4.1.1 Funkce begin 4.1

Jedná se o funkci spouštěnou hned po připojení, která resetuje kontrolér a uvede jeho proměnné do výchozího stavu. Dále jsou na konci funkce pomocí příkazu *XQT* souběžně spuštěny funkce **ask** a **movement**. Toto řešení bylo zvoleno, jelikož při spuštění funkce příkazem z konzole je možné posílat další povely až po jejím dokončení.

4.1.2 Funkce ask 4.2

Funkce **ask** slouží ke kontrole připojení, kdy je hodnota proměnné *r* kontrolérem opakovaně měněna na nulu a třídou v PC naopak na hodnotu jedna. Pokud bude po jedné sekundě hodnota stále nula dojde k spuštění ukončovací funkce **quitcom**.

4.1.3 Funkce quitcom 4.3

Funkce vypíná motory manipulátoru, ukončuje běžící funkce a resetuje kontrolér. Příkaz *SELRB 1* umožňuje funkci ovládat manipulátor.

```
FUNCTION begin
        RESET
        s = 0
        r = 1
        d = 0
        e = 0
        k = 1
        b = 0
        t = 0
        z = 0
        v = 0
        XQT \!3, movement
        XQT \!4, ask
FEND
```

Ukázka kódu 4.1: Funkce begin

```
FUNCTION ask
        check:
        r = 0
        WAIT 1
        IF r = 0 THEN
                XQT !7, quitcom
        ENDIF
        GOTO check
FEND
```

Ukázka kódu 4.2: Funkce begin

```
FUNCTION quitcom
        SELRB 1
        MOTOR OFF
        QUIT !3
        QUIT !4
        RESET
FEND
```

Ukázka kódu 4.3: Funkce quitcom

4.1.4 Funkce movement 4.4 4.5 4.6

Tato funkce obstarává spouštění jednotlivých pohybů. Na jejím začátku je nejprve spuštěna kalibrace manipulátoru příkazem *MCAL*. Ten je potřeba odeslat vždy po spuštění motorů. Dále program vejde do smyčky, kde čeká na povel pro spuštění pohybu (řádek 8). Je-li smyčka zrušena, program prochází přes nastavení vstupu pro řízení pohybů JUMP na řádcích 14 a 18 až k části kódu, kde je řízeno přemístění ramene na souřadnice aktuální polohy (řádek 22). Toto je potřebné před spuštěním pohybu ARC, který by byl jinak ukončen při spuštění jako první pohyb následující po kalibraci.

Program pokračuje k FOR cyklu na řádku 33, který provede žádaný pohyb o n krocích. Tento cyklus jak při jednotlivých pohybech, tak i při vykonávání série pohybů PTP. Typ pohybu je zvolen na základě proměnné i vyhodnocené pomocí SELECT. Souřadnice pro přesun jsou ukládány jako body do speciální proměnné P označené číslicí. Pouze v případě pohybu PULSE jsou jednotlivé hodnoty pulzů ukládány do vícerozměrné proměnné C. Po dokončení pohybu se v programu změní hodnota proměnné d, která je sledováná třídou RobotDriver. Ta po srozumění se stavem změní hodnotu proměnné k a ukončí tak cyklus na řádku 51. Ten je zde, aby mohly být odeslány souřadnice s aktuální polohou před startem dalšího pohybu. Na řádku 54 se dále nachází podmínka řízená třídou, která při splnění způsobení vyskočení z FOR cyklu. Poté je zde také cyklus umožňující pozastavit vykonávání série pohybů (řádek 58).

V závěru funkce je na řádku 61 v případě PTP pohybu oznámeno dokončení série a následuje řádek 69 s možnost opakování celého průběhu PTP. Poté jsou inicializovány jednotlivé proměnné a program se vrací na začátek funkce.

4.2 Třída RobotDriver

Třída byla vytvořena tak, aby byla kompatibilní s předloženým interfacem IRobotDriver. Celá komunikace s robotem je postavena na odelsílání příkazů v jazyce SPEL po sériové lince s využitím standardní třídy SerialPort. Ta umožňuje snadnou konfiguraci komunikace a nabízí událost DataReceived volanou po přijetí jakýchkoliv dat přes rozhraní RS-232. Manuál třídy, vytvořený programem Doxygen, se nachází v příloze bakalářské práce.

RobotDriver obsahuje tyto globální proměnné:

- double actX až U - ukládá hodnotu zjištěné aktuální souřadnice.
- int pulseX až U - slouží k uložení aktuální hodnoty pulzů.
- bool in0 až 4 - ukládá stav vstupů kontroléru.
- bool busy - signalizuje probíhající odesílání dat do kontroléru.

```
FUNCTION movement
        SELRB 1
        MCAL
        w = 0

        loop:

        WHILE s = 0
        WEND

        w = 1
        b = 0

        IF I  = 5 THEN
                SENSE SW(t) = 1
        ENDIF

        IF I  = 8 THEN
                TILL SW(t) = 1
        ENDIF

        IF I  = 6 THEN
                value1 = CX(P*)
                value2 = CY(P*)
                value3 = CZ(P*)
                value4 = CU(P*)
                sum = 2 * n
                Psum = value1, value2, value3, value4
                GO Psum
        ENDIF
```

Ukázka kódu 4.4: Funkce movement-kalibrace a nastavování

```
31 repeat:
32
33 FOR var = 0 TO n - 1
34         SELECT i
35
36         CASE 0; GO Pvar
37         CASE 1; JUMP Pvar Cq LIMZl
38         CASE 2; MOVE Pvar TILL SW(t) =1
39         CASE 3; PASS P0-Pq
40         CASE 4; GO Pvar TILL SW(t) =1
41         CASE 5; JUMP Pvar Cq LIMZl SENSE
42         CASE 6; sum = n + var; ARC Pvar, Psum
43         CASE 7; PULSE C(0,var),C(1,var),C(2,var),C(3,var)
44         CASE 8; JUMP Pvar Cq LIMZl TILL
45
46         SEND
47
48         k = 0
49         d = 1
50
51         WHILE k = 0
52         WEND
53
54         IF b=1 THEN
55             GOTO break
56         ENDIF
57
58         WHILE v = 1
59         WEND
60 NEXT var
```

Ukázka kódu 4.5: Funkce movement-spouštění pohybů

```
61          IF m = 1 THEN
62                  k = 0
63                  e = 1
64
65                  WHILE k = 0
66                  WEND
67          ENDIF
68
69          IF z = 1 AND m=1 THEN
70                  GOTO repeat
71          ENDIF
72
73          break:
74
75          s = 0
76          w = 0
77          m = 0
78          t = 0
79          z = 0
80
81          GOTO loop
82  FEND
```

Ukázka kódu 4.6: Funkce movement-hlášení o konci pohybu

- bool connected - určuje stav připojení.
- string message - ukládá celistvou zprávu přijatou po RS-232.
- string received - nese útržky přijaté zprávy.
- bool answerRequest - signalizuje požadavek k odeslání odpovědi pro kontrolu připojení.
- bool commFail - určuje zda připojení selhalo.
- bool controllerBusy - signalizuje zaneprázdněnost kontroléru.
- bool sfreeMode - říká že je umožněn manuální pohyb ramen manipulátoru.
- bool pom a pom1 - pomocné proměnné pro získání aktuálních pozic po kalibraci a řízení PTP události.
- bool controllerConnected - proměnná sdělující, že je robot připojen a zkalibrován.
- bool stopPtp - proměnná pro řízení zastavení PTP pohybu.
- bool pausePtp - proměnná pro řízení pozastavení PTP pohybu.
- bool unpausePtp - proměnná pro zrušení pozastavení PTP pohybu.
- bool checkActCoords - volí zda se třída ptá na aktuální souřadnice.
- bool checkActPulse - volí zda se třída ptá na aktuální hodnoty pulsů.
- bool checkInputs - volí zda se třída ptá na stavy vstupů.
- bool loopPTP - proměnná pro volbu cyklického provádění PTP pohybu.
- bool pomLoopPTP - pomocná proměnná pro volbu cyklického provádění PTP pohybu.
- bool changeOutput - ukazuje na požadavek změny výstupu.
- bool outputState - nese stav výstupu.
- uint outputNum - určuje číslo měněného výstupu.
- bool error - signalizuje chybu.
- SerialPort ComPort - instance třídy SerialPort obsluhující sériovou komunikace.
- System.Timers.Timer conCheck - časovač pro kontrolu připojení.
- System.Timers.Timer respond - časovač pro odpověď kontroléru kontrolujícího připojení.

