VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Jiří Novotný

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ŘÍZENÍ KOLABORATIVNÍHO ROBOTA POMOCÍ PLC
COLLABORATIVE ROBOT CONTROL BY PLC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jiří Novotný**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Macho, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Automatizační a měřicí technika**

Ústav automatizace a měřicí techniky

Student: Jiří Novotný ***ID:*** 203308
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Řízení kolaborativního robota pomocí PLC

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se kolaborativním robotem YuMi firmy ABB a programovatelným automatem (PLC) SIMATIC S7-1500 firmy Siemens.
2. Definujte požadavky na komunikační rozhraní mezi robotem a PLC.
3. Na základě požadavků navrhněte komunikační rozhraní mezi robotem a PLC.
4. Navrhněte vhodnou úlohu demonstrující ovládání robota pomocí PLC.
5. Implementujte komunikační rozhraní mezi robotem a PLC.
6. Realizujte úlohu demonstrující ovládání robota pomocí PLC.
7. Vyhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

Jirgl, M. Laboratorní cvičení BPGA - Cvičení se systémy SIEMENS. FEKT VUT v Brně.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: Ing. Tomáš Macho, Ph.D.

doc. Ing. Václav Jirsík, CSc.
předseda rady studijního programu

ABSTRAKT

Tato bakalářské práce se zabývá řízením kolaborativního robota pomocí PLC. K realizaci této práce je využíván kolaborativní robot YuMi firmy ABB a PLC firmy Siemens nejnovější řady S7-1500. Seznámíme se s jednotlivými komponenty a s vývojovým prostředím pro PLC (TIA Portal) a pro robota (RobotStudio). Definujeme si požadavky na komunikační rozhraní mezi robotem a PLC. Popíšeme si demonstrující úlohu vyžadující ovládání robota pomocí PLC. Úloha demonstruje reálnou aplikaci pro koncového zákazníka. Následně implementujeme námi zvolené komunikační rozhraní PROFINET a realizujeme popsanou úlohu pro robota. Výsledná aplikace robota je poté odzkoušena v reálném výrobním provozu.

KLÍČOVÁ SLOVA

PLC, kolaborativní robot, TIA Portal, RobotStudio, komunikační rozhraní, PROFINET, řízení

ABSTRACT

This bachelor thesis deals with the control of a collaborative robot using a PLC. To implement this work is used collaborative robot YuMi from ABB and PLC from Siemens of latest series S7-1500. We will get acquainted with products which have been used in project and with the development environment for PLC (TIA Portal) and for robot (RobotStudio). We define the requirements for the communication interface between the robot and the PLC. We will describe a demonstrating task requiring control of a robot using a PLC. The task demonstrates a real application for the end user. Then we implement PROFINET communication interface and implement the described task for the robot. The final solution is then tested in real production line.

KEYWORDS

PLC, collaborative robot, TIA Portal, RobotStudio, communication interface, PROFINET, control

NOVOTNÝ, Jiří. *Řízení kolaborativního robota pomocí PLC*. Brno, Rok, 59 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Tomáš Macho, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Řízení kolaborativního robota pomocí PLC“ 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 8.6.2020 .....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing.Tomáši Macho, Ph.D. za odborné vedení, konzultace a podnětné návrhy k práci. Dále bych rád poděkoval firmě JHV - ENGINEERING s.r.o a především panu Ing. Stanislavu Pšeničkovi za odbornou pomoc a možnost realizace této práce.

Brno 8.6.2020 .....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Nedostatek zaměstnanců a kvalifikovaných pracovníků je jedním z hlavních důvodů, proč se automatizace výroby v poslední době tak rychle rozšířila. Zejména je tomu tak u robotických pracovišť, kde roboti jsou schopni pracovat na složitých úkolech bez asistence odborníků. Roboti zvyšují efektivitu výroby, eliminují výrobu zmetků a urychlují výrobní procesy. V prostorách, ve kterých robot pracuje, je nutné dodržovat mnohá bezpečnostní opatření před vnikem osoby a zabránění úrazu. Ne vždy je však možné eliminovat možnost kontaktu robota s člověkem nebo v některých případech dokonce požadujeme, aby robot pracoval ve společném prostoru s člověkem. V takových případech se využívá kolaborativních robotů.

Kolaborativní robot je opatřen mnohými bezpečnostními prvky a funkcemi, které umožňují bezpečný výskyt člověka v okolí robota. Pokud robot zaznamená kontakt s překážkou dochází k okamžitému zastavení robota nebo bezpečnému zpomalení pohybu. Kolaborativní režim robota je možné využít i k současné spolupráci s člověkem.

V této práci si definujeme pojem kolaborativní robot. Seznámíme se s kolaborativním robotem YuMi firmy ABB a s jeho možnostmi aplikace. Řekneme si, jakými způsoby jsme schopni robota řídit pomocí programovatelného automatu (PLC) SIMATIC S7-1500 firmy Siemens. Definujeme si požadavky na komunikační rozhraní mezi robotem a PLC a následně se pokusíme definovat nejlepší možnost komunikace.

Popíšeme si demonstrující úlohu vyžadující ovládání robota pomocí PLC. Úloha demonstruje reálnou aplikaci pro koncového zákazníka. Následně implementujeme námi zvolené komunikační rozhraní a realizujeme popsanou úlohu pro robota.

Tato práce je realizována ve firmě JHV - ENGINEERING s.r.o. zabývající se návrhem, konstrukcí a programováním robotických pracovišť a výrobních linek.

Cílem této práce je v praxi si odzkoušet možnost řízení kolaborativního robota pomocí PLC a realizovat zadanou úlohu pro robota s kvalitou požadovanou koncovým zákazníkem.

2 Seznámení s kolaborativním robotem a PLC

V této kapitole si řekneme, co je to kolaborativní robot. Definujeme hlavní rozdíly mezi robotem a kolaborativním robotem a nastíníme možnosti využití v praxi. Dále si řekneme, co je to programovatelný automat (PLC) a popíšeme si jeho základní vlastnosti.

2.1 Kolaborativní robot

Jak již bylo v úvodu naznačeno, kolaborativní robot, nebo také označován jako kooperativní robot či zkráceně kobot, je robot, který je schopen svými bezpečnostními prvky a funkcemi zajistit ve svém pracovním prostoru bezpečnou spolupráci robota s člověkem. Kolaborativní roboti pomáhají lidem vykonávat zejména obtížné a únavné práce. Většinou jde zpravidla o jednodušší úlohy (složitější jsou výjimkou) a bezpečnost bývá zajištěna přímo robotem. To se například vyznačuje tím, že pokud robot zaznamená kontakt s překážkou dochází k okamžitému zastavení robota nebo bezpečnému zpomalení pohybu, což zabrání případnému vzniku úrazu pracovníka. Kolaborativní roboti mohou pracovat přímo vedle člověka bez jakýchkoli zábran.

Kdežto konvenční průmyslové roboty pracují obvykle v prostorech, které jsou pro člověka nebezpečné nebo nepříjemné, a kde se tedy ani nepočítá s tím, že by zde roboty spolupracovaly s lidskou obsluhou, protože je snaha přítomnost osob v takovém prostředí zcela vyloučit. Pracují tedy v oddělených prostorech, které musí byt striktně izolovány od lidské obsluhy. Bezpečnost je tedy zajištěna externími snímači, spínači, klecí, mechanickým oplocením nebo optickými závorami, které jsou přímo napojeny na bezpečnostní řídící systém a při jakémkoli narušení zastaví nebo výrazně omezí pohyb robota.[1, 3]

Obr. 2.1: Příklad izolace robota oplocením a optickými závorami[1]

Obr. 2.2: Příklad kolaborativního robota v přímém kontaktu s člověkem[2]

Tedy hlavním rozdílem mezi kobotem a robotem je v provedení bezpečnosti. Přičemž všechny roboti (konvenční i kolaborativní) musí splňovat bezpečnostní normy, mezi ty nejdůležitější se řadí mezinárodní normy ISO 10218-1 a ISO 10218-2. V České republice tyto normy nalezneme pod označením ČSN EN ISO 10218-1 a ČSN EN ISO 10218-2.[3]

Roboti disponují velkým kroutícím momentem a tlačnou silou a proto je nutné dodržovat přísná bezpečnostní opatření. Podle norem, které jsme v předchozím odstavci zmínili, musí být bezpečnost zajištěna některou z uvedených funkcí (první tři jsou určeny pro konvenční průmyslové roboty, zatímco čtvrtá funkce, omezení síly a výkonu, je navržena pro koboty).

- Bezpečnostní monitorované zastavení – vstoupí-li jakákoli osoba do monitorovaného prostoru, robot se řízeným způsobem zastaví. Tato funkce je typická pro již dříve zmíněný způsob zábran, kde je bezpečnost zajištěna klecí, mechanickým oplocením nebo optickými závorami.
- Ruční navádění – po vstupu obsluhy do monitorovaného prostoru se robot řízeným způsobem zastaví. Následně obsluha může použít povolovací tlačítko a tím uvolnit pohyb robotické paže pro ruční navádění. Tato funkce slouží taktéž pro programování robotů učením, kdy například obsluha vede paži robotu po požadované dráze a robot se tento pohyb naučí a opakuje ho.
- Sledování rychlosti a vzdálenosti – bezpečnostní snímače sledují pohyby robotu a jeho vzdálenost od obsluhy. Hlídají minimální vzdálenost mezi pohybujícím se robotem a obsluhou. Pokud tuto vzdálenost robot nebo obsluha překročí, musí se robot postupně zpomalit až dojde k zastavení. Robot vybavený touto funkcí sice nemusí být uzavřen v kleci, ale nejedná se o kobota (nelze s ním bezprostředně spolupracovat).
- Omezení síly a výkonu – funkce pro zajištění bezpečnosti kolaborativních robotů. Jedná se o onu zmiňovanou vlastnost kdy robot v případě kontaktu s člověkem musí okamžitě zastavit nebo výrazně omezit svůj pohyb.

Mez stanovující možnou sílu působící na obsluhu robotem je stanovena specifikací ISO/TS 15066, která doplňuje normu ČSN EN ISO 10218.[1]

Technické omezení síly je zajištěno buď konstrukčním provedením nebo senzorikou na robotovi. Koboti mají obecně pomalejší pohyb ramen robotu, než konvenční roboti a to tak, aby byl robot schopen okamžitě zastavit (omezení hybnosti). Dále jsou většinou konstrukčně odlehčeni, s čímž se spojuje i menší nosnost, mají zaoblené tvary a v některých případech změkčenou povrchovou úpravu ramen. Koboti mají většinou zabudovány rychlostní a momentové senzory v kloubech nebo v základně robota. Také je možné provedení s citlivou vrstvou na povrchu ramene, která je tvořena velkoplošným kapacitním senzorem, který detekuje přibližování objektu k ramenu robotu a vydá pokyn k zabrzdění tak, aby v okamžiku doteku již rychlost

vyhovovala požadavkům bezpečnostních norem.[1]

Opatření pro zajištění bezpečnosti na pracovišť s kolaborativními roboty nejsou pouze ve funkcích robota, měli by být dodržovány i standardní bezpečnostní opatření dané legislativou a bezpečnostními postupy, předpisy a opatřenímy(nouzové zastavení, zabezpečení přístupu, možnost úniku z nebezpečné zóny, atd.). [3]

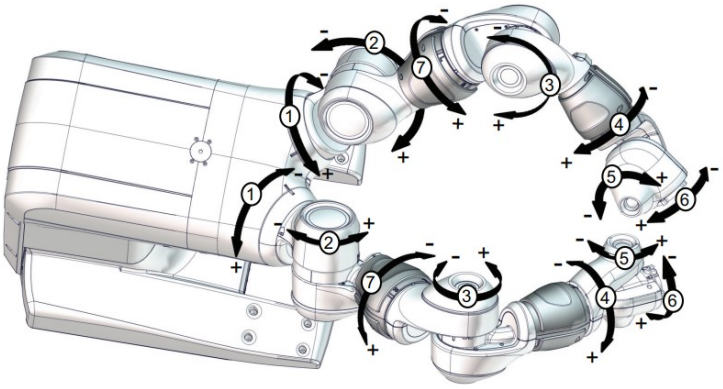
Konvenční průmyslové roboty se využívají zejména také tam, kde vyžadujeme vysokou rychlost a maximální přesnost, kde jsou prováděny pro člověka nebezpečné procesy (lakovny, svařovny apod.) a kde je potřeba manipulovat s těžkými, rozměrnými nebo horkými břemeny. Koboty využíváme zejména v provozech, které nelze kompletně automatizovat pro operace prováděné souběžně s manuálními pracemi. A také tam, kde nevyžadujeme nejvyšší rychlost a chceme manipulovat s lehčími předměty.[1]

2.1.1 YuMi

Zde si popíšeme kolaborativního robota s označením YuMi - IRB 14000, kterého budeme v další části práce řídit programovatelným automatem. Robot je produktem firmy ABB. Tento robot disponuje dvěma pažemi a je určen převážně pro montáž drobných součástek. Označení YuMi, podle výrobce, znamená „You and Me", tedy „ty a já". Robot je tedy, jak už jeho označení napovídá, schopen spolupráce s člověkem.[4]

Obr. 2.3: Robot YuMi - IRB 14000 firmy ABB[4]

Tento robot disponuje dvěma flexibilními pažemi, které se ohýbají v sedmi osách. Je schopen provádět pohyby při montáži drobných součástek ve velmi omezeném prostoru s dosahem, který odpovídá lidským vlastnostem. Což je důležitá vlastnost pro zachování minimálního půdorysu ve výrobních prostorech.



Obr. 2.4: Znázornění sedmi pohyblivých kloubů robota[18]

Osa	Typ pohybu	Rozsah pohybu	Rychlost pohybu
1	Rameno - Rotace	-168,5° až +168,5°	180 °/s
2	Rameno - Ohyb	-143,5° až +43,5°	180 °/s
7	Rameno - Rotace	-168,5° až +168,5°	180 °/s
3	Rameno - Ohyb	-123,5° až +80°	180 °/s
4	Zápěstí - Rotace	-290° až +290°	400 °/s
5	Zápěstí - Ohyb	-88° až +138°	400 °/s
5	Nástroj - Rotace	-229° až +229°	400 °/s

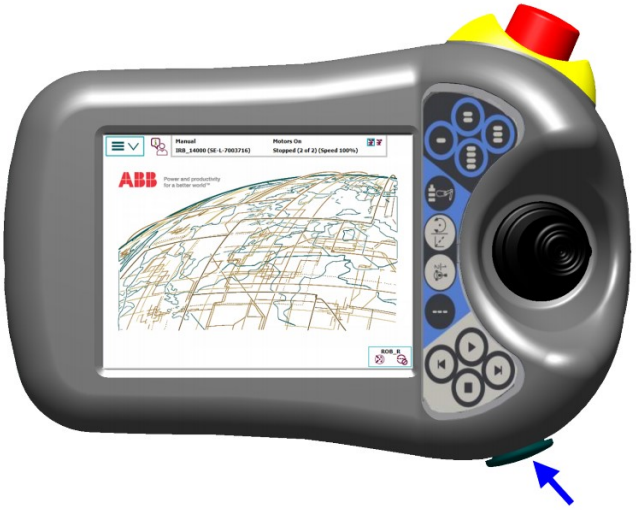
Tab. 2.1: Pracovní rozsahy a rychlosti pohybů os robota YuMi[18]

Paže jsou konstruovány z lehké a zároveň pevné hořčíkové slitiny, což umožňuje užitečné zatížení až 500 g. Slitina je pokryta pružným plastovým pláštěm zabaleným do měkkého polstrování. Tato konstrukce je účinná k absorpci síly v případě neočekávaného nárazu. V případě kontaktu dokáže svůj pohyb zastavit v řádu milisekund. Robot je schopen diagnostikovat změny v prostředí a v případě nutnosti zaznamenat přetížení. I přes všechny tyto bezpečnostní prvky je přesný a rychlý, opakovatelnost má s tolerancí 0,02 mm a disponuje maximální rychlostí okolo 1 500 mm/s. Hmotnost robota je pouhých 38 kg. Dá se tedy relativně snadno podle potřeby přenášet a přemisťovat. Je opatřen integrovaným systémem řízení (controller) a vnitřní kabeláží pro celou řadu vstupních a výstupních signálů včetně pneumatických a digitálních.

