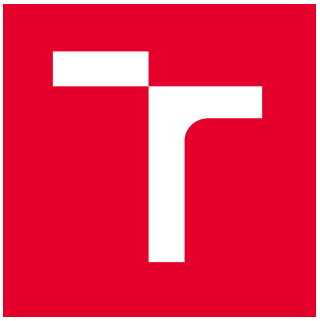


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV VÝROBNÍCH STROJŮ, SYSTÉMŮ A ROBOTIKY

INSTITUTE OF PRODUCTION MACHINES, SYSTEMS AND ROBOTICS

VIRTUÁLNÍ ZPROVOZNĚNÍ ROBOTIZOVANÉHO VÝUKOVÉHO VÝROBNÍHO SYSTÉMU

VIRTUAL COMMISSIONING OF A ROBOTIC LEARNING PRODUCTION SYSTEM

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Martin Baránek**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Jan Vetiška, Ph.D.**

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav výrobních strojů, systémů a robotiky
Student:	**Bc. Martin Baránek**
Studijní program:	Výrobní stroje, systémy a roboty
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Jan Vetiška, Ph.D.**
Akademický rok:	2023/24

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Virtuální zprovoznění robotizovaného výukového výrobního systému

Stručná charakteristika problematiky úkolu:

Stále aktuálním cílem průmyslu je zvyšování produktivity práce. Možností jak tohoto cíle dosáhnout je robotizace rutinních činností. Cílem práce je zjistit aktuální stav v oblasti virtuálního zprovoznění robotizovaných výrobních systémů a zpracovat výukový příklad, pro vzdělávání budoucích absolventů.

Cíle diplomové práce:

Rešerše dané problematiky.
Tvorba modelové úlohy.
Návrh PLC programů.
Virtuální zprovoznění.
Reálné zprovoznění.

Seznam doporučené literatury:

SICILIANO, Bruno a Oussama. KHATIB. Springer handbook of robotics. Berlin: Springer, 2008. ISBN 978-3-540-23957-4.

KOLÍBAL,Z. a kol. Roboty a robotizované výrobní technologie. VUTIUM Brno, 2016, ISBN 978-80-214-4828-5.

NOF, S. Y. Springer Handbook of Automation. Springer, 2009. 1812 s. ISBN 978-3-540-78830-0.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2023/24

V Brně, dne

L. S.

doc. Ing. Petr Blecha, Ph.D., FEng. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

ABSTRAKT

Diplomová práce se zabývá virtuálním zprovozněním výukového robotického pracoviště. Teoretická část se zabývá rešerší současného stavu poznání využívání robotů v průmyslu, způsoby jejich programování, možnostmi řízení robotických pracovišť a metodami postupů pro virtuální zprovoznění robotických pracovišť. Praktická část je věnována virtuálnímu zprovoznění pracoviště v programu Tecnomatix Process Simulate, konkrétně zpracování modelu pracoviště pro simulaci a vytvoření robotických programů, a vytvoření PLC řízení a HMI v prostředí TwinCAT 3.

ABSTRACT

Diploma thesis deals with the virtual commissioning of a educational robotic workplace. The theoretical part of the thesis deals with the research of the current state of knowledge of the use of robots in industry, methods of their programming, possibilities of controlling robotic workplaces and methods of procedures for virtual commissioning of robotic workplaces. The practical part of the thesis is devoted to the virtual commissioning of the workplace in Tecnomatix Process Simulate, specifically the processing of the workplace model for simulation and creation of robotic programs, and the creation of PLC control and HMI in the TwinCAT 3 environment.

KLÍČOVÁ SLOVA

virtuální zprovoznění, robotické pracoviště, simulace pracoviště, Process Simulate, TwinCAT 3, OPC UA, PLC

KEYWORDS

virtual commissioning, robotic workplace, simulation of a workplace, Process Simulate, TwinCAT 3, OPC UA, PLC

BIBLIOGRAFICKÁ CITACE

BARÁNEK, Martin. Virtuální zprovoznění robotizovaného výukového výrobního systému [online]. Brno, 2024 [cit. 2024-05-23]. Dostupné z: https://www.vut.cz/studenti/zav-prace/detail/157110. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav výrobních strojů, systémů a robotiky. Vedoucí práce Jan Vetiška.