4.2.1 Připojení ke kontroléru

Pro obsluhu připojení třída nabízí dvě metody: Connect() a Disconnect(). Parametry komunikace jsou nastaveny v konstruktoru třídy.

Metoda Connect() 4.7 4.8

V metodě **Connect** dojde nejprve na řádku 1 k inicializaci proměnných s aktuálními souřadnicemi. Řádkem 7 začíná kontrola, jestli již není třída připojena. Pokud ano,

dojde k přerušení metody s návratovou hotnotou false. Od řádku 12 začíná kontrola vstupního parametru.

```
1  actX = 0;
2  actY = 0;
3  actZ = 0;
4  actU = 0;
5
6  //return if alredy connected
7  if (connected == true)
8          return false;
9
10 controllerBusy = true;
11
12 //checking of parameters
13 string com = address.Substring(0, 3);
14 string num = address.Substring(3, address.Length - 3);
15 uint number;
16
17 if (com != "COM" && com != "com" ||
                                    uint.TryParse(num,out
                                    number) == false)
18         throw new Exception("Bad format of port
                                            adress!");
```

Ukázka kódu 4.7: Metoda connect-kontrola vstupních parametrů

Na řádku 19 dojde k nastavení adresy portu. Následně je otevřena komunikace vyčištěn vstupní buffer a změněna hodnota proměnné *connected*, kvůli umožnění následného odesílání příkazů od řádku 27. To je realizováno metodou **send** která bude popsána v následujících kapitolách. Poslané příkazy resetují kontrolér, spustí motory a změní hodnotu proměnné, která ukazuje zda je kontrolér zaneprázdněn. Na řádku 34 je zahájeno potvrzování připojení. Pod řádkem 37 je spuštění funkce **begin**, následované spuštěním kontroly připojení a ukončením metody s návratovou hodnotou true.

Metoda Disconnect() 4.9 4.10

Řádek 2 ukončuje metodu pokud není třída připojena. Jestliže připojena je, následuje inicializace proměnné *sfreeMode* a na řádku 8 dojde k vyřazení funkce metod pro posílání příkazů. Řádek 11 ukončí kontrolu připojení a poté je na řádku 14 zjišťováno, jestli k odpojení nedochází po chybě v kontroléru. Je-li tomu tak, třída

```
19 //serial port address setting
20 ComPort.PortName = address;
21
22 //serial port opening
23 ComPort.Open();
24 ComPort.DiscardInBuffer();
25 connected = true;
26
27 //reset controller
28 send("RESET");
29
30 //motor engage and calibration
31 send("MOTOR ON");
32 send("w = 1");
33
34 //start responding for connection check
35 respond.Enabled = true;
36
37 //executing of controller program
38 send("XQT !2, begin");
39 //start to check connection
40 conCheck.Enabled = true;
41
42 return ComPort.IsOpen;
```

Ukázka kódu 4.8: Metoda Connect-odesílání příkazů kontroléru

odešle příkaz pro vypnutí motorů. Dělá tak z důvodu, že po chybě v kontroléru jsou ukončeny všechny funkce v něm běžící. Tedy i funkce kontrolující připojení.

```
1  //skip if not connected
2  if (connected == false)
3          return;
4
5  sfreeMode = false;
6
7  //discard send method
8  connected = false;
9
10 //Stopping of connection checking
11 conCheck.Enabled = false;
12
13 //motor stopping
14 while(error == true)
15 {
16         if (busy == false && answerRequest == false)
17         {
18                 busy = true;
19                 write("motor off" + "\r");
20                 break;
21         }
22 }
23 error = false;
```

Ukázka kódu 4.9: Metoda Disconnect-část 1

Na řádku 25 dochází k vypnutí odpovídání kontroléru. Poté dojde k uzavření komunikace (řádek 28) a následné inicializaci proměnných *busy*, *answerRequest* a *controllerBusy*. Dále se na řádku 34 rozhoduje o vyvolání události **onConnectionLost**. K tomu dojde tehdy, byla-li metoda **Disconnect** zavolána vlivem ztráty připojení. Na konci metody (řádek 42) je posléze ještě vyvolán event **onDisconnect**.

4.2.2 Odesílání příkazů

O posílání povelů se starají dvě privátní metody **send** a **urgent**. Obě používají společnou metodu **write**, odesílající zprávu po jednotlivých znacích. Kód metody **send** můžeme vidět v ukázce 4.11. Celá metoda běží v nekonečné smyčce, ve které se čeká na odeslání dřívější zprávy. Cyklus je na řádku 4 ukončen, pokud dojde k přerušení spojení. Metoda **urgent** má vyšší prioritu než metoda **send**, protože je

```
24 //stop to respond (connection checking)
25 respond.Enabled = false;
26
27 //closing port
28 ComPort.Close();
29 busy = false;
30
31 answerRequest = false;
32 controllerBusy = false;
33
34 if(commFail==true)
35 {
36         if(onConnectionLost != null)
37                 onConnectionLost(this, EventArgs. Empty);
38 }
39 commFail = false;
40
41 //disconnect envent
42 if (onDisconnect != null)
43         onDisconnect(this, EventArgs.Empty);
```

Ukázka kódu 4.10: Metoda Disconnect-část 2

využívána k odesílání zpráv souvisejících s kontrolou připojení. Jediný rozdíl v kódu je tedy na řádku 8, kde v metodě **urgent** chybí podmínka s *answerRequest*.

```
1  while (true)
2  {
3          //skip sending while not connected
4          if (connected == false)
5                  break;
6
7          //checking if ready
8          if (busy == false && answerRequest == false)
9          {
10                 busy = true;
11                 write(order + "\r");
12                 break;
13         }
14 }
```

Ukázka kódu 4.11: Metoda send

Metoda **write** 4.12 rozdebírá ve for cyklu vstupní string *orderToWrite* na jednotlivé znaky, které postupně odesílá na řádku 8.

```
1  string CommandSent;
2  int length, j = 0;
3
4  length = orderToWrite.Length;
5  for (int i = 0; i < length; i++)
6  {
7          CommandSent = orderToWrite.Substring(j, 1);
8          ComPort.Write(CommandSent);
9          j++;
10 }
```

Ukázka kódu 4.12: Metoda write

4.2.3 Přijímání zpráv

Zpracovávání zpráv kontroléru je postaveno na události **DataReceived** třídy SerialPort. V metodě **ComPort_dataReceived** 4.13 4.14, která událost obsluhuje je na začátku vyvolána událost **onMessage** a poté jsou na řádku 4 přečtena přijatá data. Hned poté probíhá kontrola, jestli se v nich nenachází znak >, ohlašující dokončení

předchozího povelu. Pokud změní se hodnota proměnné busy na false a vyvolá se příslušná událost. Na řádku 12 je následně restartován časovač hlídající připojení.

```
1  if (onMessage != null)
2          onMessage(this, EventArgs.Empty);
3
4  message = ComPort.ReadExisting();
5  if (message.Contains(">"))
6  {
7      busy = false;
8      if (onCommandConfirmed != null)
9          onCommandConfirmed(this, EventArgs.Empty);
10 }
11
12 conCheck.Enabled = false;
13 conCheck.Enabled = true;
```

Ukázka kódu 4.13: Obsluha události DataReceived - část 1

Přez sériový port přicházejí útržky zpráv, které jsou na řádku 15 spojovány dohromady. Níže pak probíhá while cyklus, který z přijatého řetězce vybírá jednotlivé zprávy ukončené znakem \r, které pak následně od řádku 24 vyhodnocuje na základě obsaženého stringu. Po zkontrolování všech podmínek je přijatý řetězec oříznut o první zprávu a pokud stále odsahuje ukončovací znak, celý cyklus se opakuje.

4.2.4 Kontrola připojení

Na kontrole připojení se podílejí dva časovače vyvolávající cyklicky se opakující události. Ty jsou obsluhovány motodami **conChecking** a **responding**. Metoda **conChecking** je volána za jednu sekundu. Pokud k tomu dojde, proměnná *commFail* se nastaví na true a dojde k odpojení metodou **Disconnect**. Resetování časovače je prováděno v metodě **ComPort_DataReceived** na řádku 12, čímž je odpojení zabráněno.