K robotu YuMi firma nabízí různé variace integrovaných ruk (efektorů), které je možné dle požadavků nakonfigurovat ze servo chapadel (s citlivou zpětnou vazbou regulace síly), zdvojených přísavek a kamerového vidění.[5]

Obr. 2.5: Možné konfigurace ruky pro robota YuMi vyráběné firmou ABB[4]

K ovládání robota na pracovišti využíváme ovládací panel, kterému se říká FlexPendant a připojujeme ho ke kontroléru robota (ovládací panel je dodávaný společně s robotem).[5, 6]

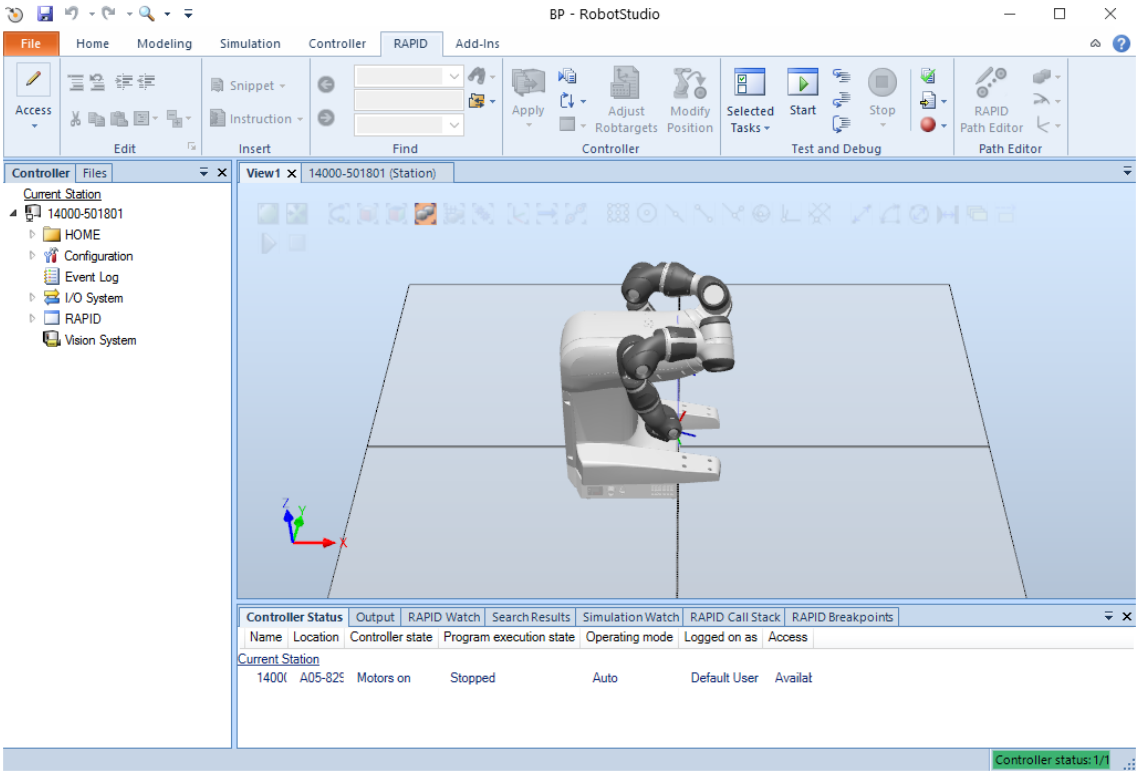


Obr. 2.6: Ovládací panel robota YuMi - FlexPendant[18]

2.1.2 RobotStudio, RobotWare a programovací jazyk RAPID

Pro programování robota YuMi využíváme v základní verzi volně dostupný software RobotStudio společnosti ABB (za příplatek si můžeme např. spouštět simulace a většinu offline funkcí). Robota YuMi je možné programovat i pomocí učení, kdy robot zaznamenává řadu pohybů a jejich souřadnic. Na připojeném tabletu ve spuštěné

aplikaci YuMi můžeme přímo vidět, jak software okamžitě mění tyto pohyby do základního kódu, kterým se YuMi ovládá. Tímto způsobem se však dají vytvářet pouze jednoduché aplikace, ty složitější už musíme vytvořit (napsat) v RobotStudiu.[5, 6]



Obr. 2.7: Prostředí softwaru RobotStudio

RobotStudio umožňuje programování, učení a optimalizaci robotů na PC. To vše je možné provádět přímo na pracovišti (tzv. online režimu), kde k podobnému účelu slouží i FlexPendant, ale i mimo pracoviště (tzv. offline režim) např. v kanceláři bez narušení výroby. Tento software se tedy používá k modelování, programování a simulaci robotických buněk. Umožňuje pracovat s offline kontrolérem (virtuálním IRC5 kontrolérem - VC), který je spuštěn lokálně na PC. RobotStudio využívá přesnou kopii skutečného softwaru (RobotWare), který je spuštěn na robotu i během výroby. Pohyby robota namodelované a odzkoušené pomocí simulace na virtuální řídící jednotce jsou přímo převáděny do kódu. Takto vytvořený program lze poté jednoduše nahrát do kontroléru robota na pracovišti.[7]

Řídící software RobotWare je základem všech řídících jednotek od společnosti ABB. Poskytuje vysokou přizpůsobivost, krátké doby cyklu a vysokou přesnost, vynikající spolehlivost a bezpečnost. Disponuje rozsáhlou komunikační schopností pro integraci a řízení. Taktéž má vestavěnou funkci řízení procesů. Mezi základní možnosti softwaru patří řízení koordinace pohybu, události pohybu a vstupů výstupů.

Dále nabízí obsáhlou škálu funkcí jako je například ABB technologie pohybu obsahující TrueMove (zajištující přesné pohyby po naprogramované trajektorii bez ohledu na rychlost robota) a QuickMove (pro optimalizaci pohybu, zajišťováním maximálního zrychlení v každém okamžiku). Nejzákladnější vlastností tohoto softwaru je programovací jazyk robota s názvem RAPID.[8]

Programovací jazyk RAPID, řadící se mezi vyšší programovací jazyky, ve kterém je vytvářen kód i v RobotStudiu, je vysoce flexibilní programovací nástroj a jeden z nejvýkonnějších softwarových jazyků v robotickém průmyslu. Skládá se z mnoha specifických instrukcí s argumenty popisující práci robota (např. instrukce pro pohyb s argumenty nastavujícími rychlost, způsob nebo přesnost pohybu). Existují zde tří základní typy rutin: procedury (PROC), funkce (FUNC) a trap (TRAP) rutiny. Procedury slouží jako podprogramy. Funkce mají návratovou hodnotu konkrétního typu a mohou obsahovat cykly nebo podmínky. Trap rutina je určena k zachytávání přerušení a vykonávání obslužného kódu příslušného přerušení. Data ve kterých jsou uložené nějaké informace lze rozdělit na tři typy: konstantní (constant - CONS), proměnné (variable -VAR) a perzistentní (persistent - PERS). Konstantní jsou taková data kterým nelze během běhu programu měnit hodnotu, hodnota jim je přiřazena při programování. Proměnné mohou svoji hodnotu měnit kdykoliv při běhu programu. Persistentní data lze libovolně měnit stejně jako obyčejná proměnná s tím rozdílem, že takováto proměnná si při každé změně zachová i svoji předchozí hodnotu. Tento jazyk má i mnoho dalších vlastnosti, podobných jiným programovacím jazykům, jako jsou například rutinní parametry, aritmetické a logické výrazy, multitasking a mnohé další.[9]

```
MODULE MainModule

    CONST num x := 5;
    VAR num y;
    VAR num z;

    PROC main()
        y := 2;
        z := x * y;
        TPWrite "The variable z is :" \Num := z;
    ENDPROC

ENDMODULE
```

Obr. 2.8: Ukázka jednoduchého programu napsaném v jazyce RAPID

V této práci je používáno RobotStudio převážně v online režimu. Budeme s jeho pomocí nastavovat základní HW konfiguraci controlleru a vytvářet program pro robota pomocí programovacího jazyka RAPID.

2.2 Programovatelný automat (PLC)

Programovatelný automat neboli PLC (Programmable Logic Controller) je nedílnou součástí automatizace, jedná se o řídicí systém přizpůsobený pro řízení průmyslových a technologických procesů v podmínkách průmyslové výroby(odolný proti rázům, prachu, výkyvům teplot, atd.).[10]

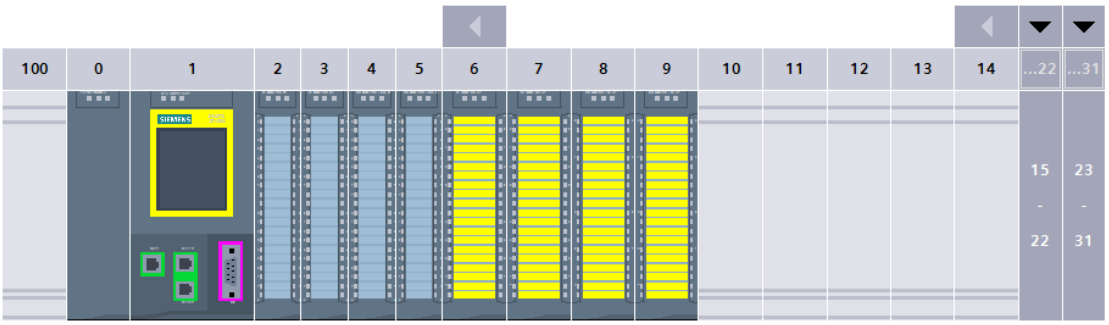
2.2.1 SIMATIC S7-1500

Řídicí systém Simatic S7-1500 je aktuálně nejnovějším produktem z řad řídících systémů společnosti SIEMENS. Jedná se o modulární systém PLC. Tento systém je určen k řízení strojů střední velikosti i náročných aplikací, které vyžadují maximální výkonnost a spolehlivou komunikaci. Oproti předchozím modelům PLC disponuje vyšším výkonem a efektivitou. Co se týče celkového výkonu, jsou výrazně rozšířeny technologické funkce a zdokonaleny funkce zabezpečení a ochrany dat, lepší integrace funkční bezpečnosti a výkon jádra systému. Efektivita byla zvýšena zejména v oblasti konstrukce jako je ovládání a diagnostiky systému a jeho začlenění do vývojového prostředí TIA Portal.[12]

Obr. 2.9: PLC S7-1500 (tři základní možnosti provedení)[11]

Vysoký výkon umožňuje dosáhnout rychlého zpracování signálů, mimořádně krátkých dob odezvy a vysoké kvality řízení. Řídící systém S7-1500 nabízí několik rozdílných verzí CPU, počínaje standardním CPU s možností integrace C/C++ (s označením S7-1500C). U kompaktních CPU (s C/C++) s digitálními a analogovými I/O

jsou integrovány čítačové a pulsní vstupy, které mohou být přímo zaznamenávány na CPU. Standardní a bezpečnostní (safety) programy mohou běžet na stejném kontroléru s použitím fail-safe CPU (s označením S7-1500F). Tyto CPU mohou být rozšířeny o funkci pro řízení pohonů (Motion Control). Takto vybavené kontroléry jsou označeny jako S7-1500T případně S7-1500TF (s integrovaným safety).[13] Jednotlivé řady CPU se liší především v rychlosti zpracování informace a velikosti dostupné paměti. SIMATIC S7-1516F-3 PN/DP - Řídící systém, který budeme v této práci používat, má rychlost zpracování bitové operace 10 ns a pracovní paměť 1.5 MB pro program a 5 MB pro data. Velikost datového bloku je 5 MB. Jedná se o CPU s integrovaným safety.[14]



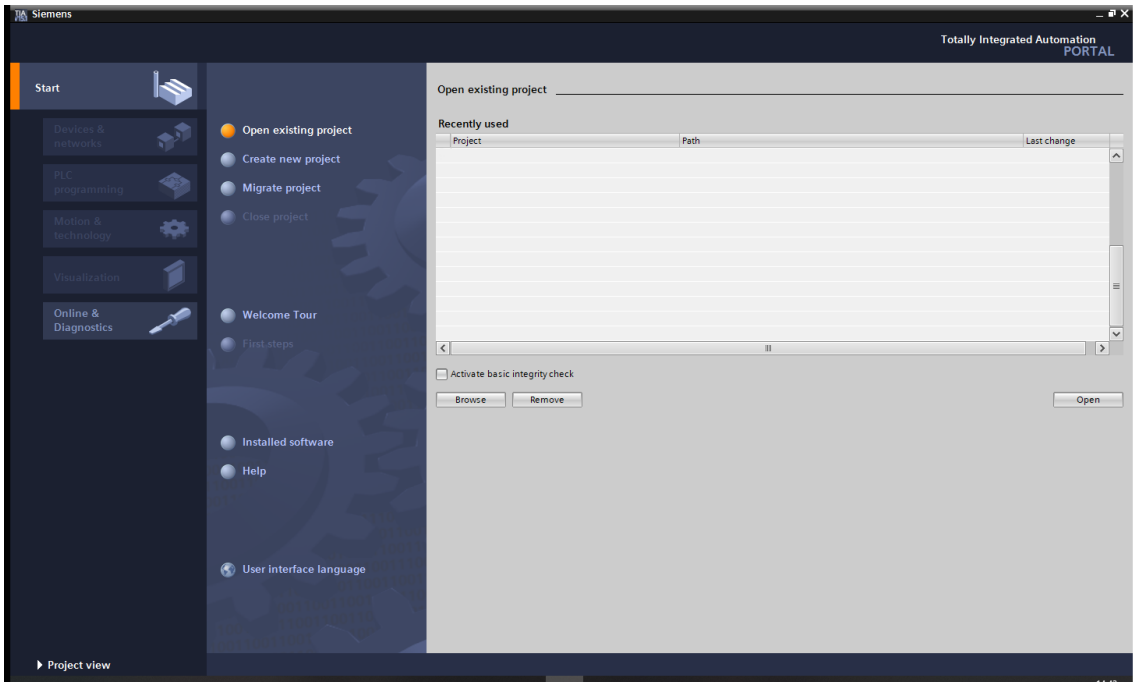
Obr. 2.10: Zobrazení PLC SIMATIC S7-1516F-3 PN/DP v prostředí TIA Portal

2.2.2 TIA Portal

Totally Integrated Automation Portal neboli TIA Portal, je software pro programovaní, konfiguraci a diagnostiku řídících systémů společnosti SIEMENS. Nespornou výhodou tohoto vývojového prostředí (oproti vývojovým prostředím od jiných společností) je, že obsahuje společné prostředí pro vývoj aplikačních programů pro řídicí systémy tvořené programovatelnými automaty (PLC), decentralizovanými periferiemi (stanicemi I/O) a pro vývoj operátorských rozhraní pro stroje a zařízení s použitím operátorských panelů (HMI) i pro dispečerské systémy (kategorie SCADA).[15]

2.2.3 Program v prostředí TIA Portal

Vývojové prostředí TIA Portal disponuje všemi základními programovacími jazyky dle mezinárodní normy IEC 61 131-3, která sjednocuje programovací jazyky a jejich syntaxi a definuje společné prvky pro všechna vývojová prostředí od různých výrobců. To však neznamená, že programy psané v jiných vývojových prostředích jsou stejné. Každý z výrobců si jazyky dále modifikuje, jsou si sice podobné, protože

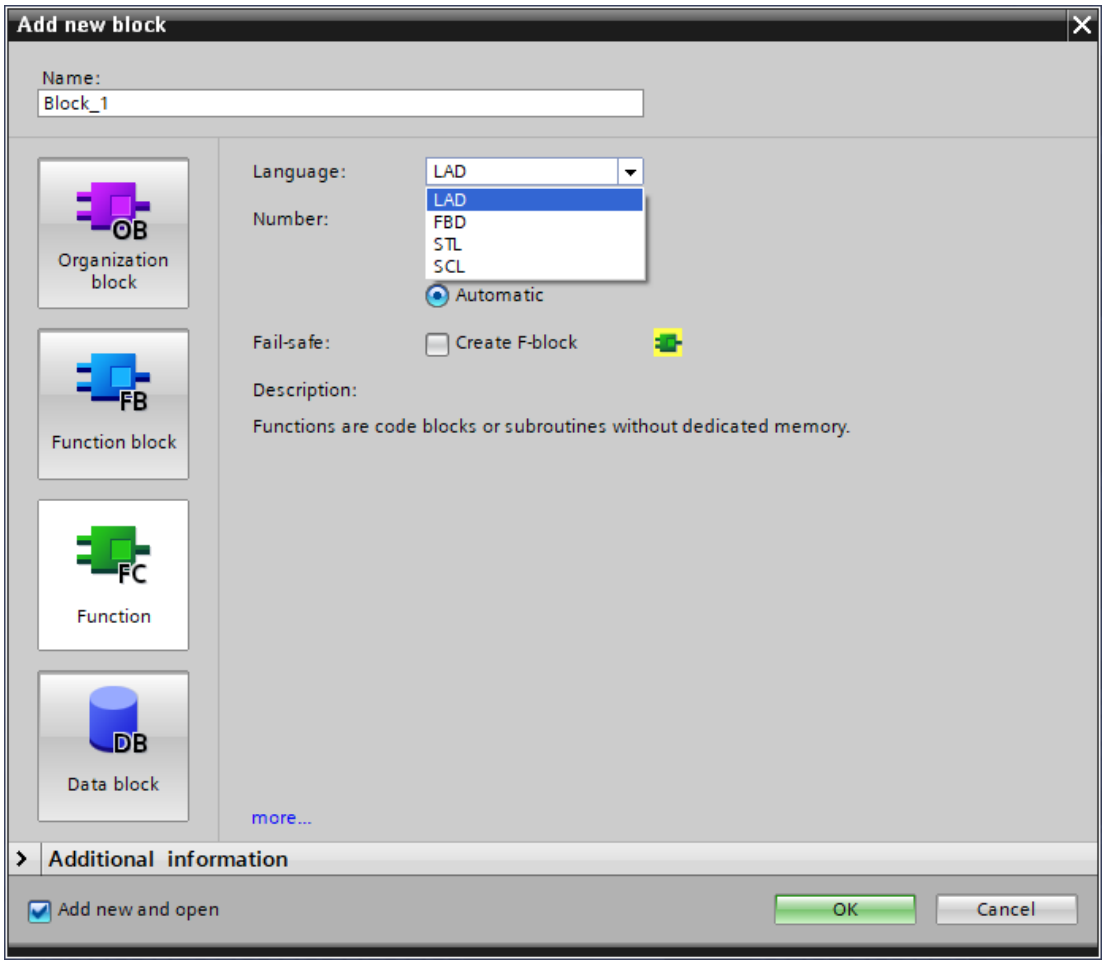


Obr. 2.11: Základní obrazovka programovacího prostředí TIA Portal

mají společný základ, ale ne stejné. To má za následek nepřenositelnost programu napsaném v prostředí TIA Portal do jiných vývojových prostředí.