PODĚKOVÁNÍ

Tímto bych rád poděkoval vedoucímu práce Ing. Janu Vetiškovi, Ph.D. za jeho rady, odbornou pomoc a trpělivost se mnou. Velké poděkování patří také hlavně mojí rodině a mým nejbližším za podporu během celého studia.

ČESTNÉ PROHLÁŠ ENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Jana Vetišky, Ph.D. a s použitím literatury uvedené v seznamu.

V Brně dne 23. 5. 2024 ……

Baránek Martin

1 ÚVOD

Automatizace výrobních a montážních procesů s využitím průmyslových robotů je celosvětově dlouhodobě rostoucí trend. Motivace pro stále vyšší zájem firem o implementaci robotů do výrobního řetězce jsou zřejmé – snaha o konkurenceschopnost ve vysoce kompetitivním prostředí trhu zvyšováním efektivity procesů, kvality, nebo snižování procesních časů a nákladů spojených se zaměstnancem.

Bezesporu nejširší uplatnění a utilizaci průmyslových robotů nabízí sériová výroba. Často relativně jednoduché, ale především repetitivní úkony jsou schopny vykonávat rychleji, kvalitněji a s vyšší opakovatelností než lidský pracovník. Dělnické pozice pro sériovou výrobu je navíc běžně dlouhodobě obtížné obsadit z důvodu často nižších platových ohodnocení. Výhodou plně automatizovaného výrobního systému je také možnost jednoduššího plánování a simulování z důvodu absence stochastického prvku, tedy lidského pracovníka.

Návrh robotického pracoviště, jeho sestavení a uvedení do provozu je časově náročná úloha vyžadující spolupráci týmu složeného z konstruktérů, programátorů a elektrotechniků. Ve snaze zjednodušit a zefektivnit tento proces se stává běžnou praktikou virtuální zprovoznění. Virtuální zprovoznění je v podstatě návrh pracoviště, který typicky zahrnuje zhotovení digitálního 3D modelu, tzv. offline programování robotů, kdy jsou programy napsány ve virtuálním prostředí robotického controlleru, a vytvoření programů pro PLC řízení. Následně je možno pomocí simulace ověřit funkčnost pracoviště, interakci mezi jednotlivými komponentami nebo správnost programů. Virtuální zprovoznění tedy nabízí větší flexibilitu při ladění, implementaci změn v procesu návrhu, umožňuje testovat chování při kritických podmínkách, pomáhá identifikovat a odstranit chyby před reálným nasazením do provozu a eliminuje potřebu fyzického prototypu pracoviště.

Předmětem této práce je zmapovat současný stav poznání v oblasti virtuálního zprovoznění robotických pracovišť, vytvořit modelovou úlohu na základě reálné stanice nacházející se v prostorech laboratoří Ústavu výrobních strojů, systémů a robotiky na Vysokém učení technickém v Brně, navrhnout a vytvořit programy pro PLC řízení v prostředí TwinCAT 3, provést virtuální zprovoznění v prostředí Tecnomatix Process Simulate a následně ověřit toto řešení reálným zprovozněním předlohového pracoviště.

2 MOTIVACE

Hlavním zdrojem motivace pro zpracování diplomové práce zabývající se virtuálním zprovozněním robotických pracovišť byla vidina příležitosti rozšířit znalosti o průmyslové robotice, automatizaci výrobních procesů a tvorba řídících PLC programů, které jsem nabyl v průběhu studia magisterského programu na Ústavu výrobních strojů, systému a robotiky.

Automatizaci a robotizaci bych se rád věnoval i po ukončení studia. Vzhledem k aktuálním trendům v průmyslu, kde je neustálý důraz na zvyšování efektivity práce, snaha o digitalizaci a virtualizaci napříč různými procesy věřím, že virtuální zprovoznění bude standardním způsobem zprovozňování robotických pracovišť a své dovednosti budu moct využít v praxi.