V metodě **responding** je na začátku změněna hodnota globální proměnné *answerRequest* na true a dále je vstoupeno do nekonečné smyčky, kde se čeká na zpracování předchozího příkazu. Obsah smyčky je ukázán v úryvku 4.15. V něm je nejprve zkontrolována podmínka připojení a následně je pomocí metody **urgent** změněna proměnná *r* na 1, čímž se zabrání odpojení. Poté jsou odesílány příkazy požadující poslání informací z kontroléru a měnící jeho nastavení. Na konci je *answerRequest* změneno na false a je vyskočeno z cyklu.

```
14 //processing of received data
15 received = received + message;
16
17 while (received.Contains("\r"))
18 {
19         int sign = received.IndexOf("\r");
20         int length = received.Length;
21         message = received.Substring(0, sign);
22
23         //message check
24         if (message.Contains("cX"))
25         {
26                 readPosition(message,true);
27         }
28
29                         .
30                         .
31                         .
32
33
34         //cut received string
35         received = received.Substring(sign + 1, length -
                                                sign - 1);
36 }
```

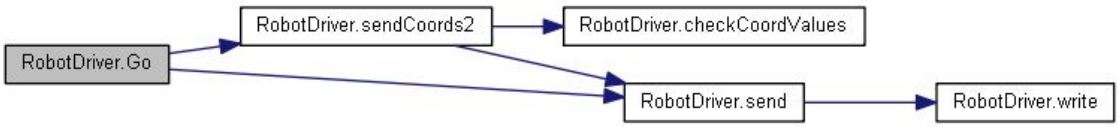
Ukázka kódu 4.14: Obsluha události DataReceived - část 2

```
14 if (connected == false)
15     break;
16
17 if (busy == false)
18 {
19     if(sfreeMode == true)
20     {
21      urgent("r = 1");
22      urgent("print \"cX\",CX(P*),CY(P*),CZ(P*),CU(P*)");
23     }
24     else
25     {
26         urgent("r = 1;print \"w\",w,\"d\", d,\"e\",e;");
27
28         if(checkInputs == true)
29            urgent("print \"iN\",SW(0),SW(1),SW(2),
                                                SW(3),SW(4)");
30     }
31                          .
32                          .
33                          .
34     answerRequest = false;
35         break;
36 }
```

Ukázka kódu 4.15: Metoda responding

4.2.5 Ovládání pohybů manipulátoru

Třída obsahuje metody pro vykonávání různých druhů pohybů s možností zadat požadovanou polohu několika způsoby. Do kontroléru jsou nejprve odeslány souřadnice bodů nebo hodnoty pulzů. Většina metod k tomu využívá jednu ze společných metod **sendCoords1** až **3**, ve kterých zároveň probíhá kontrola připojení a zaneprázdněnosti kontroléru. Metody **Pass** a **Pulse** odesílají data s polohou mírně odlišně, ale na stejném principu. Na diagramu 4.1 vygenerovaného Doxygenem[1] pro metodu **Go** s využitím Graphviz [2], vidíme volání jednotlivých metod. Veřejná metoda **checkCoordValues** počítá, zda jsou zaslané souřadnice v dosahu ramene manipulátoru a v případě že ne, vypíše vyjimku. Metodu je možno pomocí vstupního parametru přepnout do módu kdy návratová hodnota true určuje splnění rozsahu. Kontrola dosahu byla vytvořena, protože při zadání špatných souřadnic byla kontrolérem odesílána chyba ukončujících všechny probíhající funkce.



Obr. 4.1: Diagram metody Go s voláním dalších metod.

Kód metody **Go** je vypsán v 4.16 a metodu **sendCoords1** vidíme v ukázce 4.17. Uvnitř metody **Go** jsou po odeslání souřadnic poslány hodnoty proměnných určující typ pohybu a zahajující jeho vykonávání. V metodě **sendCoords1** jsou od řádku 13 odesílány souřadnice s nastavením *InvariantCulture*, které zajišťuje vypsání plovoucí čárky ve tvaru tečky.

```
1 public void Go(double[] coordinates)
2 {
3         sendCoords1(coordinates);
4
5         send("i=0;s=1");
6 }
```

Ukázka kódu 4.16: Metoda Go

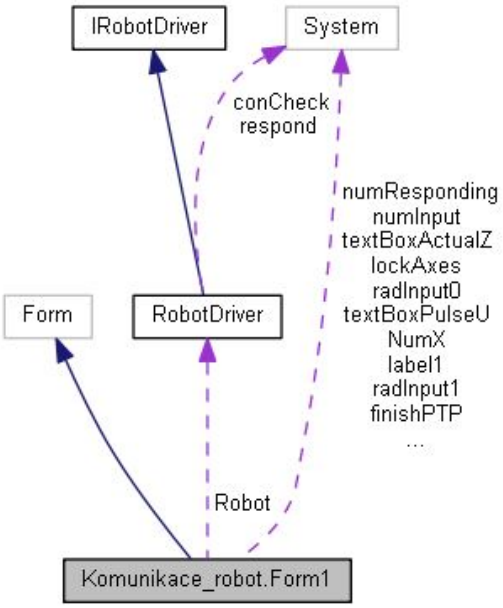
[1]Nástroj pro tvorbu dokumentace na základě zdrojového kódu.
[2]Open source software pro grafickou vizualizaci.

```
1  private void sendCoords1(double[] coordinates)
2  {
3  if (controllerBusy == true)
4          throw new Exception("Controller is busy");
5
6  if (connected == false)
7          throw new Exception("Controller not connected!");
8
9  controllerBusy = true;
10
11 checkCoordValues(coordinates[0], coordinates[1],
                                    coordinates[2],
                                    coordinates[3]);
12
13 send("P0=" + coordinates[0].ToString(
14 System.Globalization.CultureInfo.InvariantCulture)
15 + "," + coordinates[1].ToString(
16 System.Globalization.CultureInfo.InvariantCulture)
17 + "," + coordinates[2].ToString(
18 System.Globalization.CultureInfo.InvariantCulture)
19 + "," + coordinates[3].ToString(
20 System.Globalization.CultureInfo.InvariantCulture)
21 + ";n=1");
22 }
```

Ukázka kódu 4.17: Metoda sendCoords1

4.3 Grafické rozhrání

Byla vytvořena aplikace s grafickým uživatelským prostředím, která používá třídu RobotDriver a demonstruje tak funkci jednotlivých metod, událostí a parametrů. Diagram tříd využívaných aplikací je vyobrazen na obrázku 4.2.



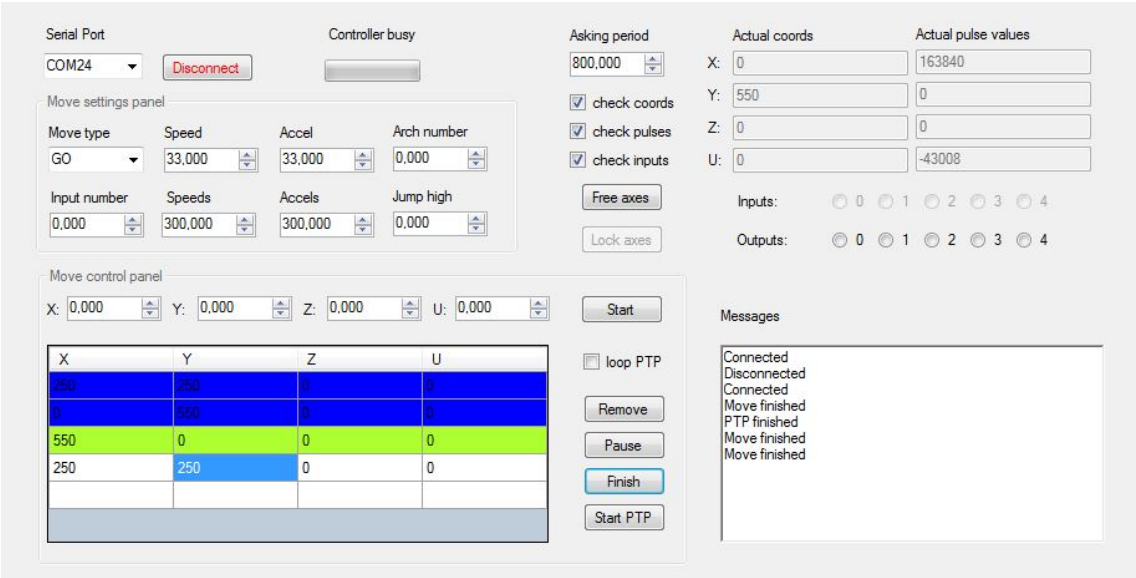
Obr. 4.2: Diagram aplikace.