TIA Portal disponuje programovacími jazyky LAD, FBD, STL, SCL a GRAPH. Jazyk LAD odpovídá normovanému jazyku LD (Ladder Diagram), je založen na principu reléové logiky a řadí se tedy mezi grafické programovací jazyky. Program je tvořen jednotlivými příčkami na které jsou přidávány prvky z instrukční sady, jako například spínací a rozpínací prvky nebo funkce a funkční bloky. Jazyk FBD (Function Block Diagram) je dalším z grafických programů. Program je tvořen pomocí funkčních bloků, které jsou propojovány a kombinovány (např. sériově nebo paralelně). Funkčními bloky jsou většinou logické případně aritmetické operace nebo funkce. STL je založen na normovaném jazyku IL (Instruction List), skládá se ze seznamu instrukcí, které se vždy píší na samostatný řádek. Instrukce je tvořena z operátoru případně modifikátoru a jednoho nebo více operandů. Při rozsáhlejších programech se tento jazyk stává značně nepřehledným. Jazyk SCL vychází z normovaného jazyka ST (Structured Text), který se řadí mezi textové programovací jazyky. Jedná se o strukturovaný jazyk, podobný syntaxi vyšších programovacích jazyků. Zápis programu se provádí ve formě strukturovaného textu s použitím metod a způsobů vyššího programovacího jazyka včetně deklarací, podmínek, knihoven a dalších. GRAPH odpovídá normovanému jazyku SFC (Sequential Function Chart) a jedná se o další grafický programovací jazyk. Tento jazyk je odvozený na základě algoritmů Petriho sítí, tedy využívá konceptů stavů a přechodů mezi nimi.

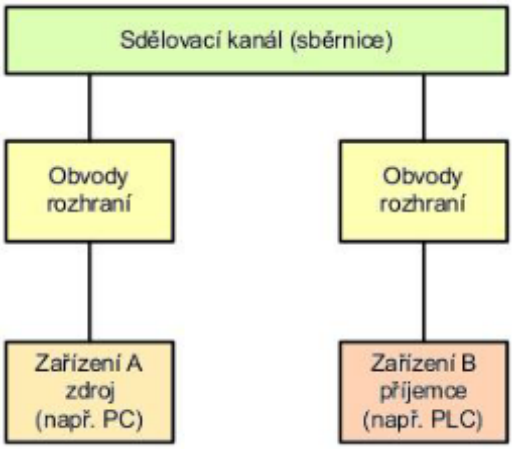
Program v prostředí TIA Portal je koncipován do bloků, které jsou dále děleny podle jejich funkcí. Základním blokem programu je blok OB neboli organizační blok. Tím nedůležitějším z organizačních bloků je blok Main(OB1), který je vykonáván cyklicky a jsou v něm volány a následně zpracovávány bloky tvořící program pro PLC. Dalším takovým blokem je například Startup(OB100), který je vykonán pouze jednou a to po restartu PLC. Dále je tu blok DB neboli datový blok, ten slouží k ukládání dat programu. Blok FB neboli funkční blok je používán k tvorbě podprogramů (funkcí). Při použití FB v programu, je vytvořena instance tohoto funkčního bloku, které je přidělen i datový blok. Instance funkčního bloku je vykonávána pouze tehdy je-li volána a její zpracovaná data a parametry jsou ukládány do datového bloku, kde zůstávají uloženy i po vykonání daného FB. Podobným blokem je blok FC neboli Funkce, jehož jediným rozdílem oproti FB je, že po ukončení volání instance FC svá data nikde neukládá (pro instanci FC není vytvořen DB) a tedy jsou tyto data nenávratně ztracena.[16]



Obr. 2.12: Vytváření nového bloku programu v prostředí TIA Portal

3 Komunikační rozhraní mezi robotem a PLC

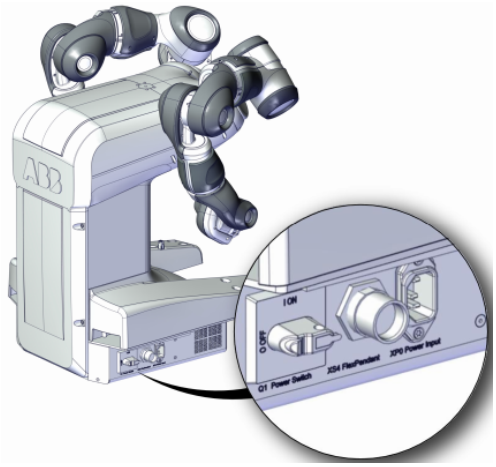
Komunikační rozhraní slouží pro komunikaci mezi dvěma čí více zařízeními. Skládá se ze zdroje a příjemce zprávy, obvodů pro úpravu signálu a sdělovacího kanálu. Zdroj a příjemce mohou zprávu generovat i přijímat. Obvody pro úpravu signálu upravují zprávu na signál vhodný pro přenos přes sdělovací kanál. Sdělovací kanál slouží pro přenos zprávy od zdroje k příjemci (může být bezdrátový nebo kabelový).[17]



Obr. 3.1: Komunikační rozhraní[17]

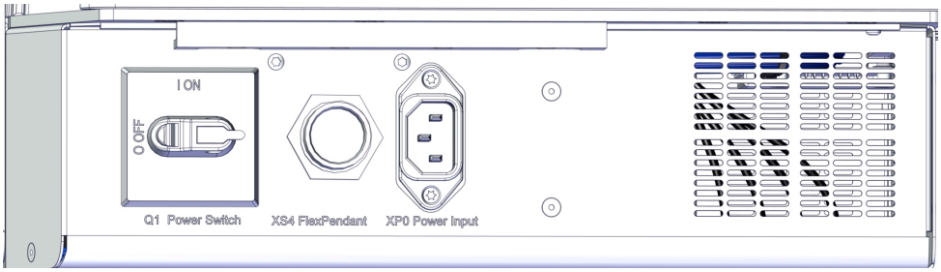
3.1 Komunikační rozhraní robota YuMi

Veškerá komunikační rozhraní robota se nachází na integrovaném kontroléru ve spodní části robota.



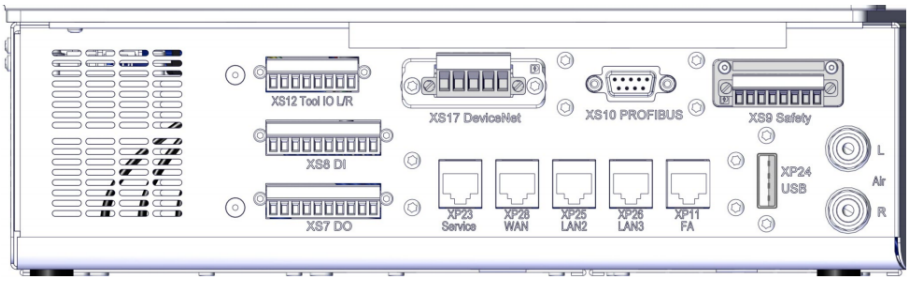
Obr. 3.2: Umístění kontroléru na robotu[18]

Na pravé straně kontroléru se nachází komunikační rozhraní pro připojení ovládací konzole FlexPendant, taktéž napájení a hlavní vypínač.



Obr. 3.3: Pravá strana kontroléru robota[18]

Na levé straně kontroléru se nachází většina komunikačních rozhraní robota (Obr. 3.4). Prvním rozhraním s označením XS12 je 4+4 digitálních I/O pro ovládání nástrojů robota (mohou být propojeny s XS8 nebo XS9). S označením XS17 je rozhraní pro připojení sběrnice DeviceNet a XS10 pro sběrnici PROFIBUS DP. XS9 je pro připojení bezpečnostních signálů (safety). Připojení XS8 a XS7 je určeno pro 8 digitálních vstupů a 8 digitálních výstupů. Servisní vstup XP23 slouží pro prvotní připojení a nastavení robota, poté následují v řadě připojení pro WAN (XP28), dvakrát LAN (XP25 a XP26) a pro PROFINET IO nebo EtherNet/IP (XP11). Kontrolér má také USB port a v poslední řadě dva vzduchové vstupy pro ovládání pneumatických nástrojů na rukách robota.[18]



Obr. 3.4: Levá strana kontroléru robota[18]

3.2 Komunikační rozhraní PLC S7-1516F-3 PN/DP

Samotný modul PLC pouze s CPU má integrované připojení třikrát pro PROFINET (dvakrát IRT a jednou RT) a jedno připojení pro PROFIBUS. S modulárním rozšířením samozřejmě přibývají další možnosti připojení, například rozšíření základním I/O digitálním či analogovým modulem. PLC má také vstup pro paměťovou kartu, která slouží jako paměť programu.

3.3 Možnosti komunikačního rozhraní robot PLC

Robota je možné řídit pomocí PLC několika způsoby. Robota s PLC můžeme propojit buď díky rozšiřujícímu modulu s digitálními nebo analogovými I/O a nebo jednou ze dvou průmyslových sběrnic (PROFIBUS nebo PROFINET).

3.3.1 Digitální a analogový I/O

Robota lze řídit pomocí PLC s použitím rozšiřujících analogových a digitálních modulů. Tato možnost řízení je však značně omezená, zatím co PLC můžeme rozšířit téměř libovolně, rozhraní robota nám dovoluje připojit pouze 4+4 digitálních I/O pro ovládání nástrojů robota a 8 digitálních vstupů a 8 digitálních výstupu pro řízení robota.

Obr. 3.5: Rozšiřovací modul pro S7-1500 (pro 16 digitálních vstupů)[19]

3.3.2 PROFIBUS

Komunikaci robota s PLC můžeme zajistit průmyslovou sběrnicí PROFIBUS, neboli přesněji pro náš účel PROFIBUS DP určený pro oblast řízení strojů. PROFIBUS je standardně tvořen dvoužilovým stíněným kabelem, případně optickým kabelem (existují i bezdrátové varianty), fialové barvy izolační vrstvy. Jedná se o klasickou sériovou sběrnici. Dosahuje maximální přenosové rychlosti 12 Mbit/s (při délce jednoho segmentu sběrnice do 100 m). Jednotlivá zařízení (stanice, uzly, účastníci komunikace) jsou ke sběrnici připojena paralelně (v liniové topologii) v počtu do 32 zařízení (včetně opakovačů) na segment. V síti může být více segmentů, avšak maximální počet zařízení je 127. Sběrnici je nutné opatřit zakončovacím odporem (terminátorem). Každá stanice má pevně přidělenou adresu, podle které je na sběrnici rozpoznávána. [20]

Obr. 3.6: Znázornění kabelu PROFIBUS[21]

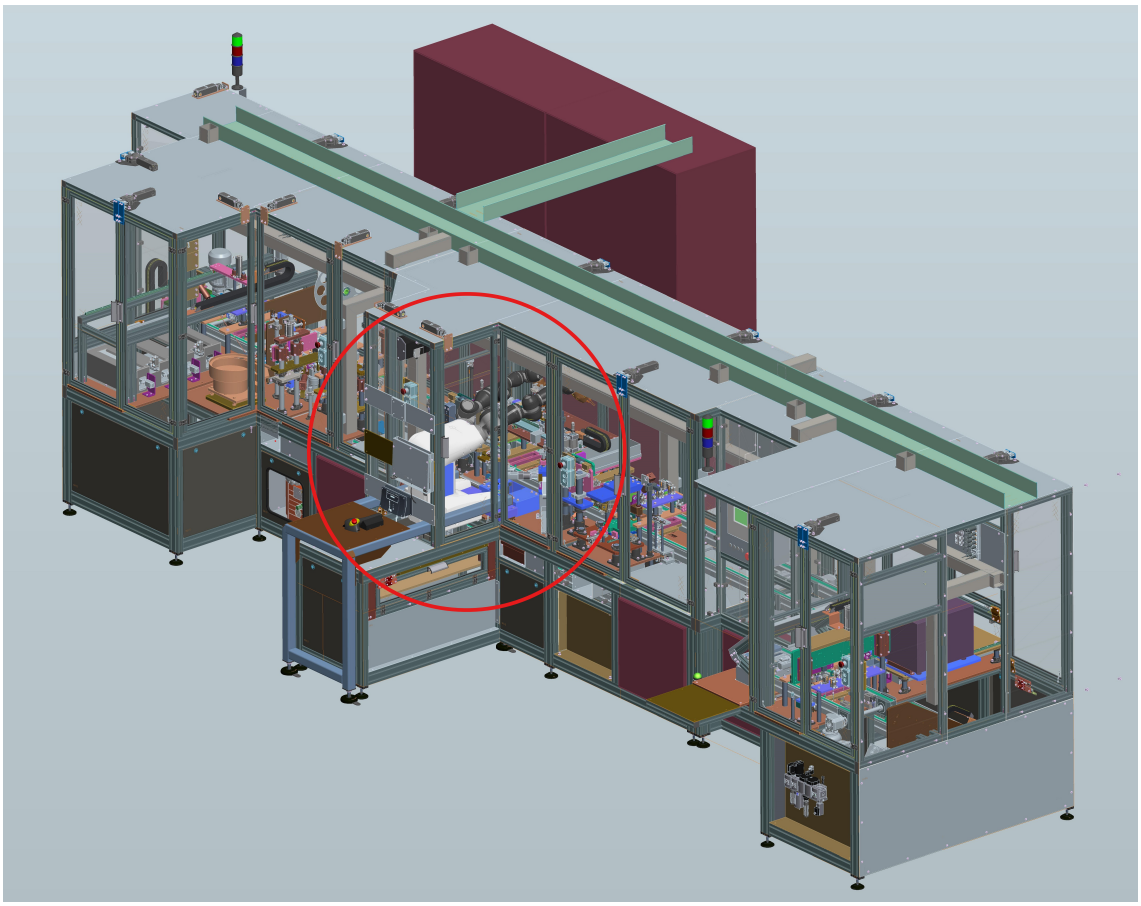
3.3.3 PROFINET

Propojení zajišťující komunikaci mezi robotem a PLC muže být tvořeno průmyslovou sběrnicí PROFINET, tj. komunikační standard pro všechny úrovně průmyslové automatizace založený na průmyslovém Ethernetu. Jedná se v podstatě o modernějšího nástupce průmyslové sběrnice PROFIBUS, který je touto sběrnicí snadno nahraditelný bez nutnosti modifikace změny existujících zařízení. Poznáme jej většinou podle zelené barvy kabelu. Protože protokol Profinet využívá fyzickou vrstvu Ethernetu, nabízí vedle jednoduššího propojení s nadřazenými systémy také možnost realizovat síťové topologie. Standardní rychlost této sběrnice je 100Mbit/s, případně 1Gbit/s. PROFINET nabízí i možnost komunikace v reálném čase, např. pro cyklická uživatelská data nebo událostmi řízená přerušení, s časovou odezvou 5-10 ms. Propojovací kabel mezi dvěma jednotlivými stanicemi může mít délku až 100 m a počet připojených stanic je teoreticky bez omezení. Propojení stanic je v podstatě libovolné, zařízení adresujeme pomocí IP adres. Adresní prostor je téměř neomezený, tedy dokud nedojdou volné IP adresy. Každému účastníkovi (zařízení, stanici) komunikace je tedy kromě unikátní adresy MAC přidělena IP adresa (statická či dynamická) a uživatelem daný název.[20, 22]

Obr. 3.7: Znázornění kabelu PROFINET[23]

4 Návrh demonstrující úlohy

Tato úloha demonstruje reálnou aplikaci pro koncového zákazníka, který se zabývá výrobou roznětných systémů. Jedná se o výrobní linku pro výrobu elektrických rozbušek. Už samotné to, že se jedná o produkt v nesprávných rukou člověku nebezpečný, může být jedním z důvodů proč při výrobě použít robota. Robot se nachází přibližně ve středu výrobní linky (viz Obr. 4.1), jeho úkolem je obsluha dvou stanic výrobní linky.