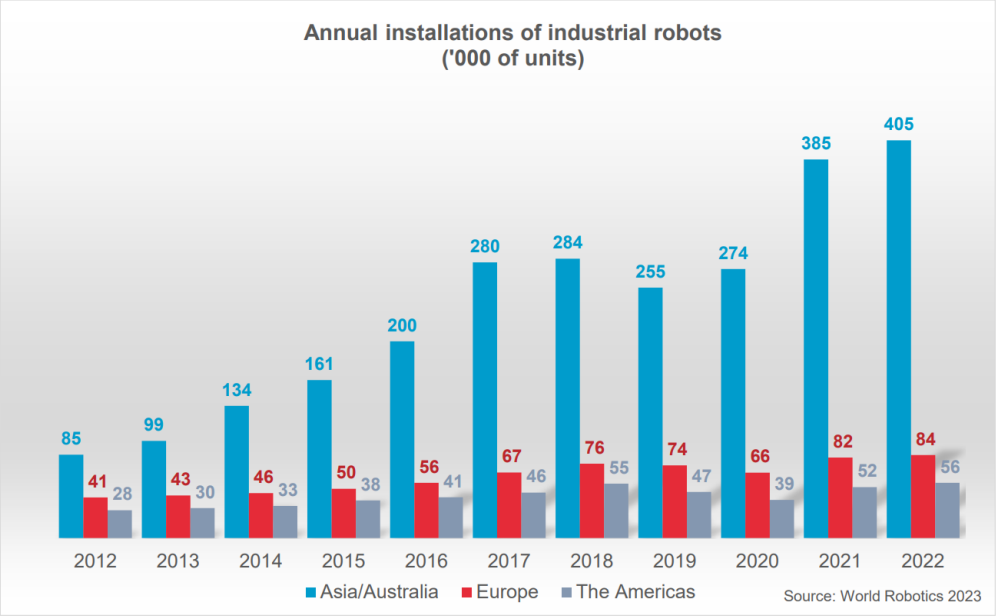
3 PŘEHLED SOUČASNÉHO STAVU POZNÁNÍ

3.1 Roboty v průmyslu

Využití průmyslových robotů ve výrobních procesech je dnes již běžnou praxí i v menších firmách. Obecná snaha automatizovat všechny možné úkony je opodstatněná. Pořizovací a režijní náklady robotizovaných pracovišť se za poslední roky snižují a ekonomická návratnost této investice je velmi rychlá, často v nižších jednotkách let. Nahrazení lidské pracovní síly robotem umožňuje využít lidský potenciál pro složitější úlohy, než je často monotónní práce, dosáhnout vyšší efektivity, kvality a opakovatelnosti. [1]

Zásadní roli zastupují roboty i v konceptu průmyslu 4.0 a jeho principech, jako jsou digitalizace, virtualizace, modularita a interoperabilita. V kontextu digitalizace a virtualizace je velkým fenoménem koncept digitálních dvojčat, kde celé lince, resp. každé její součásti jako jsou stroje a roboty, je vytvořen virtuální protějšek s daty a vlastnostmi, které reprezentují reálny stav daného zařízení. Digitální dvojčata je pak možno využívat k simulacím nebo optimalizaci procesů. [1]

Podle mezinárodní robotické federace (IFR) počet celosvětově instalovaných robotických jednotek v roce 2022 překročil rekordní milník 553 000 kusů, což je více než trojnásobek ve srovnání s daty z roku 2012. Největším trhem pro průmyslové roboty je Asie. Jen tam bylo v tomto roce instalováno 73 % z celkového počtu. Druhý v pořadí je evropský trh a třetí americký (viz obr. 1). Více než 79 % všech robotů bylo instalováno v pěti zemích. Od roku 2013 patří prvenství Číně, druhá příčka patří Japonsku, třetí Spojené státy americké, čtvrtá je Jižní Korea a páté Německo. Momentální zpomalení celosvětové ekonomiky se samozřejmě promítá i do tohoto odvětví, avšak dlouhodobé vyhlídky předpokládají neustálý růst prodejů robotických jednotek. [2]



Obr. 1) Počet ročně instalovaných průmyslových robotů (tisíce) [2]

3.2 Programování robotů

Operace, resp. pohyby, které robot vykonává jsou řízeny robotickým programem uloženým v jeho paměti. Takových programů může být v robotu uloženo několik. Pořadí jejich načtení a vykonání je možno během procesu měnit na základě signálu z řídícího PLC. [3,5]

Obsahem takového programu (viz obr. 2) jsou typicky definice souřadnic bodů vztažené k některému souřadnému systému, definice rychlostí a zón, definice koncového efektoru a bloky robotických operací. Bloky operací jsou z většiny tvořeny pohybovými příkazy, příkazy pro nastavení hodnoty digitálních výstupních signálů (např. pro otevření chapadla), příkazy pro vyčkání robotu na zpětnou vazbu od řídícího PLC ve formě změny hodnoty digitálního vstupního signálu (např. senzorem potvrzené otevření chapadla) atd. Pohybové příkazy pak v sobě nesou obvykle tyto informace:

- koncový bod daného robotického pohybu,
- typ pohybu, kterým se robot do daného bodu přemístí (lineární, point-to-point)
- rychlost pohybu,
- definice zóny,
- definice koncového efektoru.[3, 4]

Programování robotů lze rozdělit na dva, resp. tři způsoby. Na základě interakce mezi programátorem a robotem rozlišujeme online programování, offline programování a jejich kombinaci, tedy hybridní programování. [3,5]

3.2.1 Online programování

Online programování probíhá v přímé interakci operátora s robotem. Jedním z nejstarších způsobů, který se využívá dodnes, je tzv. „Lead through programming“. Princip spočívá v tom, že robot je naveden pomocí Teach pendantu (viz obr. 2) potřebného bodu, jehož pozice je uložena do programu a následně je možnost tento bod přiřadit některému z pohybů. Především u kolaborativních robotů se v posledních letech nabízí možnost tzv. „Walk through programming“. Oproti předchozímu způsobu je zde robot naváděn přímo rukou operátora, což celý proces ještě zjednodušuje. Předností metod online programování je jejich jednoduchost, intuitivnost a možnost okamžité zpětné vazby při následném spuštění programu, avšak pro komplexnější trajektorie obsahující větší množství bodů jsou takřka nepoužitelné z důvodu jejich časové náročnosti. Nevýhodou je potřeba fyzického pracoviště a přítomnost na něm, což vyžaduje náklady ve formě pořízení a sestavení pracoviště ještě před tím, než je možno ověřit jeho funkčnost nebo náklady spojené s přerušením výroby. [3,5,6]

Obr. 2) Teach pendant pro robot značky KUKA

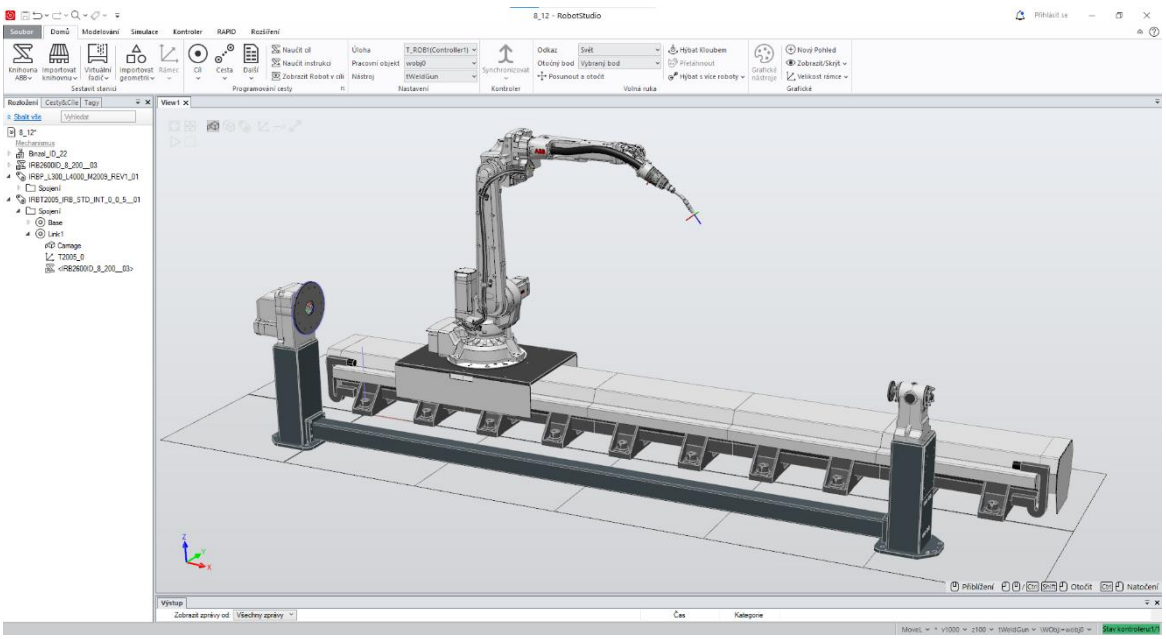
3.2.2 Offline programování

Offline programování (OLP) nevyžaduje přítomnost reálného robotu a robotický program může být vytvořen externě. Tato metoda využívá softwarové nástroje umožňující robot naprogramovat ve virtuálním prostředí (viz obr. 3), do kterého je možno importovat 3D modely samotného robotu, koncových efektorů, nebo celé robotické pracoviště se všemi součástmi. Typicky existují dvě varianty OLP tvorby robotického programu.[5,6,7]