Na obrázku 4.3 můžeme vidět ovládací panel, v jehož levé části se nahoře nachází rozbalovací seznam pro volbu sériového portu, tlačítko připojení a ukazatel stavu kontroléru. Níže pak máme panel s nastavením parametrů pohybu. Úplně dole se poté nachází panel pro zadávání souřadnic a ovládání pohybu. V něm je možné buďto zadat souřadnice či pulsy pro jediný pohyb, nebo vypsat sérii hodnot do tabulky, kterou využívají pohyby PTP. V ní se pak při vykonávání pohybů označuje ten, jenž je aktuálně prováděný. Na pravé straně panelu jsou nahoře vypisovány aktuální souřadnice a hodnoty pulsů, přičemž je zde také okénko s popiskem *Asking period*, měnící periodu získávání hodnot z kontroléru. Pokud je nastavena příliš nízká, může docházet k chybám komunikace. Dále jsou nahoře zobrazeny stavy jednotlivých vstupů a nachází se tam ovládání výstupů. Ve spodní tabulce se vypisují některé nastalé události.

Funkce jednotlivých tlačítek a zaškrtávacích polí je popsaná níže:

- Connectdisconnect - ovládá připojení k robotu.
- check coords - volí zda budou přijímány aktuální souřadnice.
- check pulses - volí zda budou přijímány aktuální hodnoty pulsů.
- check inputs - volí zda budou přijímány stavy vstupů.

- Free axes - uvolní ramena manipulátoru pro manuální nastavení.
- Lock axes - uzamkne ramena manipulátoru.
- Start - ovládá spuštění jednotlivých pohybů.
- Start PTP - ovládá spuštění pohybů typu PTP.
- loop PTP - spustí smyčku provádění PTP pohybů.
- Remove - odstraní řádek z tabulky.
- Pause - pozastaví provádění PTP pohybů.
- Finish - ukončí provádění PTP pohybů.

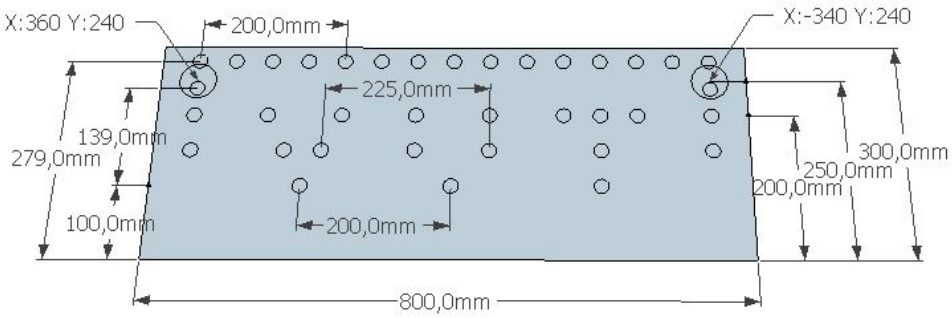


Obr. 4.3: Ovládací panel aplikace.

5 DEMONSTRACE SCHOPNOSTÍ MANIPULÁTORU

Pro ukázku funkce magnetického úchopu a prezentaci schopností robotu, bylo vybráno přeskládávání mincí z nápisu VUT na BRNO. Pro tento úkol byla vytvořena aplikace využívající třídu robotdriver s jednoduchým uživatelským rozhraním. Do něj stačí pouze zadat hodnotu souřadnice Z nad jednotlivými písmeny, Stisknout tlačítko *Connect* a po dokončení kalibrace spustit *Start*.

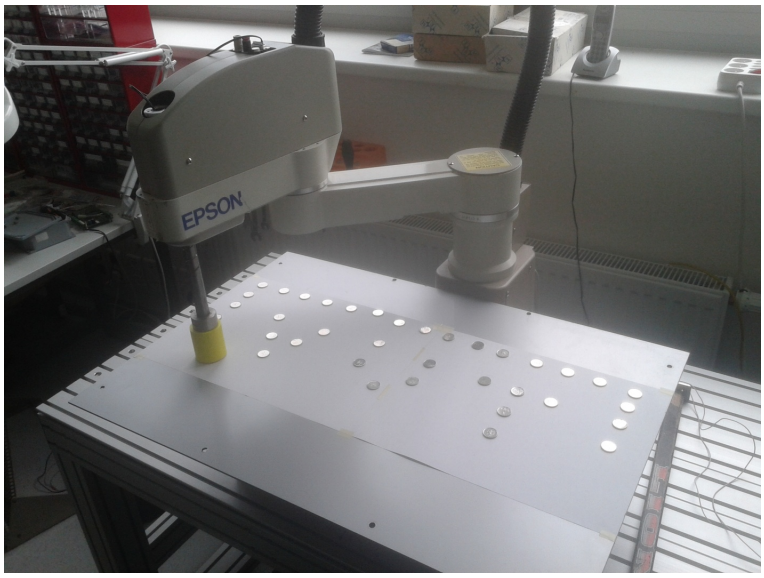
Přeskládávání probíhá na ploše o rozměrech přibližně 80 krát 30 cm. Před startem je třeba na plochu umístit mince v předepsaném rozložení tak, aby tvořily nápis VUT. Umístění jednotlivých mincí je popsáno na obrázku 5.1.



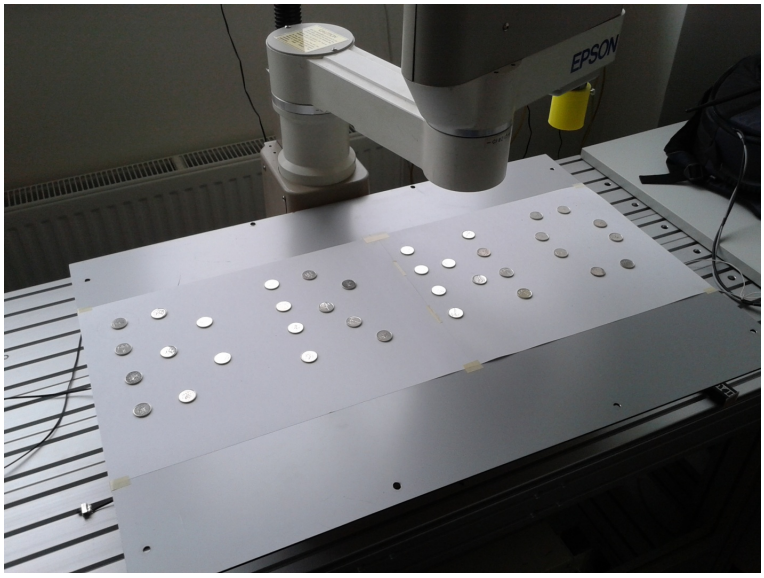
Obr. 5.1: Rozložení mincí.

V pravém a levém horním rohu se nacházejí větší kruhy které slouží jako referenční body určující správné umístění desky. K nastavení správné polohy byla využita předešlá aplikace s ovládáním pohybů. Po stisknutí tlačítka *Free axes* byly sledovány aktuální souřadnice a posunem ramene nad oba body bylo kontorolováno, zda jejich poloha odpovídá hodnotám v obrázku.

V aplikaci jsou hodnoty parametrů *CheckActPulse*, *CheckInputs* a *CheckActCoords* nastaveny na false. Tím je umožněno, aby byl *AskingPeriod* nastaven na 300 ms. Tím se urychlí odesílání požadavků kontroléru a tedy i získávání informací o jeho stavu. Po stisknutí tlačítka *start* jsou souřadnice bodů, mezi kterými se bude robot přesouvat uloženy do listu. Ten je poté použit jako vstupní parametr pro metodu **JumpPTP**. Parametr *LoopPTP* je nastaven na true, takže pohyb probíhá nepřetržitě. Nahrávání bodů do kontroléru chvíli trvá (kolem půl minuty) a poté je zahájen pohyb. Ovládání spínání a vypínání magnetu je řízeno částí kódu vypsanou v ukázce 5.1. Nejprve je série pohybů pozastavena metodou **pausePTP**. Robot tedy vykoná aktuální pohyb a zůstane stát. Dále program vstoupí do smyčky zkoumající hodnotu proměnné *pom*. Ta se změní na true po vyvolání události **onMoveEnd** a



Obr. 5.2: Skládani nápisu - VUT.