Obr. 4.1: Náhled výrobní linky + zvýrazněné umístění robota

Vyráběný produkt se skládá ze dvou vodičů (o průměru cca 0,5 mm), dvou malých rezistorů (o rozměrech cca 4x1 mm s vývody o průměru cca 0,3 mm) a malé kapsle s třaskavinou (o rozměrech cca 4x5 mm).

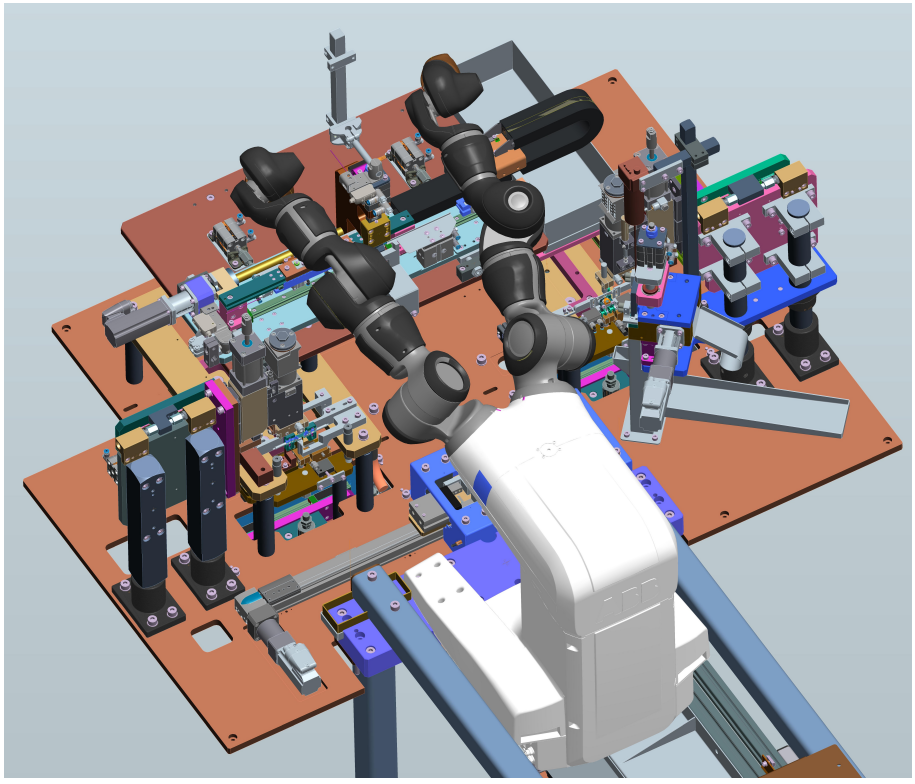
Do první stanice obsluhované pravou rukou robota přijíždí paletka s připraveným odizolovanými vodiči. Robot má za úkol uchopit rezistor a srovnat jeho pozici ve střižníku. Poté se s rezistorem se zkrácenými vývody přesunout na pozici operace sváření vodiče s rezistorem. Jakmile jsou na koncích obou vodičů přivařeny rezistory, paletka s vodiči se přesouvá do druhé stanice, obsluhované levou rukou robota.

Zde má robot za úkol zajistit přesun hřebenů s kapslemi ze zásobníku do střižníku umístěným před robotem. Dále zajistit odběr a následný přesun jedné ustřižené kapsle ze střižníku na pozici operace sváření. Kde jsou vývody kapsle přivařeny k vývodům rezistorů. Takto zpracovaný výrobek je připraven k přesunu do dalších stanic výrobní linky, kde bude podroben kamerové kontrole kvality a následným zapouzdřením výrobku.

Požadavek na dobu cyklu jedné paletky, od příjezdu na první stanici až po odjezd z druhé stanice (včetně přejezdu mezi nimi), je dosáhnout na hodnotu 9 sekund. Přesun paletky mezi stanicemi trvá přibližně 1 sekundu, tedy čistý čas pro operaci robota je 8 sekund (ale čas přejezdu paletky může být využit na přípravu).

Dalším kritickým parametrem pro realizaci úlohy robota je přesnost. Je nutné zajistit dostatečnou kvalitu sváru, aby výsledný produkt odpovídal požadavkům zákazníka. Vzhledem k tomu, že svařujeme drátky okolo 0,5 mm, bude požadovaná přesnost přibližně na hranici přesnosti robota YuMi (tj. 0,02 mm).

4.1 Popis úkolu robota



Obr. 4.2: Pohled na pracoviště robota

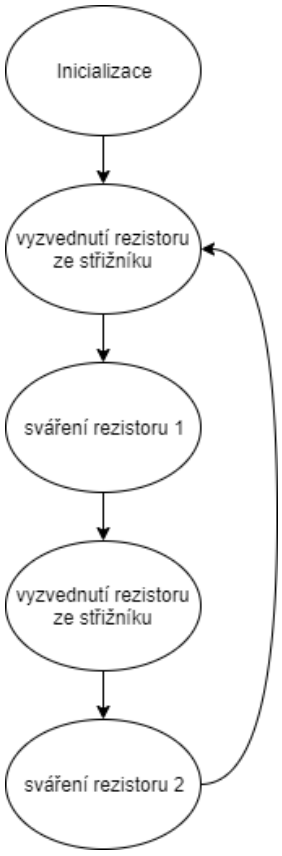
Proč právě robot YuMi? Protože robot musí na stanicích výrobní linky pracovat ve velice omezeném prostoru (Obr. 4.2), a proto je obrovskou výhodou flexibilita, podobná lidským rukám, které se robot Yumi velice přibližuje. Taktéž využijeme další vlastnosti robota, kterými jsou přesnost, rychlost a opakovatelnost, neboť předměty se kterými bude robot manipulovat na stanicích dosahují skutečně malých rozměrů.

4.1.1 Pravá strana robota

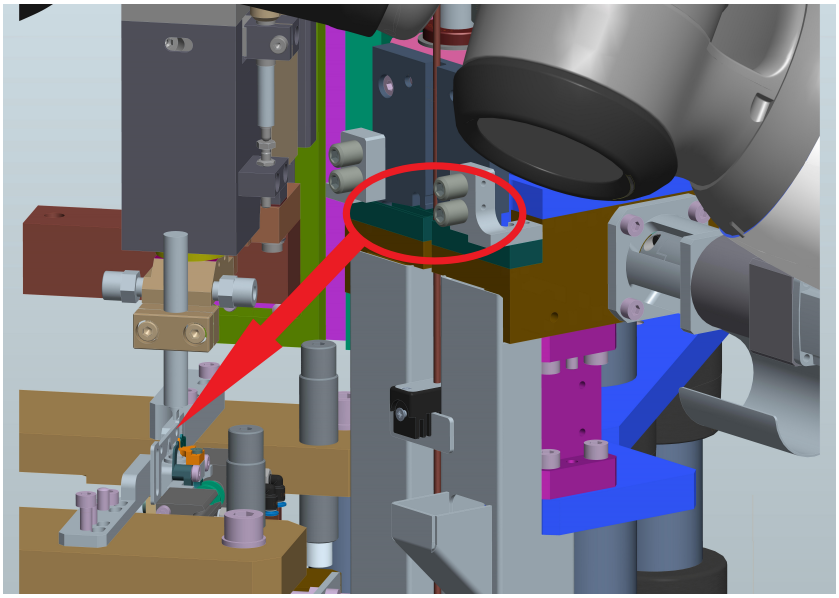
Na stanici po pravé straně robota provádíme sváření vodičů s rezistory. Nástroj pravého ramene robota je konstrukčně přizpůsoben pro uchopení malých rezistorů.

Úplně prvním krokem při spuštění nebo inicializaci robota je vždy očištění nástroje od případných nečistot nebo zbytků z posledních úkonů robota. K tomuto účelu slouží menší kartáč, umístěný pod střižníkem, kterým nástroj párkrát projede. Až poté je pravé rameno připraveno vykonávat výrobní cyklus.

Rezistory jsou postupně přiváděny do střižníku (zvýrazněno na Obr. 4.4), kde čekají dokud je neuchopí robot. Jelikož nelze zaručit, že se odpor do střižníku dostane vždy na stejnou pozici, je nutné jeho pozici kontrolovat kamerou. Kamera přes PLC do robota odešle aktuální informaci o poloze rezistoru. Robot je tak schopen rezistor uchopit vždy na stejném místě. Po uchopení rezistoru se s ním přesune na střed střižníku tak, aby se odstřihly přebytečně dlouhé vývody rezistoru na cca 2 mm z obou stran. S připraveným rezistorem se musí přesně přiblížit k prvnímu ze dvou vodičů, které čekají na paletce pod svářečkou. Po přivaření rezistoru se cyklus opakuje pro druhý vodič. Teprve poté se paletka přesouvá do další stanice a na stejné místo přijíždí nová paletka čekající na přivaření rezistorů.



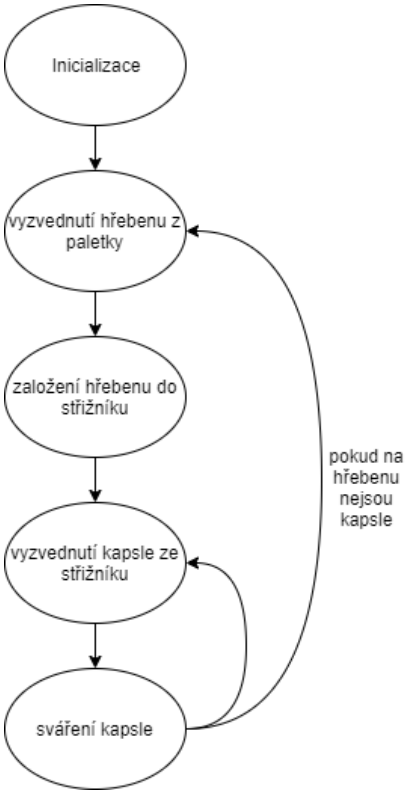
Obr. 4.3: Základní cyklus pravé strany robota



Obr. 4.4: Pohled na stanici na pravé straně robota (vyznačen střižník a pohyb směrem ke svářečce)

4.1.2 Levá strana robota

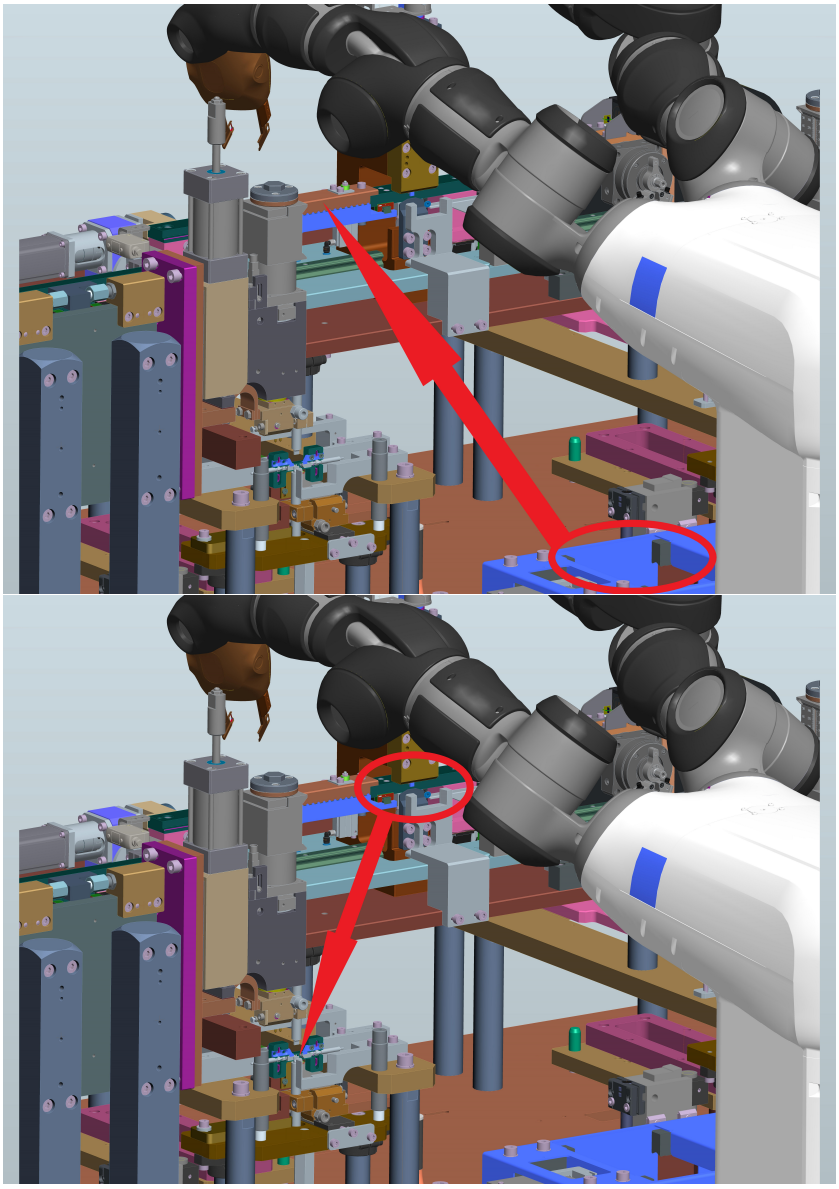
Na stanici po levé straně robota provádíme sváření vývodů kapsle s třaskavinou a vývodů rezistorů navařených v předchozí stanici na dva vodiče. Nástroj levého ramene je taktéž konstrukčně přizpůsoben pro uchopení hřebenu s kapslemi a zároveň i pro uchopení jedné samostatné kapsle.
Stejně jako u pravého ramene robota je nutné před spuštěním cyklu zajistit čistotu nástroje, to je zajištěno kartáčem po levé straně robota, kde projetím přes kartáč očistíme nečistoty z nástroje.
Dolů před robota je přiváděna paletka s 10 hřebeny srovnanými v řadě za sebou a na každém hřebenu je 21 kapslí. Robot vždy vyzvedne jeden hřeben s kapslemi a vloží jej do střižníku.
Ve střižníku se hřeben posouvá tak, aby se odstřihla vždy pouze jedna kapsle.



Obr. 4.5: Základní cyklus levé strany robota

Robot musí před střižením kapsli uchopit, po odstřižení se s ní přemístit přesně ke svářečce tak, aby se vývody rezistorů a kapsle dotýkaly a čekat na přivaření.

Tento cyklus robot opakuje dokud jsou na hřebeni ve střižníku kapsle. Pokud kapsle dojdou, musí robot vyzvednout další hřeben s kapslemi v řadě a vložit jej do střižníku, poté se zase vše opakuje v cyklu.



Obr. 4.6: Pohled na stanici po levé straně robota (vyznačeny pohyby hřebenu s kapslemi do střižníku a následný pohyb kapsle ke svářečce

5 Návrh komunikačního rozhraní

Mezi základní požadavky na komunikační rozhraní patří zejména spolehlivost a přenosová rychlost. Dalším kritériem při výběru komunikačního kanálu může být dostupnost, cena, jednoduchost instalace a nebo zákazníkem preferovaný dodavatel. U komunikačního rozhraní jistě oceníme jistou flexibilitu (popř. standardizaci), zejména co se týká možnosti připojení dalších periferních zařízení na stejný komunikační kanál. Také v případě inovace či poruchy a případné výměny stávajícího komunikačního zařízení je velmi často nutné zajistit rychlou reakci. Je důležité, aby nové zařízení podporovalo stávající komunikační rozhraní a nebylo nutné měnit celou architekturu komunikačních linek.

5.1 Požadavky na komunikační rozhraní mezi robotem a PLC

Požadavky na komunikační rozhraní mezi robotem a PLC vyplývají ze zadání úlohy robota. V první řadě je nutné definovat jaký typ dat si bude robot s PLC předávat. Robot by měl být schopen přijímat binární i analogová data.

- Vstupními daty pro robota budou binární data typu: povolení příjezdu nebo odjezdu robota (pro každý cyklus robota zvlášť); ovládání chapadel; systémové informace (start/stop, zapnutí/vypnutí motorů atd.). A analogová data typu: číslo kroku; korekce pozice z kamery nebo z OP; rychlost robota.
- Naopak výstupními daty z robota budou binární data typu: systémové informace (je robot zapnut, jsou motory spuštěny, detekce kolize); informaci ve které oblasti se robot právě nachází. A také analogová data typu: rychlost robota; aktuální krok cyklu robota, aktuální pozice otevření chapadel.

Dále je nutné zajistit přenosovou rychlost komunikace, aby mohla být garantována včasná reakce požadavků na řízení z PLC nebo robota. Zde jsme limitováni pouze parametry komunikace kontroléru robota a CPU PLC.