První variantou je obyčejné textové prostředí podobné běžným programátorským vývojovým prostředím. Jedná se o nejjednodušší princip tvorby programu z pohledu funkčnosti prostředí, a i z tohoto důvodu je součástí každého softwaru. Samotný kód je pak psán ručně s využitím různých doplňků jako jsou našeptávače, moduly pro signálovou logiku, nebo moduly pro konkrétní technologické procesy jako je svařování atp. Funkčnost takto vytvořeného programu je následně možno ověřit simulací v grafickém prostředí softwaru.[5,7]

Druhou variantou je tvorba programu v grafickém 3D rozhrání daného softwaru. Zásadní předností tohoto způsobu je jeho přehlednost. Umožňuje jednodušší tvorbu bodů trajektorií, volně pohybovat s robotem, získat velmi přesný časový odhad robotických cyklů, nebo okamžitou zpětnou vazbu při chybě v programu. Samotné programování probíhá výběrem z předdefinovaných příkazů a funkcí z nabídky. [5,7]

Robotické programy vytvořené OLP metodou je běžně při nasazování na reálný robot potřeba odladit a upravit, protože souřadnice bodů trajektorií pohybů mají typicky vůči bodům na reálném pracovišti odchylku vznikající rozměrovými rozdíly mezi digitálním pracovištěm a jeho reálnou podobou. Korekce odchylek bodů se provádí ručním přepsáním hodnot souřadnic v programu na Teach pendantu, nebo přemístěním robotu v manuálním ovládacím módu do požadovaného bodu a změnou původní definice na aktuální polohu. Jedná se pak tedy o kombinaci online a offline přístupu. [5,7]



Obr. 3) Ukázka 3D prostředí softwaru ABB RobotStudio

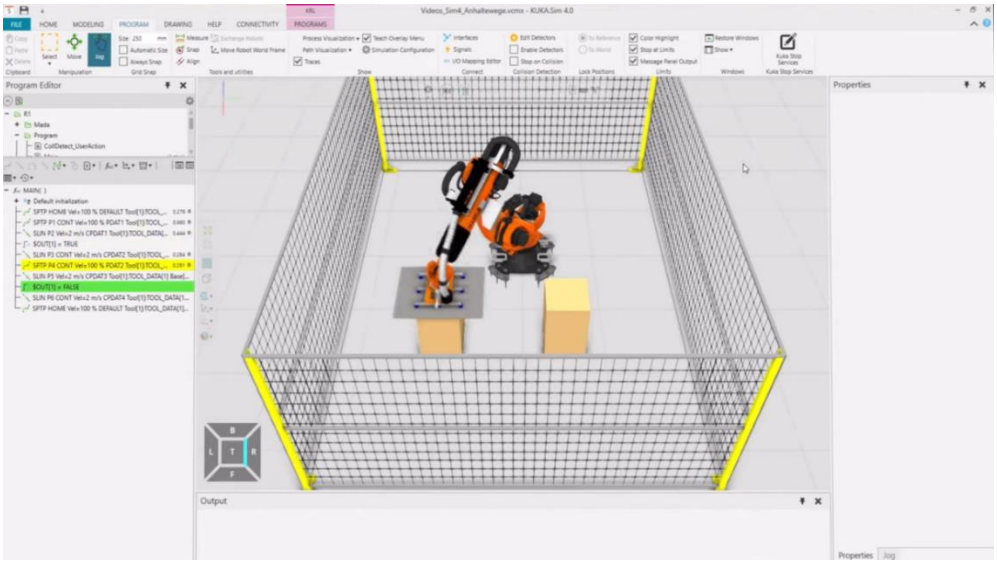
3.2.3 Software pro offline programování

Každý výrobce průmyslových robotů v rámci svého produktového portfolia nabízí softwarové řešení pro offline programování. Software výrobců typicky navíc nabízí možnost simulace robotických cyklů, digitální knihovnu modelů robotů včetně virtuálních controllerů a různých příslušenství s možností importování do uživatelského projektu, funkce pro nahrání vlastních 3D modelů součástí, nebo moduly pro práci se signálovou logikou. Běžně existuje více variant licencí těchto softwarů. Základní variantu licence je možno rozšířit pomocí příplatkových balíčků o další funkce, které obsahují např. moduly pro technologické operace, jako je svařování nebo lepení. Softwary nabízené výrobci robotů jsou [8, 9]:

- KUKA.Sim – KUKA AG
- RobotStudio® Suite – ABB
- ROBOGUIDE – FANUC
- MotoSim – Yaskawa
- Robotics Suite – Stäubli
- RT ToolBox – Mitsubishi
- RoboDK – Universal Robots

KUKA.sim

Firma KUKA nabízí pro offline programování a simulaci robotů této značky softwarové řešení KUKA.Sim (viz obr. 4). Program je zde tvořen v KRL (KUKA Robot Language) editoru, ve kterém je možno robot naprogramovat i bez hlubších znalostí jazyku. Součástí základní licence v nejnovější verzi KUKA.Sim 4.0 je dostupná rozsáhlá online knihovna s aktuálně podporovanými modely robotů, integrovaný import CAD modelů v běžných formátech, moduly pro modelování vlastních komponent, funkce pro kontrolu kolizí, kontrola dosažitelnosti, možnost přesného zjištění doby cyklu odpovídající reálným časům, editor signálů a možnost simulace doběhových drah robotu. Software podporuje možnost zobrazení vytvořených simulací ve VR (virtuální realitě) nebo exportování simulace pro zobrazení na mobilním zařízení pomocí aplikace „Mobile viewer app“.[9]



Obr. 4) Prostředí softwaru KUKA.Sim 4.0 [9]

Zákazník má možnost rozšířit funkcionalitu základní verze zakoupením zvlášť licencovaných modulárních balíčků[9]:

- KUKA.Sim Modelling – Možnost vytvoření individuální knihovny součástí z vlastních CAD dat, kterým je možno definovat kinematiku a fyzikální data. Vlastní modely je možno použít pro senzoriku, signálovou logiku, nebo materiálový tok v simulaci.
- KUKA.Sim Connectivity – Rozšiřující balíček pro virtuální zprovoznění pracoviště. Umožňuje propojení se simulátory chování SIMIT a WinMOD, nebo propojení se simulátory PLC hardware jako je TwinCAT od firmy Beckhoff, PLCSIM Advanced pro TIA Portal od Siemens, nebo CodeSys.
- KUKA.Sim ArcWelding – Rozšíření pro programování robotických svařovacích operací, který simuluje funkce doplňkového softwaru pro řídící systém svařovacích robotů ArcTech Basic. Umožňuje definovat optimální polohu robota pro svařovací proces, nebo definovat způsob nájezdu robota ke svaru.

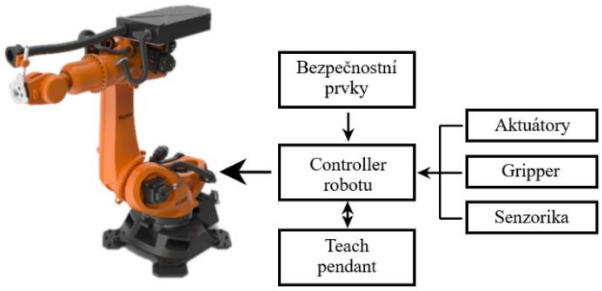
3.3 Řízení robotického pracoviště

Řízení robotického pracoviště jako celku zajišťuje počítač, který zapisuje vstupní signály, vyhodnocuje logiku ve formě programu a nastavuje hodnotu výstupních signálů. Podle velikosti pracoviště, jeho komplexnosti a počtu funkčních prvků je možno volit jeden ze tří způsobů řízení:

- řízení controllerem robotu,
- řízení přes OPC server,
- řízení pomocí PLC[5,7].