Obr. 5.3: Skládani nápisu - BRNO.

následně je na řádku 9 magnet sepnut. Pohyb je poté obnoven a hned zase pozastaven. Následně se celý proces opakuje s tím rozdílem, že na řádku 19 dojde k vypnutí magnetu.

```
1  public void Go(double[] coordinates)
2              while(true)
3              {
4              Robot.pausePTP();
5
6              while (pom == false){}
7              pom = false;
8
9              Robot.output(0, true);
10             System.Threading.Thread.Sleep(100);
11
12             Robot.unpausePTP();
13             System.Threading.Thread.Sleep(300);
14             Robot.pausePTP();
15
16             while (pom == false){}
17             pom = false;
18
19             Robot.output(0, false);
20             System.Threading.Thread.Sleep(100);
21
22             Robot.unpausePTP();
23             System.Threading.Thread.Sleep(300);
24             }
```

Ukázka kódu 5.1: Řízení spínání magnetu

6 ZÁVĚR

V rámci teoretické části bakalářské práce jsem se seznámil s manipulátorem H554BN typu SCARA a jeho kontrolérem. Na základě získaných znalostí byly sepsány informace o základních částech a vlastnostech robotu, které byly zařezeny do teoretické kapitoly nazvané Popis robotu Epson. Její první podkapitola se zabývala převážně stavbou, jednotlivými rozměry a rozsahem pohybu manipulátoru. V následující podkapitole pak byl sepsán význam jednotlivých částí kontroléru SRC 310. Byly nastudovány a sepsány jeho jednotlivé módy a také bezpečnostní prvky, jenž má zabudované. Poslední teoretická podkapitola stručně popisuje základní vlastnosti programovacího jazyka SPEL III, ve kterém jsou psány programy nahrávané do kontroléru.

V praktické části byl robot zprovozněn připevněním zástrčky k napájecímu kabelu a vytvořením potřebných konektorů. Jako optimální způsob řízení robota skrze PC bylo určeno ovládání programu nahraného do kontroléru. To je prováděno změnou jeho proměnných pomocí příkazů odesílaných třídou, napsanou v jazyce C#. Díky tomu je možné během provádění pohybů odesílat kontroléru další povely. Třída byla použita v naprogramované aplikaci s uživatelským grafickým rozhraním, skrze které je možné robota snadno ovládat. Dále byl navržen a sestrojen magnetický úchop, který byl následně využit v demonstrační úloze. Tou bylo opakované přeskládávání mincí z nápisu VUT na BRNO, které dopadlo zdárně a funkčnost úchopu i třídy RobotDriver tak byla ověřena.

LITERATURA

[1] NERUDA, Martin. *3D TISKÁRNA REBELIX: dostupná tiskárna se vším, co potřebujete k tisku* [online]. [cit. 2015-01-02]. Dostupné z: `http://reprap4u.cz/`

[2] SEIKO EPSON CORPORATION. I/O. *SRC-320: robot controller* [online]. Rev.4. s. 41-44. [cit. 2015-05-18]. Dostupné z: `http://www.dgrobot.com/products/EPSON%E6%8E%A7%E5%88%B6%E5%99%A8%E8%AF%B4%E6%98%8ESRC320_Controller_Manual%28R7%29.pdf`

[3] SEIKO EPSON CORPORATION. REMOTE1: Pin assignment. *SRC-300: robot controller* Rev.4. s. 32.

[4] SEIKO EPSON CORPORATION. Part Names and Functions: safety features. *SRC-300: robot controller* Rev.4. s. 9-10.

[5] SEIKO EPSON CORPORATION. When Not Using the Operating Unit: Operating unit disconnected (REMOTE2). *SRC-300: robot controller* Rev.4. s. 35.

[6] SEIKO EPSON CORPORATION. Part Names and Functions. *SRC-300: robot controller* Rev.4. s. 3-6.

[7] Relé Finder 40.52.9.012.0000, 12 V, 8 A. *RR NÁŘADÍ* [online]. [cit. 2014-12-16]. Dostupné z: `http://www.rr-naradi.cz/rele-finder-405290120000-12-v-8-a#prettyPhoto`

[8] SEIKO EPSON CORPORATION. User Wires and Air Tubes. *BN TYPE: SCARA robot* Rev.5. s. 12.

[9] SEIKO EPSON CORPORATION. Hands: Attaching a hand. *BN TYPE: SCARA robot* Rev.5. s. 12.

[10] SEIKO EPSON CORPORATION. Manipulator Part Names. *BN TYPE: SCARA robot* Rev.5. s. 1.

[11] SEIKO EPSON CORPORATION. External Dimensions. *BN TYPE: SCARA robot* Rev.5. s. 21.

[12] SEIKO EPSON CORPORATION. The Motion Range and Robot Coordinates. *BN TYPE: SCARA robot* Rev.5. s. 22-23.

[13] SCARA. In: *Robot Hall of Fame* [online]. 2006 [cit. 2014-12-25]. Dostupné z: `http://www.robothalloffame.org/inductees/06inductees/scara.html`

[14] SEIKO EPSON CORPORATION. PREPARATION FOR OPERATION: Mode. *User's manual for SRC-300/320: Epson robot* Rev.2. s. 19-21.

[15] SEIKO EPSON CORPORATION. BASIC FUNCTIONS FOR SAFETY. *User's manual for SRC-300/320: Epson robot* Rev.2. s. 2-4.

[16] SEIKO EPSON CORPORATION. *SPEL fo Windows Help V1.20E.*

[17] Materiály pro 3D tisk. FUTUR3D. *FUTUR3D: human and technology* [online]. [cit. 2015-01-03]. Dostupné z: `http://www.futur3d.net/materialy-pro-3d-tisk`

[18] SEIKO EPSON CORPORATION. The hand and operating acceleration/deceleration speed. *BN TYPE: SCARA robot* Rev.5. s. 15.

[19] Discontinued. EPSON. *EPSON* [online]. [cit. 2015-01-08]. Dostupné z: `http://global.epson.com/products/robots/support/discontinue/`

[20] SEIKO EPSON CORPORATION. Installation: Power. *SRC-300: robot controller* Rev.4. s. 14.

[21] SEIKO EPSON CORPORATION. TEACH Port. *SRC-300: robot controller* Rev.4. s. 21-25.

[22] SEIKO EPSON CORPORATION. *SPEL III Ver. 6.2: reference manual for SRC-3* Rev.4.

[23] SEIKO EPSON CORPORATION. APPLIED SECTION: RS-232C (Overview). *User's manual for SRC-300/320: Epson robot* Rev.2. s. 108.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

SCARA Selective Compliance Assembly Robot Arm

ABS Akrylonitrilbutadienstyren

PTP Point to point

I/O Input/Output

SEZNAM PŘÍLOH

A REFERENČNÍ MANUÁL TŘÍDY

A.1 Dokumentace třídy RobotDriver

Veřejné metody

- bool checkCoordValues (double x, double y, double z, double u, bool mode =false)

  Metoda kontrolující, zda se zadané souřadnice nacházejí v dosahu ramene manipulátoru.
- void output (uint num, bool state)

  Metoda pro spínání a vypínání výstupů kontroléru.
- void setAccel (uint A, uint B, uint C, uint D, uint E, uint F)

  Metoda nastavuje zrychlení a zpomalení pro pohyby GO, JUMP, PASS a PULSE. Hodnoty jsou zadávány v procentech.
- void setAccel (uint A, uint B)

  Metoda nastavuje zrychlení a zpomalení pro pohyby GO, JUMP, PASS a PULSE. Hodnoty jsou zadávány v procentech.
- void setAccels (uint A)

  Metoda nastavuje zrychlení pro pohyby ARC a MOVE.
- void freeAxes ()

  Metoda pro uvolnění ramen manipulátoru, která slouží ručnímu nastavování polohy.
- void lockAxes ()

  Metoda uzamknutí ramen manipulátoru (po jejich uvolnění).
- void stopPTP ()

  Metoda ukončující provádění série pohybů (PTP). Provede poslední pohyb a ukončí se.
- void pausePTP ()

  Metoda pozastavující provádění série pohybů (PTP). Provede poslední pohyb a zastaví se.
- void unpausePTP ()

  Metoda pro pokračování v sérii pohybů po pozastavení.
- void setSpeed (uint A, uint B, uint C)

  Metoda nastavuje rychlost pro pohyby GO, JUMP, PASS a PULSE. Hodnoty jsou zadávány v procentech.
- void setSpeeds (uint A)

  Metoda nastavuje rychlost pro pohyby ARC a MOVE.
- bool Connect (string address, string parameters="")

  Metoda pro připojení třídy k kontroléru. Po jejím odeslání dojde k zapnutí motorů manipulátoru a jeho kalibraci. Teprve po ní dojde k přijetí aktuálních souřadnic a změně stavu ControllerBusy na false. Při úspěšném připojení vrací hodnotu true.