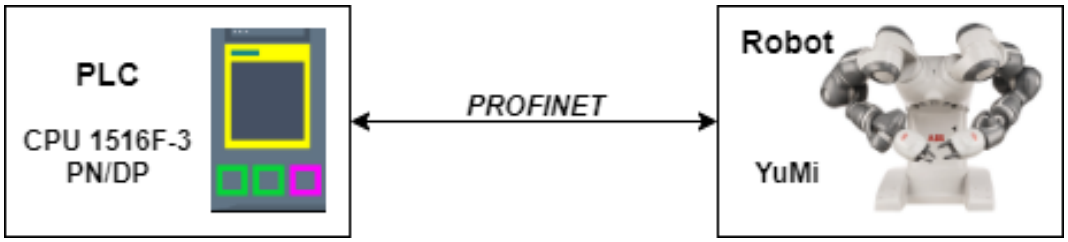
5.2 Řešení

Při výběru komunikačního rozhraní se musí brát ohled na to, kde se dané zařízení, v našem případě robot YuMi, bude nacházet, jaké možnosti již máme zavedeny a kolik a jaká data potřebujeme tímto rozhraním předávat. Pokud ke komunikaci potřebujeme analogová data v takovém rozsahu, nelze použít binární komunikaci.

S ohledem na množství dat, která budeme chtít pro náš úkol řízení robota pomocí PLC předávat nebo naopak získávat, můžeme vyloučit komunikaci pomocí digitálních či analogových signálů, neboť robot má v tomto směru velice omezené komunikační možnosti.

Dalším možným komunikačním rozhraním, které vyloučíme, je PROFIBUS. Teoreticky by tato průmyslová sběrnice byla svou rychlostí pro přenos dat postačující, ale s ohledem na možnost instalace na výrobní lince, kde náš robot pracuje, by byla její instalace komplikovaná.

Po celé lince je instalována síť PROFINET, která zajišťuje komunikaci instalovaných snímačů a distribuovaných periferií s řídícím PLC. Jediné zařízení, které komunikuje přes průmyslovou sběrnici PROFIBUS, je svářečka, která nemá jiné možnosti komunikace. Tedy vzhledem k možnostem instalace a množství přenášených dat, které si chceme s robotem vyměňovat, je průmyslová sběrnice PROFINET pro tento úkol nejlepším řešením.



Obr. 5.1: Návrh komunikačního rozhraní

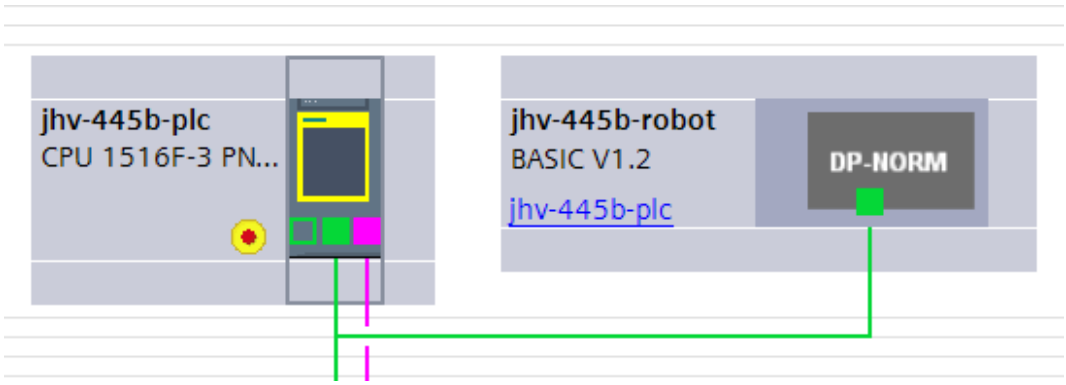
6 Implementace komunikačního rozhraní

Pouhé propojení PLC a robota sítí PROFINET však nestačí, je nutné každé zařízení správně nakonfigurovat. K tomu využijeme vývojová prostředí pro PLC a pro robota.

6.1 Hardwarová konfigurace PLC

Prvním krokem je vytvoření PLC projektu. Vzhledem k tomu, že se jedná o reálnou aplikaci pro koncového zákazníka a v době kdy je tato práce zpracovávána existuje projekt s rozpracovaným programem pro PLC. Následující úpravy jsou prováděny v tomto existujícím projektu.

V TIA Portalu v záložce pro hardwarová nastavení již máme dané PLC a další instalované komponenty s definovaným propojením k PLC (PROFINET nebo PROFIBUS). Je také nutné vytvořit připojení našeho robota. Budeme k tomu potřebovat přidat do TIA Portalu GSDML soubor robota. GSDML soubor obsahuje obecná i pro robota specifická nastavení pro komunikaci a konfiguraci připojení sítí PROFINET. Tento soubor lze získat z disku robota, například pomocí softwaru Robot Studia z podadresáře aplikace RobotWare. Po přidání GSDML souboru, již stačí najít naše zařízení v Hardware katalogu (dostupném v TIA Portalu) a přidat jej a propojit s PLC viz. obrázek 6.1.

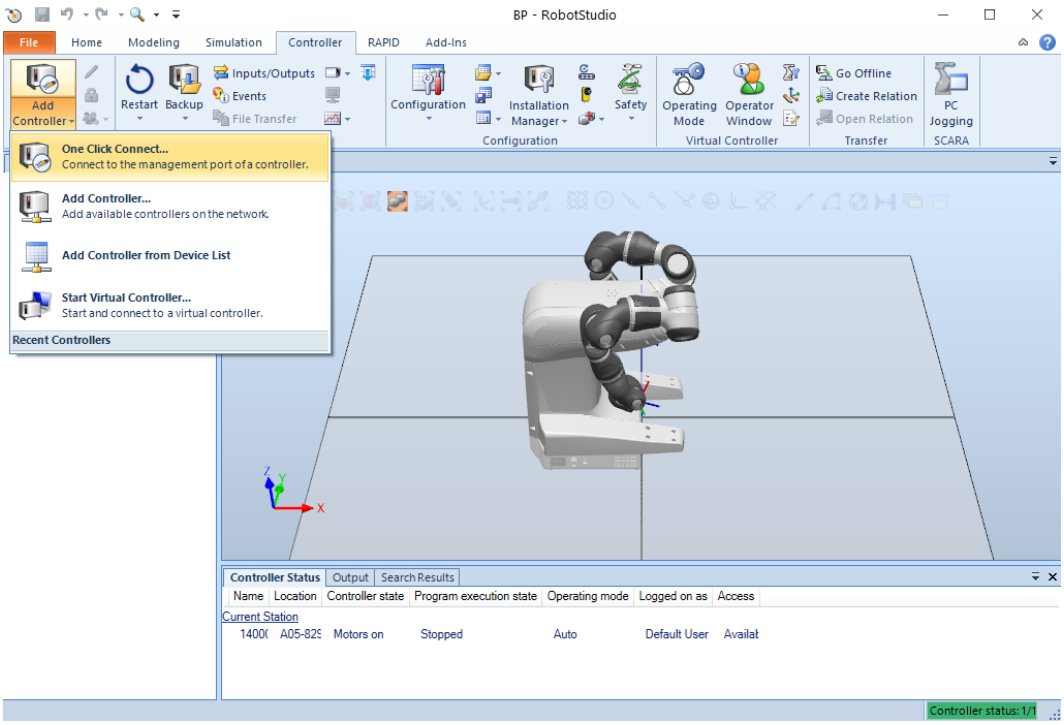


Obr. 6.1: Propojení robota a PLC v hardwarové konfiguraci v prostředí TIA Portal

Posledním krokem je přiřazení jména a nastavení IP adresy robota. Tímto je hardwarové nastavení komunikace hotové a můžeme jej nahrát do PLC.

6.2 Hardwarová konfigurace robota YuMi

Jako první musíme připojit PC k robotu pomocí komunikačního kabelu do servisního portu s označením XP23. Poté se v RobotStudiu v záložce Controller (Controler ⟶ Add Controler) připojíme ke kontroléru robota.



Obr. 6.2: První připojení kontroléru robota k PC v prostředí RobotStudio

Nyní můžeme přistoupit ke konfiguraci komunikace. V první řadě musíme nastavit IP adresu. Nastavená IP adresa musí být stejná, jako námi zvolená adresa robota v prostředí TIA Portal v záložce pro hardwarová nastavení. Dále je potřeba zvolit port, který budeme ke komunikaci využívat (přes servisní port nelze robota řídit). Následně v záložce I/O System v možnosti Industrial Network nastavíme název stanice, také zde v možnosti PROFINET Industrial Network můžeme nastavit rozsah vstupních a výstupních bitů. Další nastavení byly provedeny na základě již zpracovaných odzkoušených projektů a zkušeností kolegů programátorů.

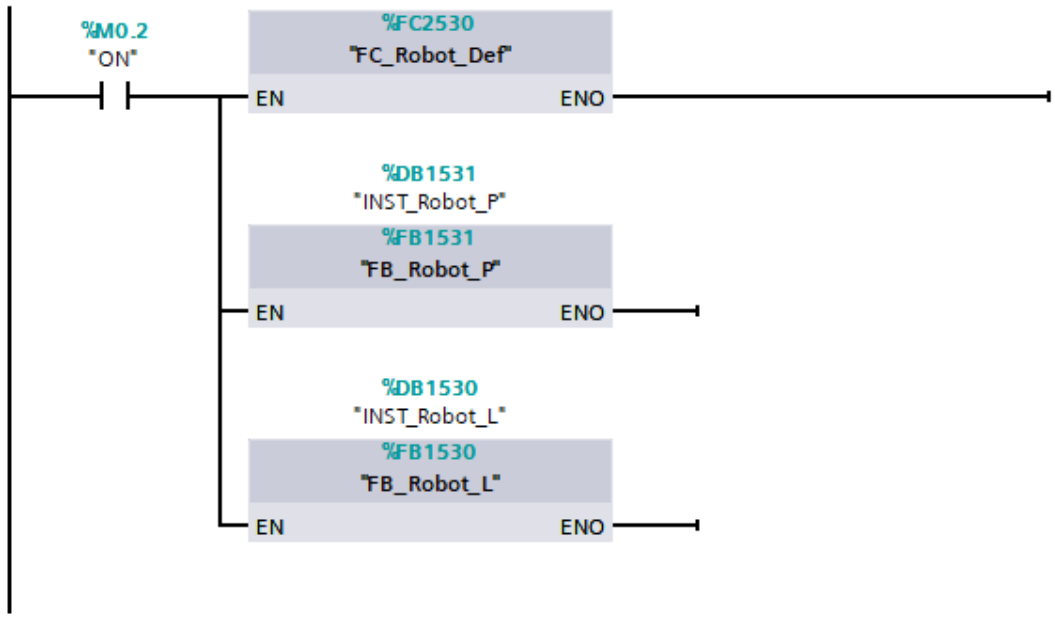
Připojením PC k PLC se zároveň připojujeme do sítě PROFINET. Prostřednictvím vývojových prostředí můžeme pohodlně přistupovat k PLC a i k robotu YuMi. V takovém případě pracujeme v tzv. online režimu, kde můžeme provádět případná další nastavení nebo programování robota přímo na pracovišti.

7 Realizace úlohy

Jak už bylo řečeno, robota YuMi budeme řídit pomocí PLC prostřednictvím průmyslové sběrnice PROFINET. V první řadě si musíme definovat jakým způsobem (jakými signály) budeme chtít robota řídit. Vzhledem k tomu, že robota řídíme ze stejného PLC jakým je řízena celá výrobní linka, můžeme pro řízení robota využít některých již definovaných proměnných pro řízení stanice, na které se nachází robot YuMi.

7.1 Programování PLC

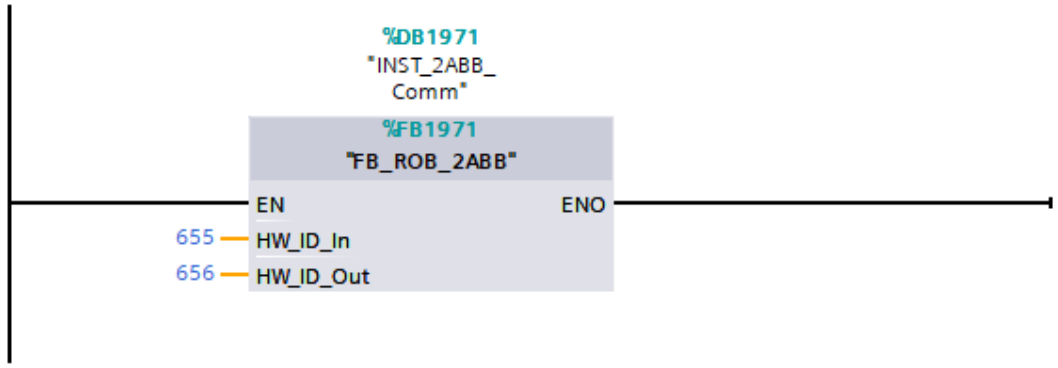
Celá výrobní linka, tedy i stanice na které se nachází robot YuMi, je naprogramována jako stavový automat. Do robota si posílám číslo kroku (stavu) ve kterém se stanice aktuálně nachází (který má robot aktuálně vykonávat), dále posíláme základní proměnné pro řízení robota, jako je např. zapnutí/vypnutí motorů, start/stop/reset programu nebo nastavení rychlosti pohybu robota. Dále do robota posílám souřadnice z kamery (pouze pro P-pravou stranu robota) a řídící signály z OP (operátorského panelu) sloužící například pro různé korekce nebo manuální ovládání chapadla. A vzhledem k tomu že tento robot disponuje dvěma pažemi (P-pravá a L-levá) a každá z nich obsluhuje jinou stanici musím tyto data posílat pro každou stranu robota zvlášť.



Obr. 7.1: Funkce pro robota volané v bloku main(OB1)

Stejně tak si musím posílat informace o stavu robota zpět do PLC. Posílám informace jako je krok, pozice/oblast ve které se robot právě nachází, v jakém režimu robot právě pracuje (automat/servis), detekce kolize robota s překážkou a další (to vše dvakrát, pro každou stranu robota zvlášť).

Pro robota jsou v PLC vytvořeny dva funkční bloky FB ("FB_Robot_P" a "FB_Robot_L"), které zpracovávají informace poslané od robota a jeden blok FC ("FC_Robot_Def"), ve kterém provádíme komunikaci a ošetřujeme některá data posílaná do robota (např. krok). Tyto bloky jsou volány z organizačního bloku main(OB1) viz. obr.7.1.



Obr. 7.2: FB provádějící komunikaci robot-PLC

Komunikaci provádím v FB ("FB_ROB_2ABB") volaném v "FC_Robot_Def" viz. obr.7.2. Do DB instance tohoto FB jsou ukládány veškerá data která posílám do robota (CMD) nebo přijímám z robota (RESP). Jako vstupní parametry jsou systémové konstanty tzv. HW ID (hardware identifier), které jsou napevno přiděleny každému I/O modulu zařízení v komunikační síti (obr.7.3).

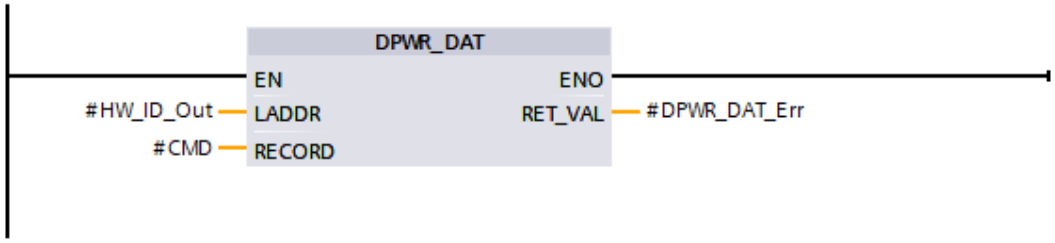
INST_2ABB_Comm

		Name	Data type	Offset	Start value
1		Input			
2		HW_ID_In	HW_IO	0.0	0
3		HW_ID_Out	HW_IO	2.0	0
4		Output			
5		InOut			
6		Static			
7		CMD	Array[0..1] of "UDT_ROB_ABB_cmd"	4.0	
8		RESP	Array[0..1] of "UDT_ROB_ABB_resp"	132.0	

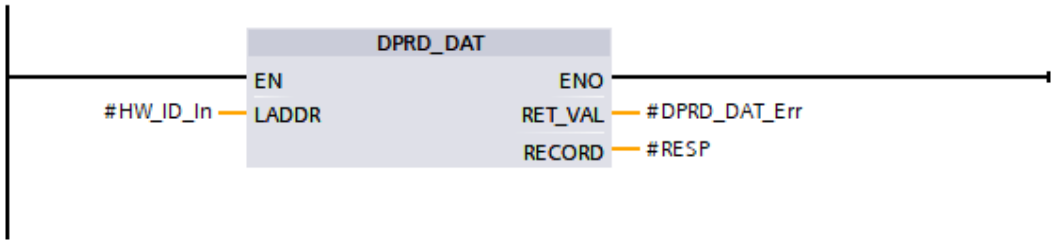
Obr. 7.3: DB instance FB provádějící komunikaci

K přenosu dat využívám instrukce "DPRD_DAT" a "DPWR_DAT". Tyto instrukce lze nalézt v seznamu s rozšířenými instrukcemi v záložce s funkcemi pro

distribuované I/O. Pomocí instrukce “DPRD_DAT“ načítám konzistentní data z I/O modulu a zapisuji do dat v parametru “RECORD“ (obr.7.5).
Instrukce “DPWR_DAT“ slouží k přenosu konzistentních dat v parametru “RECORD“ do I/O modulu (obr.7.4). Dalším vstupním parametrem těchto funkcí je již zmiňované HW ID, které definuje I/O modul zařízení se kterým si data vyměňuji.