- void Disconnect ()

  Metoda ukončující připojení mezi třídou a kontrolérem. Po jejím odeslání dojde k vypnutí motorů a přerušení komunikace.
- void Pulse (int[] pulses)

  Metoda vykonávající pohyb PULSE. Hodnoty pulsů pro nadbytečné osy budou ignorovány.
- void Pulse (int?axis1=null, int?axis2=null, int?axis3=null, int?axis4 =null, int?axis5=null, int?axis6=null)

  Metoda vykonávající pohyb PULSE. Hodnoty pulsů pro nadbytečné osy budou ignorovány.
- void PulsePTP (List< int[]> pulses)

  Metoda vykonávající sérii pohybů PULSE. Hodnoty pulsů pro nadbytečné osy budou ignorovány.
- void Go (double[] coordinates)

  Metoda pro vykonání pohybu GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Go (double x=double.NaN, double y=double.NaN, double z=double.↩ NaN, double u=double.NaN, double v=double.NaN, double w=double.NaN)

  Metoda pro vykonání pohybu GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void GoPTP (List< double[]> coordinates)

  Metoda pro vykonání série pohybů GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Go (uint input, double[] coordinates)

  Metoda pro vykonání pohybu GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Go (uint input, double x=double.NaN, double y=double.NaN, double z =double.NaN, double u =double.NaN, double v =double.NaN, double w =double.NaN)

  Metoda pro vykonání pohybu GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void GoPTP (uint input, List< double[]> coordinates)

  Metoda pro vykonání série pohybů GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Jump (uint q, double l, double[] coordinates)

  Metoda pro vykonání pohybu JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Jump (uint q, double l, double x=double.NaN, double y=double.NaN, double z=double.NaN, double u=double.NaN, double v=double.NaN, double w=double.NaN)

  Metoda pro vykonání pohybu JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

- void JumpPTP (uint q, double l, List< double[]> coordinates)

  Metoda vykonávající sérii pohybů JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Jump (bool mode, uint input, uint q, double l, double[] coordinates)

  Metoda pro vykonání pohybu JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Jump (bool mode, uint input, uint q, double l, double x=double.NaN, double y=double.NaN, double z=double.NaN, double u=double.NaN, double v=double.NaN, double w=double.NaN)

  Metoda pro vykonání pohybu JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void JumpPTP (bool mode, uint input, uint q, double l, List< double[]> coordinates)

  Metoda vykonávající sérii pohybů JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Arc (double[] coordinates, double[] coordinates1)

  Metoda vykonávající pohyb ARC. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Arc (double x=double.NaN, double y=double.NaN, double z=double.↩ NaN, double u=double.NaN, double x1=double.NaN, double y1=double.NaN, double z1=double.NaN, double u1=double.NaN)

  Metoda vykonávající pohyb ARC. Hodnoty souřadnic jsou zadány v milimetrech.
- void Arc (List< double[]> coordinates)

  Metoda vykonávající sérii puhybů typu ARC. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Move (uint input, double[] coordinates)

  Metoda provádějící pohyb MOVE. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Move (uint input, double x=double.NaN, double y=double.NaN, double z=double.NaN, double u =double.NaN, double v =double.NaN, double w =double.NaN)

  Metoda provádějící pohyb MOVE. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void MovePTP (uint input, List< double[]> coordinates)

  Metoda provádějící sérii pohybů MOVE. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
- void Pass (List< double[]> coordinates)

  Metoda provádějící pohyb PASS. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Vlastnosti

- string DefaultConnectionParameters `[get]`
  Vrací řetězec s informací o parametrech komunikace.
- string Connected `[get]`
  Vrací řetězec obsahující "true", pokud je robot připojen.
- double[] RobotPosition `[get]`
  Vrací aktuální souřadnice uložené v řadě hodnot typu double. Pořadí souřadnic je: x,y,z,u
- double RobotX `[get]`
  Vrací aktuální hodnotu souřadnice x
- double RobotY `[get]`
  Vrací aktuální hodnotu souřadnice y
- double RobotZ `[get]`
  Vrací aktuální hodnotu souřadnice z
- double RobotU `[get]`
  Vrací aktuální hodnotu souřadnice u
- double RobotV `[get]`
  Vrací NaN
- double RobotW `[get]`
  Vrací NaN
- int PulseX `[get]`
  Vrací aktuální hodnotu pulsů osy x
- int PulseY `[get]`
  Vrací aktuální hodnotu pulsů osy y
- int PulseZ `[get]`
  Vrací aktuální hodnotu pulsů osy z
- int PulseU `[get]`
  Vrací aktuální hodnotu pulsů osy u
- bool Busy `[get]`
  Vrací informaci o stavu kontroléru. Pokud je hodnota true, pak kontrolér zpracovává předešlý příkaz.
- bool controllerWorking `[get]`
  Vrací hodnotu informující zda je kontrolér zaneprázdněn (například prováděním pohybu).
- string ConnectionTimeout `[get, set]`
  Není inplementováno
- bool CheckActCoords `[get, set]`
  Vrací true pokud jsou kontrolovány aktuální souřadnice. Nastavuje kontrolu aktuálních souřadnic.
- bool CheckActPulse `[get, set]`
- double AskingPeriod `[get, set]`
  Vrací a nastavuje hodnotu periody pro odesílání požadavků do kontroléru (v milisekundách)

- bool CheckInputs `[get, set]`
- bool Input0 `[get]`

  Vrací hodnotu vstupu 0 na kontroléru.
- bool Input1 `[get]`
- bool Input2 `[get]`
- bool Input3 `[get]`
- bool Input4 `[get]`
- bool LoopPTP `[get, set]`

  Vrací a nastavuje zda je série pohybů (PTP) provádena cyklicky.

Události

- EventHandler onConnect

  Je vyvolána po připojení třídy k robotu. (Po dokončení kalibrace)
- EventHandler onDisconnect

  Je vyvolána po ukončení připojení mezi třídou a robotem.
- EventHandler onConnectionLost

  Je vyvolána po jestliže bylo připojení ztraceno vlivem odpojení nebo chybou.
- EventHandler onCommandConfirmed

  Je vyvolána pokud je poslední příkaz kontrolérem zpracován.
- EventHandler onPTPEnd

  Je vyvolána po dokončení série pohybú (PTP).
- EventHandler onMoveEnd

  Je vyvolána po dokončení pohybu.
- EventHandler onMessage

  Je vyvolána po přijetí zprávy z kontroléru.
- EventHandler onError

  Je vyvolána pokud došlo k chybě.

A.1.1 Dokumentace k metodám

void RobotDriver.Arc (double[] *coordinates*, double[] *coordinates1*)

Metoda vykonávající pohyb ARC. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

coordinates	Hodnoty souřadnic bodu 1 v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

coordinates1	Hodnoty souřadnic bodu 2 v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.Arc (double *x* = `double.NaN`, double *y* = `double.NaN`, double *z* = `double.NaN`, double *u* = `double.NaN`, double *x1* = `double.NaN`, double *y1* = `double.NaN`, double *z1* = `double.NaN`, double *u1* = `double.NaN`)

Metoda vykonávající pohyb ARC. Hodnoty souřadnic jsou zadány v milimetrech.
Parametry

x	Souřadnice x bodu 1. Rozsah 550 až -550.
y	Souřadnice y bodu 1. Rozsah 550 až -550.
z	Souřadnice z bodu 1. Rozsah 0 až -150.
u	Souřadnice u bodu 1. Rozsah 450 až -270.
x1	Souřadnice x bodu 2. Rozsah 550 až -550.
y1	Souřadnice y bodu 2. Rozsah 550 až -550.
z1	Souřadnice z bodu 2. Rozsah 0 až -150.
u1	Souřadnice u bodu 2. Rozsah 450 až -270.

void RobotDriver.Arc (List< double[]> *coordinates*)

Metoda vykonávající sérii puhybů typu ARC. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
Parametry

coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Uložené do listu v pořadí: bod 1,bod 2,bod 1,bod 2... Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

bool RobotDriver.checkCoordValues (double *x*, double *y*, double *z*, double *u*, bool *mode* = `false`)

Metoda kontrolující, zda se zadané souřadnice nacházejí v dosahu ramene manipulátoru.

Parametry

x	Souřadnice x.
y	Souřadnice y.
z	Souřadnice z.
u	Souřadnice u.
mode	Proměnná přepínající mód metody. Je-li mode true, metoda vrátí hodnotu true, pokud je souřadnice v dosahu. Jestliže je mode false, metoda vyhodí vyjimku při špatných hodnotách souřadnic.

Návratová hodnota

bool RobotDriver.Connect (string *address*, string *parameters* = "")

Metoda pro připojení třídy k kontroléru. Po jejím odeslání dojde k zapnutí motorů manipulátoru a jeho kalibraci. Teprve po ní dojde k přijetí aktuálních souřadnic a změně stavu ControllerBusy na false. Při úspěšném připojení vrací hodnotu true.

Parametry

address	Adresa sériového portu
parameters	Parametry komunikace (nejsou využívány)

Návratová hodnota

void RobotDriver.Disconnect ( )

Metoda ukončující připojení mezi třídou a kontrolérem. Po jejím odeslání dojde k vypnutí motorů a přerušení komunikace.

void RobotDriver.freeAxes ( )

Metoda pro uvolnění ramen manipulátoru, která slouží ručnímu nastavování polohy.

void RobotDriver.Go (double[] *coordinates*)

Metoda pro vykonání pohybu GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.Go (double *x* = `double.NaN`, double *y* = `double.NaN`, double *z* = `double.NaN`, double *u* = `double.NaN`, double *v* = `double.NaN`, double *w* = `double.NaN`)

Metoda pro vykonání pohybu GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

x	Souřadnice x. Rozsah 550 až -550.
y	Souřadnice y. Rozsah 550 až -550.
z	Souřadnice z. Rozsah 0 až -150.
u	Souřadnice u. Rozsah 450 až -270.
v	Souřadnice v. Nepoužita.
w	Souřadnice w. Nepoužita

void RobotDriver.Go (uint *input*, double[] *coordinates*)

Metoda pro vykonání pohybu GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

input	Číslo vstupu ovládajícího zastavení pohybu.
coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.Go (uint *input*, double *x* = `double.NaN`, double *y* = `double.NaN`, double *z* = `double.NaN`, double *u* = `double.NaN`, double *v* = `double.NaN`, double *w* = `double.NaN`)

Metoda pro vykonání pohybu GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

input	Číslo vstupu ovládajícího zastavení pohybu (sepnuto = stát).
x	Souřadnice x. Rozsah 550 až -550.
y	Souřadnice y. Rozsah 550 až -550.
z	Souřadnice z. Rozsah 0 až -150.
u	Souřadnice u. Rozsah 450 až -270.
v	Souřadnice v. Nepoužita.
w	Souřadnice w. Nepoužita

void RobotDriver.GoPTP (List< double[]> *coordinates*)

Metoda pro vykonání série pohybů GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.GoPTP (uint *input,* List< double[]> *coordinates*)

Metoda pro vykonání série pohybů GO. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

input	Číslo vstupu ovládajícího zastavení pohybu (sepnuto = stát).
coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.Jump (uint *q,* double *l,* double[] *coordinates*)

Metoda pro vykonání pohybu JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

q	Parametr volící oblouk,vykonaný při pohybu. Hodnoty v rozsahu 0 až 7.

l	Parametr nastavující výšku oblouku. Hodnoty 0 až -150 (mm).
coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.Jump (uint *q*, double *l*, double *x* = `double.NaN`, double *y* = `double.NaN`, double *z* = `double.NaN`, double *u* = `double.NaN`, double *v* = `double.NaN`, double *w* = `double.NaN`)

Metoda pro vykonání pohybu JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

q	Parametr volící oblouk,vykonaný při pohybu. Hodnoty v rozsahu 0 až 7.
l	Parametr nastavující výšku oblouku. Hodnoty 0 až -150 (mm).
x	Souřadnice x. Rozsah 550 až -550.
y	Souřadnice y. Rozsah 550 až -550.
z	Souřadnice z. Rozsah 0 až -150.
u	Souřadnice u. Rozsah 450 až -270.
v	Souřadnice v. Nepoužita.
w	Souřadnice w. Nepoužita

void RobotDriver.Jump (bool *mode*, uint *input*, uint *q*, double *l*, double[] *coordinates*)

Metoda pro vykonání pohybu JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

mode	Volba mezi zastavováním horizontálního, nebo vertikálního pohybu (SENSE,TILL). Pokud je true: vertikálního (SENSE).
input	Volba vstupu pro řízení zastavení pohybu (sepnuto = stát).
q	Parametr volící oblouk,vykonaný při pohybu. Hodnoty v rozsahu 0 až 7.

l	Parametr nastavující výšku oblouku. Hodnoty 0 až -150 (mm).
coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.Jump (bool *mode,* uint *input,* uint *q,* double *l,* double *x* = `double.NaN`, double *y* = `double.NaN`, double *z* = `double.NaN`, double *u* = `double.NaN`, double *v* = `double.NaN`, double *w* = `double.NaN`)

Metoda pro vykonání pohybu JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

mode	Volba mezi zastavováním horizontálního, nebo vertikálního pohybu (SENSE,TILL). Pokud je true: vertikálního (SENSE).
input	Volba vstupu pro řízení zastavení pohybu (sepnuto = stát).
q	Parametr volící oblouk,vykonaný při pohybu. Hodnoty v rozsahu 0 až 7.
l	Parametr nastavující výšku oblouku. Hodnoty 0 až -150 (mm).
x	Souřadnice x. Rozsah 550 až -550.
y	Souřadnice y. Rozsah 550 až -550.
z	Souřadnice z. Rozsah 0 až -150.
u	Souřadnice u. Rozsah 450 až -270.
v	Souřadnice v. Nepoužita.
w	Souřadnice w. Nepoužita

void RobotDriver.JumpPTP (uint *q,* double *l,* List< double[]> *coordinates*)

Metoda vykonávající sérii pohybů JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

q	Parametr volící oblouk,vykonaný při pohybu. Hodnoty v rozsahu 0 až 7.

l	Parametr nastavující výšku oblouku. Hodnoty 0 až -150 (mm).
coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.JumpPTP (bool *mode,* uint *input,* uint *q,* double *l,* List< double[]> *coordinates*)

Metoda vykonávající sérii pohybů JUMP. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
Parametry

mode	Volba mezi zastavováním horizontálního, nebo vertikálního pohybu (SENSE,TILL). Pokud je true: vertikálního (SENSE).
input	Volba vstupu pro řízení zastavení pohybu (sepnuto = stát).
q	Parametr volící oblouk,vykonaný při pohybu. Hodnoty v rozsahu 0 až 7.
l	Parametr nastavující výšku oblouku. Hodnoty 0 až -150 (mm).
coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.lockAxes ( )

Metoda uzamknutí ramen manipulátoru (po jejich uvolnění).

void RobotDriver.Move (uint *input,* double[] *coordinates*)

Metoda provádějící pohyb MOVE. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.
Parametry

input	Vstup ovládající zastavení pohybu. (sepnuto = stát)
coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.Move (uint *input,* double *x* = `double.NaN`, double *y* = `double.NaN`, double *z* = `double.NaN`, double *u* = `double.NaN`, double *v* = `double.NaN`, double *w* = `double.NaN`)

Metoda provádějící pohyb MOVE. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

input	Vstup ovládající zastavení pohybu. (sepnuto = stát)
x	Souřadnice x. Rozsah 550 až -550.
y	Souřadnice y. Rozsah 550 až -550.
z	Souřadnice z. Rozsah 0 až -150.
u	Souřadnice u. Rozsah 450 až -270.
v	Souřadnice v. Nepoužita.
w	Souřadnice w. Nepoužita

void RobotDriver.MovePTP (uint *input,* List< double[]> *coordinates*)

Metoda provádějící sérii pohybů MOVE. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

input	Vstup ovládající zastavení pohybu. (sepnuto = stát)
coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270.

void RobotDriver.output (uint *num,* bool *state*)

Metoda pro spínání a vypínání výstupů kontroléru.

Parametry

num	Číslo přepínaného výstupu. Volí se v rozmezí 0 až 15.
state	Stav výstupu. Při true je sepnuto.

void RobotDriver.Pass (List< double[]> *coordinates*)

Metoda provádějící pohyb PASS. Hodnoty souřadnic jsou zadány v milimetrech. Nadbytečné souřadnice budou ignorovány.

Parametry

coordinates	Hodnoty souřadnic v pořadí x,y,z,u. Rozsah souřadnice x: 550 až -550, y: 550 až -550, z: 0 až -150, u: 450 až -270. Přesun mezi jednotlivými body je vykonán jako jeden pohyb.

void RobotDriver.pausePTP ( )

Metoda pozastavující provádění série pohybů (PTP). Provede poslední pohyb a zastaví se.

void RobotDriver.Pulse (int[] *pulses*)

Metoda vykonávající pohyb PULSE. Hodnoty pulsů pro nadbytečné osy budou ignorovány.