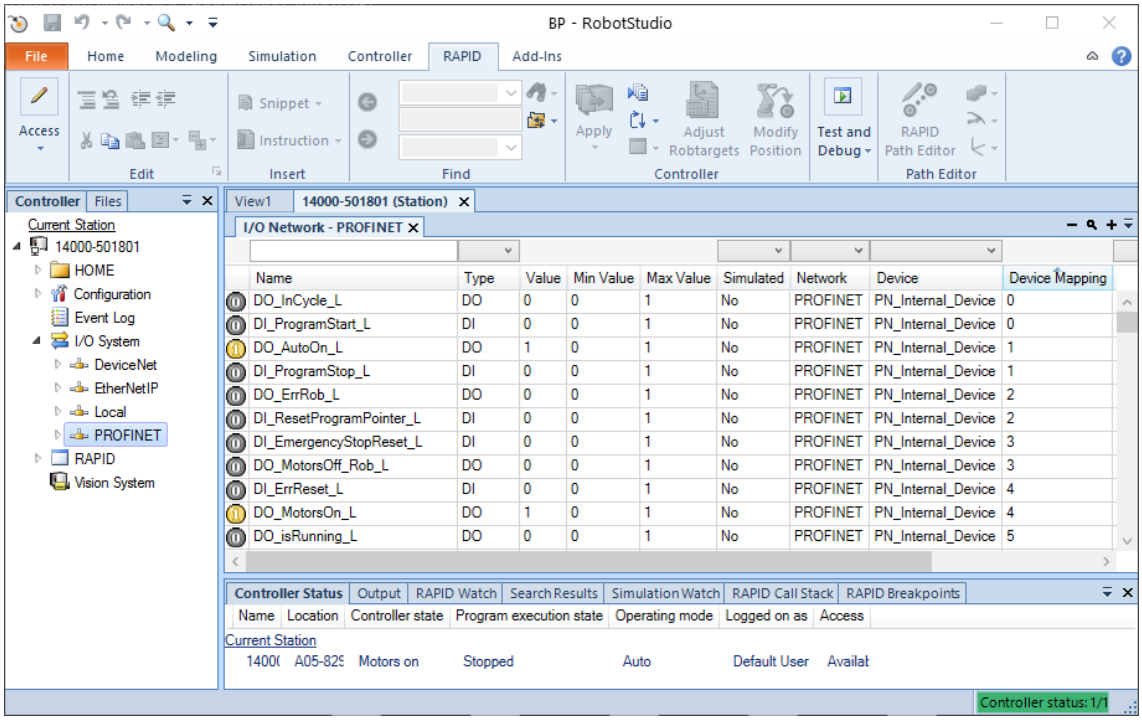
Obr. 7.4: Instrukce “DPRD_DAT“



Obr. 7.5: Instrukce “DPRD_DAT“

7.2 Programování robota YuMi

V první řadě musím definovat proměnná data určená ke komunikaci s PLC, to provedu v kartě “RAPID” (nebo “Controller”) v záložce “Configuration” a nastavení “I/O System“ a poté “Signal”. Zde jsem definoval veškerá data, která v programu využívám. Všechny signály definované pro komunikaci pomocí sítě PROFINET, můžeme poté vidět v záložce “I/O System” a následně PROFINET, viz. obrázek 7.6. Data musí odpovídat datům definovaným v PLC (jak vstupní tak i výstupní data). To znamená, že je třeba vždy zvolit odpovídající název proměnné (ideálně podobný jako v PLC), definovat bitový rozsah dat a zda se jedná o vstupní nebo výstupní data (DI/DO pro digitální vstup/výstup, případně GI/GO pro více bitová (skupinová) data), viz. obrázek 7.6.



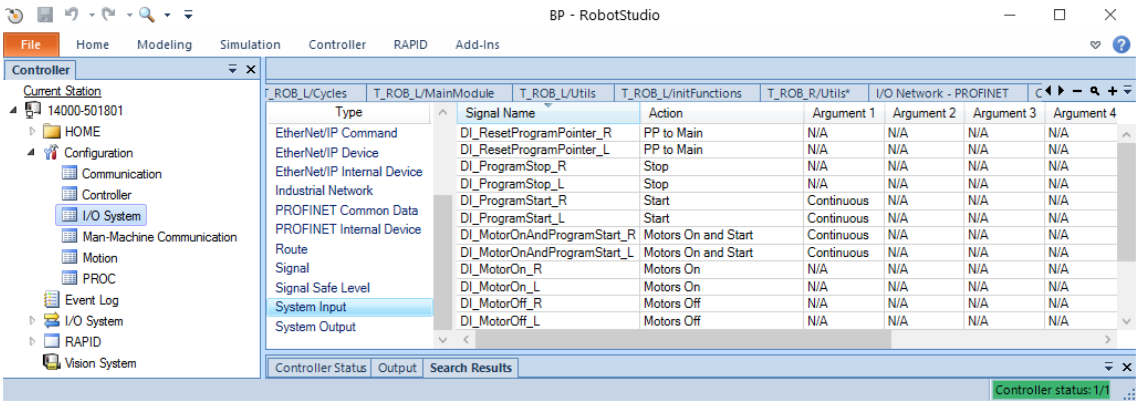
Obr. 7.6: Zobrazení proměnných signálů ke komunikaci s PLC v RobotStudiu

K prvotnímu zpracování dat lze využít tzv. “Crross Connection” signálů. To znamená, že do zvolené proměnné zapíšeme výsledek bitové logické operace (OR nebo AND) dvou až pěti proměnných. To využívám například k provázání signálů od robota se signály z jeho chapadel, viz. obrázek 7.7.

Name	Resultant	Actor 1	Inv	Operator 1	Actor 2	Inv	Operator 2	Actor 3	Inv
cross_DO_isRunning_L	DO_isRunning_L	DO_isRunning_Comm	No	And	DO_isRunning_Grip_R	No	And	DO_isRunning_Rob_L	No
cross_DO_isRunning_R	DO_isRunning_R	DO_isRunning_Comm	No	And	DO_isRunning_Grip_R	No	And	DO_isRunning_Rob_R	No

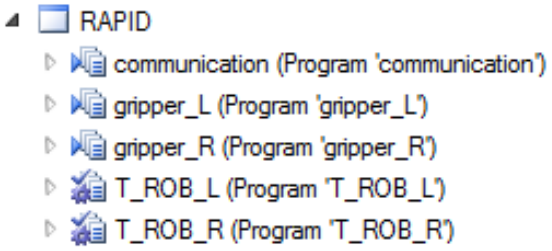
Obr. 7.7: Zpracování dat pomocí “Crross Connection” signálů v RobotStudiu

Dále můžeme zvolené signály přímo navázat na systémové proměnné. Tuto možnost nalezneme v záložce “Configuration”⟶”I/O System” a poté zvolíme buď “System Input” pro navázání na vstupní signály nebo “System Output” pro navázaní na výstupní signály. Zde využívám například pro vypnutí/zapnutí motorů robota, start/stop programu nebo detekci kolize. (Obrázek 7.8)



Obr. 7.8: Navázání proměnných na systémová data

Samotný program je rozdělen na pět úloh (Tasks), viz. obrázek 7.9.



Obr. 7.9: Struktura programu

V první úloze nazvané “communication” jsou definované některé pomocné proměnné, se kterými dále v programu pracuji. Dále jsem zde v proceduře “main()” některým pomocným proměnným přiřadil vstupní hodnoty proměnných (posílané z PLC) a některé naopak přiřadil na výstupní hodnoty proměnných (viz. obrázek 7.10).

```
SetGO GO_CmdRob_R,nActualStepRob_R;
PLC_SpeedRob_R:=(GInput(GI_SpeedRob_R));
```

Obr. 7.10: Ukázka části kódu v proceduře “main()” v úloze “communication”

Další dvě úlohy jsou určeny k ovládání chapadel (“gripper”) a jsou téměř identické, pouze s jediným rozdílem a to v použití proměnných určených zvlášť pro pravou a levou stranu robota. V těchto úlohách pouze definuji (pomocí jednoduché podmínky “IF”) za jakých podmínek se má chapadlo otevřít nebo zavřít. Současně zde provádím konfiguraci chapadla (rychlost sevření, síla sevření, rozteč chapadel při otevření atd.), pokud přijde požadavek na kalibraci je zavolána procedura “GripCalibrate()”, kde jsou nastaveny všechny hodnoty pro správnou funkci chapadel.

Poslední dvě úlohy (“T_ROB_R” a “T_ROB_L”) jsou určeny k řízení samotného robota. Ty jsou dále rozděleny na tři moduly (“Cycles”, “MainModule”, “Utils”), které jsou zobrazeny na obrázku 7.11. jsou si svojí strukturou i programem v mnoha částech velice podobné.

Obr. 7.11: Rozdělení úloh pro robota

V modulu “Utils” jsou definovány veškeré funkce, které pro řízení robota využívám (pro každou stranu robota zvlášť). Jedněmi z nejpoužívanějších funkcí jsou například funkce pro přiblíženi nástroje robota k cílové pozici (funkce nazvaná “Approach”) a funkce pro oddálení nástroje robota od cílové pozice (funkce nazvaná “Depart”). Díky těmto funkcím nám stačí definovat pouze jednu pozici pro tři různé. Příklad použití: Máme definovanou cílovou pozici robota, pomocí předdefinované funkce “RelTool()” upravíme cílovou pozici posunutím o kousek v ose pohybu, do takto vzniklé pozice se dostaneme například pomocí příkazu pro pohyb “MoveJ”, poté zavoláme funkci “Approach()”, která robota zpomaleně přesune do námi definované cílové pozice, pokud jsme úkony v cílové pozici dokončili zavoláme funkci “Depart()” pro bezpečné zpomalené oddálení od cílové pozice.

```
FUNC robtarget CountPositionCamera(Robtarget p_Pos)
    !záporné dynamické korekce - delení 100. Korekce z kamery
    CorrX:=Corr(GInputDnum(GI_CorrX_R))/100;
    CorrY:=Corr(GInputDnum(GI_CorrY_R))/100;
    CorrZ:=Corr(GInputDnum(GI_CorrZ_R))/100;
    !záporné statické korekce - delení 100. Korekce z OP
    CorrX_Rob:=Corr(GInputDnum(GI_CorrXRob_R))/100;
    CorrY_Rob:=Corr(GInputDnum(GI_CorrYRob_R))/100;
    CorrZ_Rob:=Corr(GInputDnum(GI_CorrZRob_R))/100;
    CorrRx_Rob:=Corr(GInputDnum(GI_CorrRxRob_R))/10;
    CorrRy_Rob:=Corr(GInputDnum(GI_CorrRyRob_R))/10;
    CorrRz_Rob:=Corr(GInputDnum(GI_CorrRzRob_R))/10;
    !Prevod kvaternionu do Eulera
    p_Pos_rotX:=EulerZYX(\X,p_Pos.rot);
    p_Pos_rotY:=EulerZYX(\Y,p_Pos.rot);
    p_Pos_rotZ:=EulerZYX(\Z,p_Pos.rot);
    !Pridání úhlových korekcí
    p_Pos_rotX:=p_Pos_rotX+ CorrRx_Rob;
    p_Pos_rotY:=p_Pos_rotY+ CorrRy_Rob;
    p_Pos_rotZ:=p_Pos_rotZ+ CorrRz_Rob;
    !Pridání korekcí k bodu
    p_Pos.trans.x:=p_Pos.trans.x+CorrX+CorrX_Rob;
    p_Pos.trans.y:=p_Pos.trans.y+CorrY+CorrY_Rob;
    p_Pos.trans.z:=p_Pos.trans.z+CorrZ+CorrZ_Rob;
    !Prevod z Eulera na kvaternion
    p_Pos.rot:= OrientZYX(p_Pos_rotZ, p_Pos_rotY, p_Pos_rotX);
    RETURN p_Pos;
ENDFUNC
```

Obr. 7.12: Kód funkce pro přepočet bodu

Jednou z funkcí je také funkce pro přepočet souřadnic bodu pro odběr rezistoru ze střižníku. Tato funkce pomocí dat získaných z kamery a dat pro případné korekce z OP přepíše zadaný bod, který je předán jako parametr této funkce. Funkce je zobrazena na obrázku 7.12. Jelikož je v robotu orientace nástroje definovaná pomocí kvaternionů a korekce je definovaná pomocí Eulerových úhlů je nutné pro přidání korekce orientace převést kvaterniony na Eulerovy úhly a poté zpět, k tomu využívám funkce “EulerZXY()” a “OrientZYX()” (jsou součástí jazyku RAPID).

```
PROC CountPalletePos(num Count,robtarget StartPointType1,robtarget EndPointType1,
                                        robtarget StartPointType2,robtarget EndPointType2)
    VAR num i;
    VAR num j;
    VAR num deltaX;
    VAR num deltaY;
    VAR num deltaZ;
    VAR robtarget ActualTypePosStartPos;
    FOR j FROM 1 TO 2 DO
        IF j=1 THEN
            deltaX:=abs((StartPointType1.trans.x-EndPointType1.trans.x)/(Count-1));
            deltaY:=abs((StartPointType1.trans.y-EndPointType1.trans.y)/(Count-1));
            deltaZ:=abs((StartPointType1.trans.z-EndPointType1.trans.z)/(Count-1));
            ActualTypePosStartPos:=pPallete_StartPointRP_L;
        ELSE
            deltaX:=abs((StartPointType2.trans.x-EndPointType2.trans.x)/(Count-1));
            deltaY:=abs((StartPointType2.trans.y-EndPointType2.trans.y)/(Count-1));
            deltaZ:=abs((StartPointType2.trans.z-EndPointType2.trans.z)/(Count-1));
            ActualTypePosStartPos:=pPallete_StartPointBP_L;
        ENDIF
        FOR i FROM 1 TO Count DO
            IF StartPointType1.trans.x>EndPointType1.trans.x THEN
                pPallete_array2x12_L{j,i}.trans.x:=ActualTypePosStartPos.trans.x-deltaX*(i-1);
            ELSE
                pPallete_array2x12_L{j,i}.trans.x:=ActualTypePosStartPos.trans.x+deltaX*(i-1);
            ENDIF
            IF StartPointType1.trans.y>EndPointType1.trans.y THEN
                pPallete_array2x12_L{j,i}.trans.y:=ActualTypePosStartPos.trans.y-deltaY*(i-1);
            ELSE
                pPallete_array2x12_L{j,i}.trans.y:=ActualTypePosStartPos.trans.y+deltaY*(i-1);
            ENDIF
            IF StartPointType1.trans.z>EndPointType1.trans.z THEN
                pPallete_array2x12_L{j,i}.trans.z:=ActualTypePosStartPos.trans.z-deltaZ*(i-1);
            ELSE
                pPallete_array2x12_L{j,i}.trans.z:=ActualTypePosStartPos.trans.z+deltaZ*(i-1);
            ENDIF
             pPallete_array2x12_L{j,i}.rot:=ActualTypePosStartPos.rot;
        ENDFOR
    ENDFOR
ENDPROC
```

Obr. 7.13: Kód funkce pro výpočet pozice hřebenu na paletce

Dále je tu funkce (určená pro levou stranu robota) pro výpočet pozice odebrání hřebenu s kapslemi z paletky s hřebeny. Vstupními parametry funkce je číslo maximálního počtu hřebenů založených na paletce a dvakrát pozice odebrání prvního a posledního hřebenu (pro možnost použití dvou typů paletek). Pomocí těchto údajů jsou následně dopočítány jednotlivé odebíraní pozice. Tato funkce je zobrazena na obrázku 7.13.

V dalším modulu nazvaném “MainModule” se nacházejí dvě důležité procedury volané v proceduře “main()”. První procedura upravuje číslo kroku podle hodnoty přicházející z PLC a druhá procedura na základě tohoto upraveného čísla kroku volí odpovídající cyklus robota, viz. obrázek 7.14 (opět pro pravou a levou stranu robota zvlášť). Tyto cykly jsou definované v modulu “ProgramCycle”.