Parametry

pulses	Hodnoty pulsů zadané v postupně od osy 1 až 4. Rozsah osa 1↩ : 364089 až -36409, osa 2: 159289 až -159289, osa 3: 0 až -92160, osa 4: 172032 až -172032.

void RobotDriver.Pulse (int? *axis1* = `null`, int? *axis2* = `null`, int? *axis3* = `null`, int? *axis4* = `null`, int? *axis5* = `null`, int? *axis6* = `null`)

Metoda vykonávající pohyb PULSE. Hodnoty pulsů pro nadbytečné osy budou ignorovány.

Parametry

axis1	Hodnota pulsů pro osu 1. Rozsah 364089 až -36409.
axis2	Hodnota pulsů pro osu 2. Rozsah 159289 až -159289.
axis3	Hodnota pulsů pro osu 3. Rozsah 0 až -92160.
axis4	Hodnota pulsů pro osu 4. Rozsah 172032 až -172032.
axis5	Nepoužito.
axis6	Nepoužito.

void RobotDriver.PulsePTP (List< int[]> *pulses*)

Metoda vykonávající sérii pohybů PULSE. Hodnoty pulsů pro nadbytečné osy budou ignorovány.

Parametry

pulses	Hodnoty pulsů zadané v postupně od osy 1 až 4. Rozsah osa 1↩ : 364089 až -36409, osa 2: 159289 až -159289, osa 3: 0 až -92160, osa 4: 172032 až -172032.

void RobotDriver.setAccel (uint *A*, uint *B*, uint *C*, uint *D*, uint *E*, uint *F*)

Metoda nastavuje zrychlení a zpomalení pro pohyby GO, JUMP, PASS a PULSE. Hodnoty jsou zadávány v procentech.

Parametry

A	Nastavuje zrychlení. Rozsah hodnot 1 až 100.
B	Nastavuje zpomalení. Rozsah hodnot 1 až 100.
C	Nastavuje zrychlení osy 3 (nahoru). Rozsah hodnot 1 až 100.
D	Nastavuje zpomalení osy 3 (nahoru). Rozsah hodnot 1 až 100.
E	Nastavuje zrychlení osy 3 (dolů). Rozsah hodnot 1 až 100.
F	Nastavuje zpomalení osy 3 (dolů). Rozsah hodnot 1 až 100.

void RobotDriver.setAccel (uint *A*, uint *B*)

Metoda nastavuje zrychlení a zpomalení pro pohyby GO, JUMP, PASS a PULSE. Hodnoty jsou zadávány v procentech.

Parametry

A	Nastavuje zrychlení. Rozsah hodnot 1 až 100.
B	Nastavuje zpomalení. Rozsah hodnot 1 až 100.

void RobotDriver.setAccels (uint *A*)

Metoda nastavuje zrychlení pro pohyby ARC a MOVE.

Parametry

A	Hodnota zrychlení zadaná v mm∗s^(-2). Rozsah hodnoty je 1 až 5000.

void RobotDriver.setSpeed (uint *A*, uint *B*, uint *C*)

Metoda nastavuje rychlost pro pohyby GO, JUMP, PASS a PULSE. Hodnoty jsou zadávány v procentech.

Parametry

A	Nastavuje rychlost. Rozsah hodnot 1 až 100.
B	Nastavuje rychlost osy 3 (nahoru). Rozsah hodnot 1 až 100.
C	Nastavuje rychlost osy 3 (dolů). Rozsah hodnot 1 až 100.

void RobotDriver.setSpeeds (uint *A*)

Metoda nastavuje rychlost pro pohyby ARC a MOVE.

Parametry

A	Hodnota rychlosti zadaná v mm/s. Rozsah hodnoty je 1 až 1120.

void RobotDriver.stopPTP ( )

Metoda ukončující provádění série pohybů (PTP). Provede poslední pohyb a ukončí se.

void RobotDriver.unpausePTP ( )

Metoda pro pokračování v sérii pohybů po pozastavení.

A.1.2 Dokumentace k vlastnostem

double RobotDriver.AskingPeriod `[get]`, `[set]`

Vrací a nastavuje hodnotu periody pro odesílání požadavků do kontroléru (v milisekundách)

bool RobotDriver.Busy `[get]`

Vrací informaci o stavu kontroléru. Pokud je hodnota true, pak kontrolér zpracovává předešlý příkaz.

bool RobotDriver.CheckActCoords `[get]`, `[set]`

Vrací true pokud jsou kontrolovány aktuální souřadnice. Nastavuje kontrolu aktuálních souřadnic.

bool RobotDriver.CheckActPulse `[get]`, `[set]`

Vrací true pokud jsou kontrolovány aktuální hodnoty pulsů. Nastavuje kontrolu aktuálních hodnot pulsů.

bool RobotDriver.CheckInputs `[get]`, `[set]`

Vrací true pokud jsou kontrolovány hodnoty vstupů kontroléru. Nastavuje kontrolu hodnot vstupů kontroléru.

string RobotDriver.Connected `[get]`

Vrací řetězec obsahující "true", pokud je robot připojen.

string RobotDriver.ConnectionTimeout `[get]`, `[set]`

Není inplementováno

bool RobotDriver.controllerWorking `[get]`

Vrací hodnotu informující zda je kontrolér zaneprázdněn (například prováděním pohybu).

string RobotDriver.DefaultConnectionParameters `[get]`

Vrací řetězec s informací o parametrech komunikace.

bool RobotDriver.Input0 `[get]`

Vrací hodnotu vstupu 0 na kontroléru.

bool RobotDriver.Input1 `[get]`

Vrací hodnotu vstupu 1 na kontroléru.

bool RobotDriver.Input2 `[get]`

Vrací hodnotu vstupu 2 na kontroléru.

bool RobotDriver.Input3 `[get]`

Vrací hodnotu vstupu 3 na kontroléru.

bool RobotDriver.Input4 `[get]`

Vrací hodnotu vstupu 4 na kontroléru.

bool RobotDriver.LoopPTP `[get]`, `[set]`

Vrací a nastavuje zda je série pohybů (PTP) provádena cyklicky.

int RobotDriver.PulseU `[get]`

Vrací aktuální hodnotu pulsů osy u

int RobotDriver.PulseX `[get]`

Vrací aktuální hodnotu pulsů osy x

int RobotDriver.PulseY `[get]`

Vrací aktuální hodnotu pulsů osy y

int RobotDriver.PulseZ `[get]`

Vrací aktuální hodnotu pulsů osy z

double [] RobotDriver.RobotPosition `[get]`

Vrací aktuální souřadnice uložené v řadě hodnot typu double. Pořadí souřadnic je: x,y,z,u

double RobotDriver.RobotU `[get]`

Vrací aktuální hodnotu souřadnice u

double RobotDriver.RobotV `[get]`

Vrací NaN

double RobotDriver.RobotW `[get]`

Vrací NaN

double RobotDriver.RobotX `[get]`

Vrací aktuální hodnotu souřadnice x

double RobotDriver.RobotY `[get]`

Vrací aktuální hodnotu souřadnice y

double RobotDriver.RobotZ `[get]`

Vrací aktuální hodnotu souřadnice z

A.1.3 Dokumentace událostí

EventHandler RobotDriver.onCommandConfirmed

Je vyvolána pokud je poslední příkaz kontrolérem zpracován.

EventHandler RobotDriver.onConnect

Je vyvolána po připojení třídy k robotu. (Po dokončení kalibrace)

EventHandler RobotDriver.onConnectionLost

Je vyvolána po jestliže bylo připojení ztraceno vlivem odpojení nebo chybou.

EventHandler RobotDriver.onDisconnect

Je vyvolána po ukončení připojení mezi třídou a robotem.

EventHandler RobotDriver.onError

Je vyvolána pokud došlo k chybě.

EventHandler RobotDriver.onMessage

Je vyvolána po přijetí zprávy z kontroléru.

EventHandler RobotDriver.onMoveEnd

Je vyvolána po dokončení pohybu.

EventHandler RobotDriver.onPTPEnd

Je vyvolána po dokončení série pohybú (PTP).

B CD-ROM

B.1 Obsah:

- Bakalářská práce (PDF)
- Zdrojové kódy ovladače.
- Zdrojové kódy demonstrační aplikace.