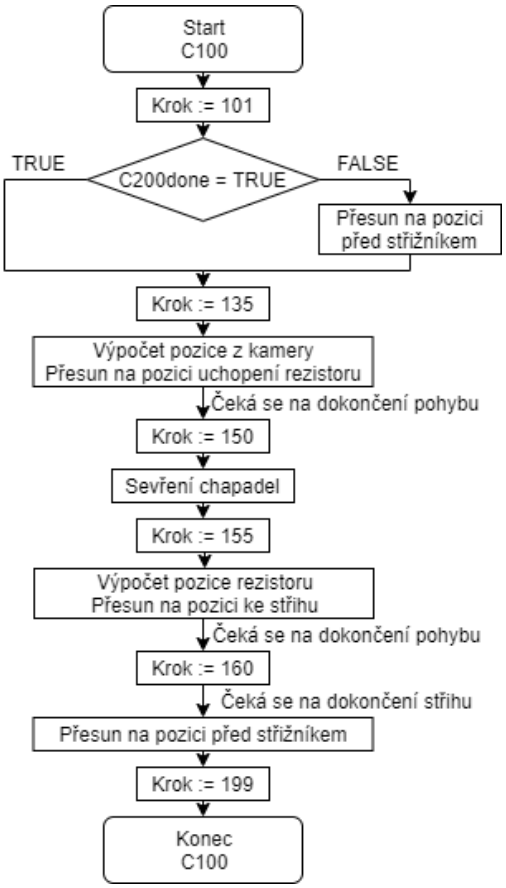
```
PROC Production()
    TEST nActualStepRob_L
    CASE 100:
        C100_stack_pick;
    CASE 200:
        C200_stack_put;
    CASE 300:
        C300_pill_pick;
    CASE 400:
        C400_pill_welding;
    CASE 900:
        C900_INIT;
    CASE 2000:
        C2000_nok;
    DEFAULT:
    ENDTEST
ENDPROC
```

```
PROC Production()
    TEST nActualStepRob_R
    CASE 100:
        C100_resistor_pick;
    CASE 200:
        C200_resistor_welding1;
    CASE 300:
        C300_resistor_welding2;
    CASE 900:
        C900_INIT;
    CASE 2000:
        C2000_nok;
    DEFAULT:
    ENDTEST
ENDPROC
```

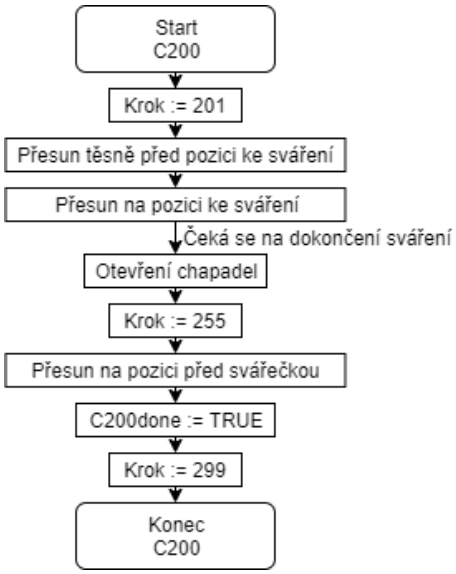
Obr. 7.14: Procedury na volbu cyklu robota na základě čísla kroku (levá a pravá strana robota)

Pro pravou stranu robota mám definováno celkem pět cyklů. První cyklus pojmenovaný “C100_resistor_pick” má za úkol dojet pro rezistor do střižníku, počkat na zkrácení vývodů a poodjet ven od střižníku. Kód tohoto cyklu je principiálně popsán diagramem zobrazeným na obrázku 7.15. Pro usnadnění pohybu v okolí střižníku, jsem si definoval tzv. “pracovní objekt” (Work object data), který definuje nový souřadnicový systém pro orientaci robota.

Další dva cykly pojmenované “C200_resistor_weldling1” a “C300_resistor_weldling1” mají za úkol přesunout rezistor ze střižníku ke svářečce a po dokončení sváření se vzdálit od svářečky. Tyto cykly jsou téměř shodné, jediný rozdíl je v použití různých pozic pro sváření rezistoru. Kód těchto cyklů je principiálně popsán diagramem zobrazeným na obrázku 7.16.

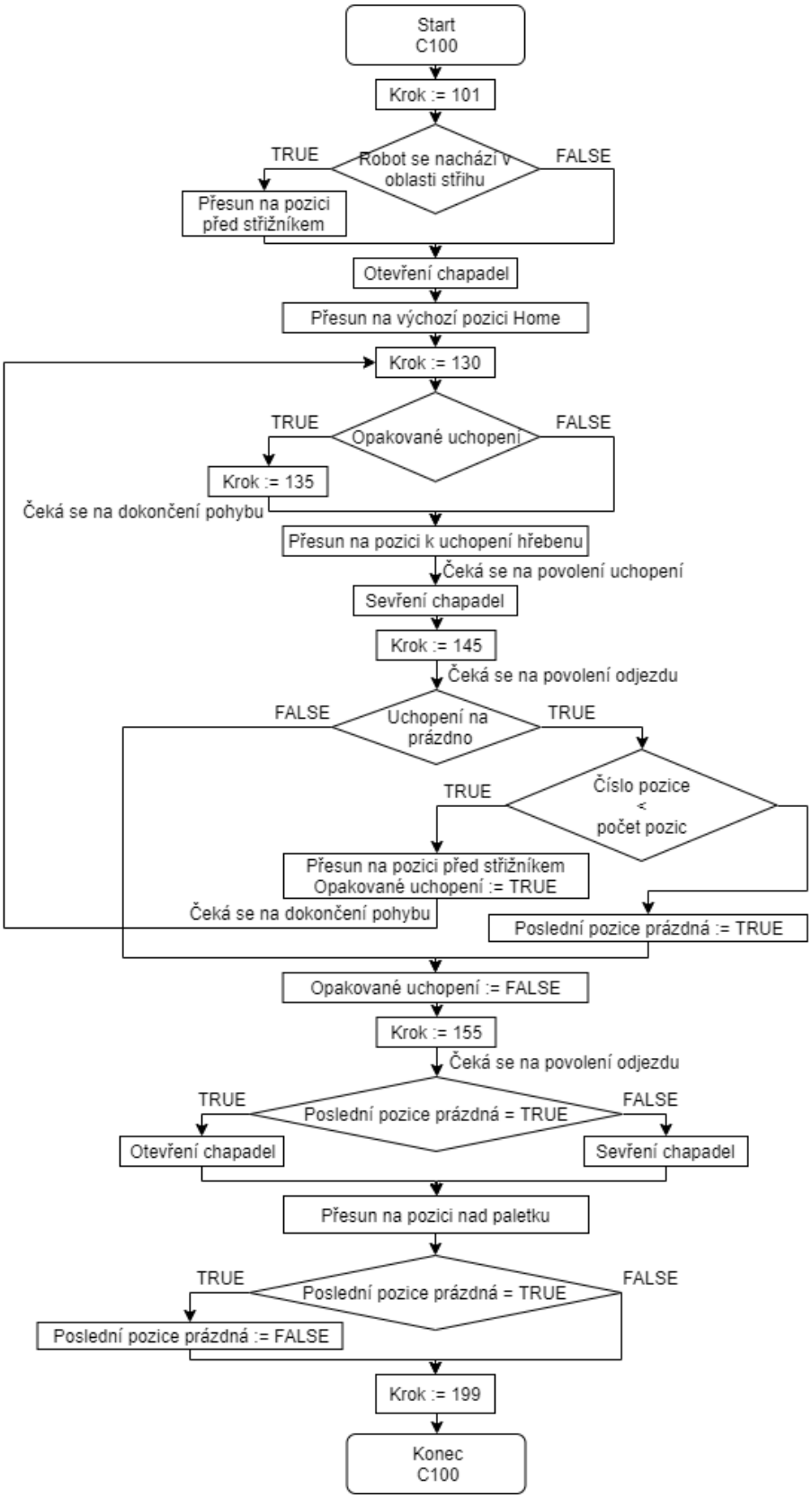


Obr. 7.15: Cyklus pravé strany robota "C100_resistor_pick"



Obr. 7.16: Cyklus pravé strany robota "C200_resistor_weldling1"

Cyklus "C900_INIT" slouží k nastavení výchozích hodnot programu a má za úkol robota bezpečně dostat do výchozí "Home" pozice. K rozpoznání kde se robot (ve velmi omezeném prostoru k pohybu) právě nachází využívám definovaných oblastí. Tyto oblasti jsem definoval vždy v působišti jednotlivých cyklů robota. Na základě zjištění této oblasti se zvolí optimálně definované průjezdní body robota k bezpečnému dosažení "'Home" pozice. Na závěr po dosažení této pozice se volá cyklus "C2000_nok". Tento cyklus má za úkol očistit nástroj robota od případných nečistot nebo od vadného rezistoru. To je provedeno opakovaným otevřením a zavřením chapadla a následným projetím kartáčem umístěným po pravé straně robota.



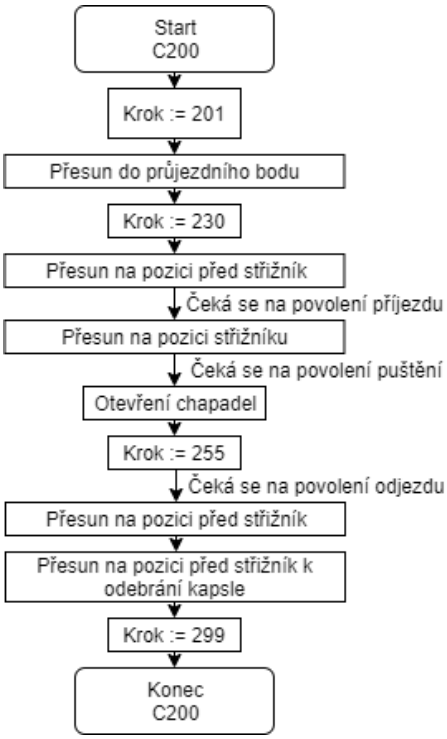
Obr. 7.17: Cyklus levé strany robota “C100_stack_pick”

Pro levou stranu robota mám definováno celkem šest cyklů. První cyklus pojmenovaný “C100_stack_pick” je určen k vyzvednutí hřebenu s kapslemi z paletky. V tomto cyklu bylo potřeba ošetřit stav, kdy na paletce nejsou obsazeny všechny pozice. V takovém případě se snažím uchopit v řadě následující hřeben a to opakuji dokud robot nezaregistruje hřeben nebo nedosáhne poslední pozice. Tento cyklus je zakončen stavem (pokud jsou v paletce hřebeny), kdy robot drží hřeben s kapslemi nad paletkou. Kód tohoto cyklu je principiálně popsán diagramem zobrazeným na obrázku 7.17

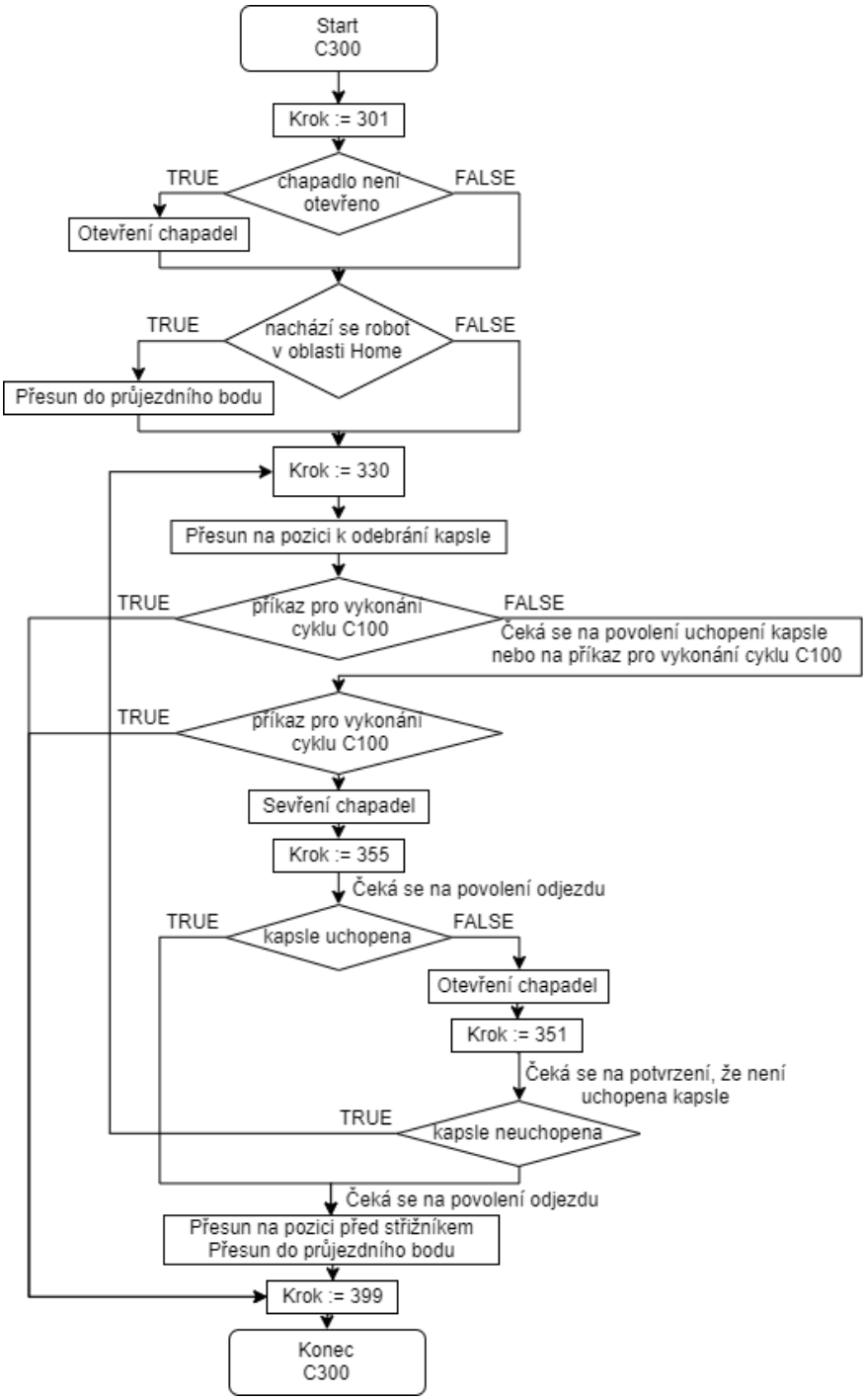
Dalším je cyklus “C200_stack_put”, ten se stará o založení hřebenu s kapslemi do střižníku. Kód tohoto cyklu je principiálně popsán diagramem zobrazeným na obrázku 7.18

Cyklus “C300_pill_pick” má za úkol vyzvednout kapsli ze střižníku a připravit se na přesun ke svářečce. Zde jsem (podobně jako v prvním cyklu) musel ošetřit možnost chybějící kapsle na hřebenu. V takovém případě se pokoušíme uchopit následující, a to v případě opakovaného neúspěchu opakujeme dokud nedosáhneme pozice poslední kapsle. Kód tohoto cyklu je principiálně popsán diagramem zobrazeným na obrázku 7.19

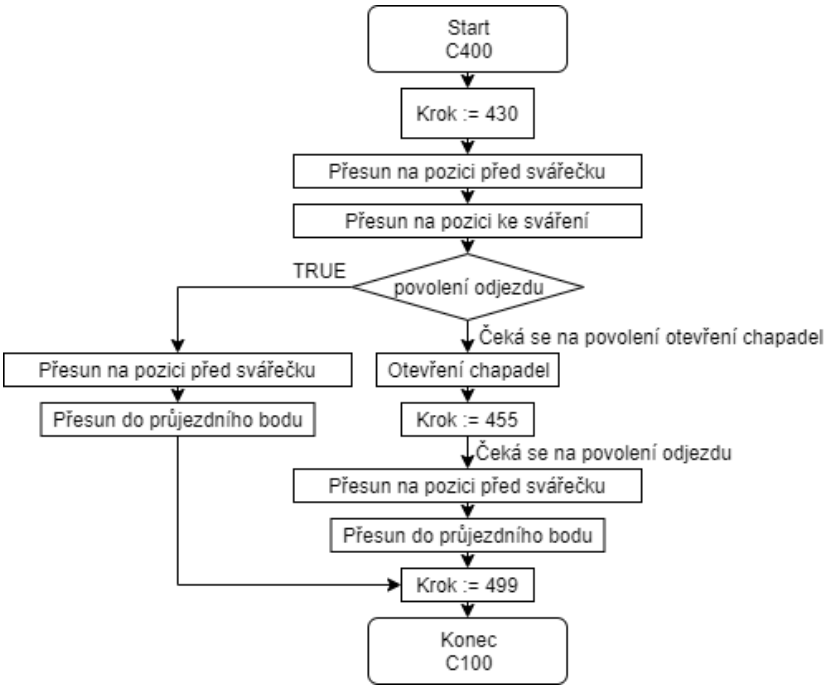
Cyklus “C400_pill_welding” má za úkol přesunout uchopenou kapsli od střižníku ke svářečce, kde vyčkává dokud nebude sváření dokončeno, poté poodjede a robot je připraven na další cyklus. Kód tohoto cyklu je principiálně popsán diagramem zobrazeným na obrázku 7.19



Obr. 7.18: Cyklus levé strany robota “C200_stack_put”



Obr. 7.19: Cyklus levé strany robota “C300_pill_pick”



Obr. 7.20: Cyklus levé strany robota “C400_pill_welding”

Poslední dva cykly pro levou stranu robota (“C900_INIT” a “C2000_nok”) plní stejnou funkci jako stejně pojmenované cykly pro pravou stranu robota. Jsou však modifikovány pro levou stranu robota, to například znamená, že očištění nástroje je prováděno na kartáči po levé straně robota a cykly využívají pozice a oblasti definované taktéž pro levou stranu robota.

Veškeré pozice a oblasti použité v programu jsem definoval manuálně pomocí FlexPendantu. Většina povolení na která se během cyklů čeká, jsou řídícími signály z PLC.

7.3 Odladění robota

Pro finální podobu programu bylo nutné vše otestovat a následně odladit. Testy jsem prováděl jak v režimu manuálním, tak poté i v režimu automatickém pomocí FlexPendantu.

Nejprve bylo nutné funkčnost programu důkladně otestovat v manuálním režimu. V první fázi testování nebyla linka ještě plně oživena a tedy PLC neposílalo žádné (nebo nesprávné) řídící signály. Pro odzkoušení bylo nutné si tyto signály nasimulovat. Některé signály lze pohodlně během zkoušení v manuálním režimu nasimulovat na FlexPendantu, jiné jsem si zase pro tyto účely přímo definoval v kódu (např. výběr typu kapsle a typu hřebenu). Na FlexPendatu jsem si postupně spouštěl jednotlivé cykly robota. V manuálním režimu jsem pomocí ukazatele v programu postupně

testoval funkčnost jednotlivých řádků kódu. Z bezpečnostních důvodu jsem pohyby robota testoval se sníženou rychlostí robota na minimum a následně jsem rychlost postupně navyšoval. Během těchto pohybů jsem robota monitoroval, zda vykonává operace dle požadavků, jsou-li definované správně koncové pozice a nedochází-li v některých úsecích pohybu ke kolizím. Takto jsem program otestoval v několika cyklech a opravil nefunkční části kódu nebo doplnil ty, které nebyli správně definovány. Aby nedocházelo ke kolizím s okolím, bylo nutné například doplnit několik průjezdních pozic a ošetřit některé stavy. Jedním z takových stavů bylo například zajistit otevření nebo zavření chapadla na začátku každého cyklu, bez ohledu na to v jakém stavu se chapadlo právě nachází po dokončení předchozího cyklu (např. se stávalo, že jsem se během testování s ukazatelem přesunul z cyklu, kde je chapadlo sevřeno, na začátek cyklu pro uchopení rezistoru a robot se snažil uchopit rezistor sevřeným chapadlem). V této fázi testování jsem některé cílové pozice robota musel definovat s určitou tolerancí. Důvodem bylo, že stanice s robotem nebyla ještě plně oživena a já nebyl schopen manipulovat s rezistorem ani s kapsli se správně zastřiženými vývody ze střižníku.

Následně kdy byla stanice zprovozněna v manuální režimu, jsem mohl ovládat pohyby střižníku a uchopovat rezistor nebo kapsli již se zkrácenými vývody. Pak bylo možné definovat přesné pozice robota k operacím sváření. Kvalita sváru byla kritickým požadavkem aplikace, bylo tedy nutné definovat pozice robota velmi precizně. V tomto stavu s uchopenými komponenty jsem s manuálně ovládanou svářečkou cykly robota ještě několikrát odzkoušel. V této chvíli jsem si byl jist, že samotná aplikace robota je odladěná a je možné jej spustit v automatickém režimu.

První spuštění robota v automatickém režimu probíhalo pod mým dohledem a s omezenou maximální rychlostí. V případě nežádoucích pohybů by bylo nutné včas zakročit. V okamžiku nesprávných pohybu bych musel zastavit pohyb robota (například pomocí stop tlačítka na FlexPendantu). K tomuto kroku však nedošlo. Následně bylo odzkoušeno, že vše funguje správně i v automatickém režimu. Teď už zbývalo jen doladit dynamické vlastnosti robota, jako je například citlivost na vnější procesní síly nebo maximální rychlost pohybu robota řízenou z PLC.

8 Zhodnocení dosažených výsledků

Cílem této práce bylo navrhnout a realizovat úlohu řízení kolaborativního robota YuMi pomocí PLC SIMATIC S7-1500. Navrhovaná úloha představovala reálnou aplikaci pro náročného koncového zákazníka. Proto jsem ke všem činnostem v průběhu zpracování přistupoval velmi zodpovědně.

Pro samotnou realizaci bylo nejdůležitější precizně definovat jednotlivé pozice robota, aby jsem zajistil dostatečnou přesnost, která byla vzhledem k zadání velice důležitá. V průběhu implementace aplikace robota jsem narážel na problém s přetížením robota, což znamenalo velmi časté a náhodné přerušování pohybu robota. Na FlexPendantu se zobrazovala chybová hláška s detekcí kolize nebo přetížení. K tomu docházelo v důsledku působení vnějších procesních sil na rameno robota během pohybu. Proto bylo nezbytně nutné správně nastavit zatížení robota. V úsecích, kde k tomu docházelo bylo nutné nastavit menší citlivost na tyto síly. Před finální předáním robotické aplikace (s celou výrobní linkou) koncovému zákazníkovi, jsem vše pečlivě odzkoušel a odladil, viz kapitola 7.3.

Kolaborativní robot YuMi je řízen pomocí PLC. Na základě dat posílaných po komunikační síti PROFINET, je řízen chod programu robota v návaznosti na cyklus stanice výrobní linky, kde je robot instalován. Požadovaná doba výrobního cyklu byla 9 sekund. To se mi podařilo splnit. Robot je schopen uchopit jednotlivé komponenty a dosáhnout s nimi cílových pozic. Přesnost robota byla na hraně přesnosti udávané výrobcem, ale přesto stále postačující ke splnění požadavků na kvalitu sváru.

Zmetkovitost (efektivnost) celé výrobní linky se provádí výpočtem, nejčastěji je udáván ukazatel OEE (celková efektivnost zařízení a výroby). Zmetkovitost ve finální fázi (před předáním linky zákazníkovi) se pohybovala okolo 1-2%, z toho přímo proces sváření a manipulace do sváření (úkol robota) je cca 1/4, tedy 0,25-0,5%. Nicméně v první fázi oživování a odlaďování výrobní linky se zmetkovitost pohybovala okolo 10%. Nyní je již linka v plném provozu u koncového zákazníka. Veškeré požadavky na funkci robota byly splněny.

9 Závěr

Tato práce se zabývá řízením kolaborativního robota YuMi. Kolaborativní robot YuMi je díky svým flexibilním pažím, pohyblivostí podobným člověku, dobře uplatnitelný pro práci ve stísněných podmínkách, kde jiné roboty by se jen stěží pohybovaly. Taktéž díky svým vlastnostem kolaborativního robota je zajisté dobrou volbou k přímé spolupráci s člověkem. Robota řídíme pomocí PLC SIMATIC S7-1500, který je aktuálně nejnovějším produktem z řady řídících systémů společnosti SIEMENS. Jedná se o modulární systém PLC. Tento systém je určen k řízení strojů střední velikosti i náročných aplikací.

Úloha demonstrující ovládání kolaborativního robota, která je v práci popsána, je reálnou aplikací pro koncového zákazníka. Firma ve které je tato práce realizována, mi tímto umožnila aplikovat své znalosti (teoretické i praktické) na reálném zadání úkolu robota, který bude později spuštěn v reálném provozu. Vzhledem k tomu, že se jedná se o zákazníka, který si přeje zůstat v anonymitě, a který v rámci celého projektu trvá na utajení informací, není možné v této práci zveřejnit žádné citlivé údaje. Není mi tedy například umožněno vytvořit detailnější popis produktu nebo jeho výrobní technologie, dále nesmím zveřejnit projekt ve kterém se vyskytují IP adresy, hesla nebo další, z pohledu bezpečnosti, citlivé údaje projektu. V rámci bakalářské práce mi bylo umožněno použít některé 3D modely výrobní linky a zveřejnit celý vlastní zdrojový kód pro robota (pouze kód bez HW konfigurace a dalších konkrétních nastavení). Zdrojový kód je zveřejněn jako příloha této práce.

V rámci předložené bakalářské práce byl vytvořen řídící program pro programovatelný automat SIMATIC S7-1500, který zpracovává data určená pro řízení robota a zajišťuje komunikaci s robotem po průmyslové sběrnici PROFINET. Tato sběrnice byla zvolena na základě dostupných možností a požadavků na přenášená data. Dále byl vytvořen software pro řízení robota YuMi, který na základě přijatých dat z PLC obsluhuje stanice výrobní linky. Úloha robota je rozdělena na dvě části. V první části robot odebírá rezistory malých rozměrů ze střižníku, kam jsou postupně přiváděny, a přesouvá se s nimi na pozici, kde jsou přivařeny k připraveným vodičům. V druhé části robot zajišťuje zakládání hřebenu, na kterém jsou připevněny kapsle s třaskavinou, do střižníku. Poté kapsle jednotlivě odebírá a přesouvá se s nimi na pozici, kde jsou přivařeny k připraveným rezistorům na vodičích.

Výsledná aplikace robota je otestovaná a plně funkční, viz kapitola 8.

Literatura

[1] Robot nebo kobot? V čem se liší? *Talentica* [online]. ČR: Eva Vaculíková, 2018, 5. 4. 2018 [cit. 2019-12-18]. Dostupné z: https://www.talentica.cz/robot-nebo-kobot/

[2] Vzniká centrum pro roboty, kteří spolupracují s lidmi. Firmy si vyzkouší jejich schopnosti. *Česká televize* [online]. Ostrava: ČT24/ČTK, 2018, 10. 5. 2018 [cit. 2019-12-18]. Dostupné z: https://ct24.ceskatelevize.cz/regiony/2474686-vznika-centrum-pro-roboty-kteri-spolupracuji-s-lidmi-firmy-si-vyzkousi-jejich

[3] BARTOŠÍK, Petr. Bezpečnost kolaborativních robotů. *AUTOMA* [online]. 2017, **2017**(8-9), 3 [cit. 2019-12-18]. Dostupné z: https://automa.cz/Aton/FileRepository/pdf_articles/11040.pdf

[4] YuMi: YuMi - IRB 14000. *ABB* [online]. [cit. 2019-12-21]. Dostupné z: https://new.abb.com/products/robotics/cs/prumyslove-roboty/yumi

[5] Budoucnost robotiky a automatizace závisí na společné práci lidí a robotů. *ABB* [online]. ABB, 2015 [cit. 2019-12-23]. Dostupné z: https://library.e.abb.com/public/191cbbc9809f411aac23d1feeac1a9d0/yumi_backgrounder.pdf

[6] Robot YuMi na filmovém festivalu ve Zlíně. In: *AUTOMA* [online]. cz: AUTOMA, 2017 [cit. 2019-12-23]. Dostupné z: https://automa.cz/cz/web-clanky/robot-yumi-na-filmovem-festivalu-ve-zline-0_10468/

[7] Robotic Industry News: Program Off-Line with ABB RobotStudio. *RobotWorx* [online]. [cit. 2020-04-28]. Dostupné z: https://www.robots.com/blogs/program-off-line-with-abb-robotstudio

[8] *RobotWare: Industrial Robot Controller Software* [online]. [cit. 2020-04-28]. Dostupné z: https://library.e.abb.com/public/c351e246eeb20a19c12570b90039d5dc/ RobotWare%20data%20sheet.pdf

[9] *Technical reference manual: RAPID Instructions, Functions and Data types* [online]. [cit. 2020-04-28]. Dostupné z: https://library.e.abb.com/public/688894b98123f87bc1257cc50044e809/Technical%20reference%20manual_RAPID_3HAC16581-1_revJ_en.pdf

[10] Programovatelné automaty (PLC). *Elektroprumysl.cz* [online]. 2016, 26.8.2016, , 1 [cit. 2019-12-23]. Dostupné z:

https://www.elektroprumysl.cz/automatizace/programovatelne-automaty-plc

[11] Vlastnosti PLC SIMATIC S7-1500. *SIEMENS* [online]. [cit. 2019-12-27]. Dostupné z: http://www1.siemens.cz/ad/current/index.php?ctxnh=7eaed34950

[12] Nejrychlejší řídicí systém pro automatizaci. *SIEMENS* [online]. [cit. 2019-12-27]. Dostupné z: https://new.siemens.com/cz/cs/products/automation/systems/industrial/plc/simatic-s7-1500.html

[13] Increase productivity with the ultimate power. *SIEMENS* [online]. , 2-3 [cit. 2019-12-29]. Dostupné z: https://www.click4business-supplies.com/resources/articles/dffa-b10140-03-7600.pdf

[14] S7-1500/S7-1500F Technical Data. *SIEMENS* [online]. 2015, , 9-10 [cit. 2019-12-29]. Dostupné z: http://www.sermax.my/img/products/industrial-products/ProgrammableLogicControllers/download/01-SIEMENS-S7-1500-Datasheet.pdf

[15] Siemens TIA Portal — jednotné vývojové prostředí pro automatizaci v průmyslu: — jednotné vývojové prostředí pro automatizaci v průmyslu. *AUTOMA* [online]. 2011, **2011**(03) [cit. 2020-05-14]. Dostupné z: https://automa.cz/cz/casopis-clanky/siemens-tia-portal-jednotne-vyvojove-prostredi-pro-automatizaci-/v-prumyslu-2011_03_43212_6058/

[16] JIRGL, Miroslav. *Laboratorní cvičení BPGA: Cvičení se systémy SIEMENS*. Brno, 2018. Skripta. FEKT VUT.

[17] Komunikační rozhraní aneb připravte se na IoT. *KONDIK.cz* [online]. 2018 [cit. 2019-12-29]. Dostupné z: https://www.kondik.cz/komunikace

[18] Product specification: IRB 14000. *ABB* [online]. 2018 [cit. 2019-12-29]. Dostupné z: https://library.e.abb.com/public/c9f66b2b80bc42bba49c20970bed1b0d/3HAC052982%20PS%20IRB%2014000-en.pdf?x-sign=09i961Kw9WYypn8AUDHOiC8hUT/kTjLE+VudbTeATcD5UcH0biKarKLuRwEL+rKw

[19] ZBOŽÍ: 6ES7523-1BL00-0AA0. *JORK* [online]. [cit. 2019-12-30]. Dostupné z: http://www.jork.shop/produkt/automatizacni-systemy/ridici-system-simatic-s7-1500/digitalni-vstupy-a-vystupy/6es7523-1bl00-0aa0-89129.htm

[20] Profinet versus Profibus. *AUTOMA* [online]. 2012, **2012**(5), 1 [cit. 2020-01-02]. Dostupné z: https://automa.cz/Aton/FileRepository/pdf_articles/9618.pdf

[21] PP001544. *Farnell* [online]. [cit. 2020-01-02]. Dostupné z: https://cz.farnell.com/pro-power/pp001544/profibus-dp-l2-fip-1x2x0-64-pvc/dp/2827742

[22] *PROFINET: Profinet - Standard pro průmyslový Ethernet v automatizaci* [online]. SIEMENS, **2005**(04) [cit. 2020-01-02]. Dostupné z: http://stest1.etnetera.cz/ad/current/content/data_files/automatizacni_systemy/prumyslova_komunikace/profinet/profinet_04_2005_cz.pdf

[23] BELDEN 70006E.00500: průmyslový kabel Profinet. *Lancomat.cz* [online]. [cit. 2020-01-02]. Dostupné z: https://www.lancomat.cz/70006e-00500-prumyslovy-kabel-profinet-cat-5e-sf-utp-awg22-drat-pvc-plast-bal-500m-barva-zelena-p15435/#gallery

Seznam symbolů, veličin a zkratek

PLC	Programovatelný logický automat – Programmable Logic Controller
CPU	Centrální procesorová jednotka – Central Processing Unit
I/O	Vstup/Výstup – Input/Output
WAN	Rozlehlá síť – Wide Area Network
LAN	Lokální síť – Local Area Network
USB	Univerzální sériová sběrnice – Universal Serial Bus
IRT	Izochronní reálný čas – Isochronous Real Time
RT	Reálný čas – Real Time
IP	Internetový protokol – Internet Protocol
MAC	Řízení přístupu k médiu – Media Access Control
HW	Hardware
DI/DO	Digitální vstup/výstup – Digital Input/Output
GI/GO	Skupina vstupů/výstupů – Group Input/Output
OP	Operátorský panel – Operator Panel
OEE	Celková efektivnost zařízení – Overall Equipment Effectiveness

Seznam příloh

A Kód robota YuMi

Kód jednotlivých modulů programu robota je na přiloženém CD.
Soubor s názvem “Kod_robot_YuMi.pdf”.