3.3.1 Řízení controllerem robotu

Využití řídícího systému průmyslového (controlleru) robotu pro řízení celého pracoviště je nejjednodušším řešením pro technologicky nenáročné aplikace, typicky manipulační a paletizační úlohy. Ovládací prvky a senzory použité v rámci pracoviště jsou přímo připojeny na rozhrání digitálních vstupů a výstupů, které komunikuje s počítačem robotu fungujícím na real-time systému. Není tedy nutné k řízení využít žádný nadřazený řídící prvek, jehož pořízení by navýšilo náklady a zbytečně zkomplikovalo způsob zapojení. Řídící systém robotů má však omezený počet digitální vstupů a výstupů, které mohou být k němu připojeny, což představuje omezení v případě potřeby rozšíření daného pracoviště o další prvky. Limitací může být také menší variabilita použitelných programů a absence některých funkčností, kterými disponují jiné způsoby řízení, jako možnost vytvoření vlastního HMI (Human-machine interface) aj. Schéma tohoto druhu řízení je vidět na obr. 5.[5]



Obr. 5) Schéma řízení pomocí robotického controlleru [5]

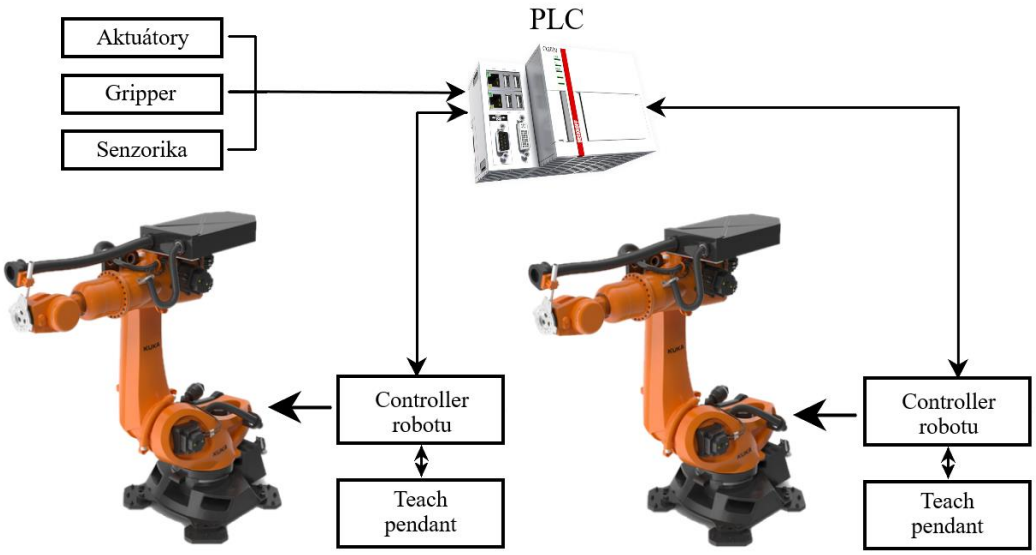
3.3.2 Řízení přes OPC UA server

Řízení pracoviště přes OPC UA (Open Platform Communication Unified Architecture) server nachází využití při požadavku pro vzdálený přístup k pracovišti. Princip tohoto způsobu spočívá v komunikaci dvěma zařízeními pomocí OPC UA standardu, který umožňuje číst nebo zapisovat data v reálném čase nezávisle na platformě. OPC server může být v rámci pracoviště spuštěn na počítači robotu, PLC, nebo sběrnicovém coupleru a vyměňovat informace se vzdáleným počítačem. Výhodou tohoto způsobu je vysoká míra flexibility, možnost vzdáleného monitorování procesních hodnot a sběr dat na cloudové uložiště. [5]

3.3.3 Řízení pomocí PLC

PLC (Programmable logic controller) řízení robotického pracoviště je obecně nejpoužívanějším způsobem. Pracoviště je řízeno a ovládáno v reálném čase z externě připojeného PLC přes některou z průmyslových sběrnic (např. Profinet, EtherCAT, apod.). Z pohledu hierarchie zapojení je PLC nadřazeno ostatním prvkům pracoviště. Výhodou použití tohoto způsobu řízení je modularita PLC hardwaru, jehož rozhrání digitálních vstupů a výstupů je řešeno pomocí karet, které je v případě potřeby možno jednoduše rozšířit. Díky tomu je tato aplikace vhodná pro rozsáhla pracoviště, nebo možná pro řízení více pracovišť pomocí jednoho PLC. Schéma zapojení tohoto způsobu řízení je vidět na obr. 6. [5]

V praktické části této práce bude zpracováno řízení výukové pracoviště pomocí PLC výrobce Beckhoff, jehož hardwarem je nyní již osazeno. Pro práci s tímto hardwarem dodává výrobce software TwinCAT, který bude využit pro virtuální zprovoznění a programování. [5]



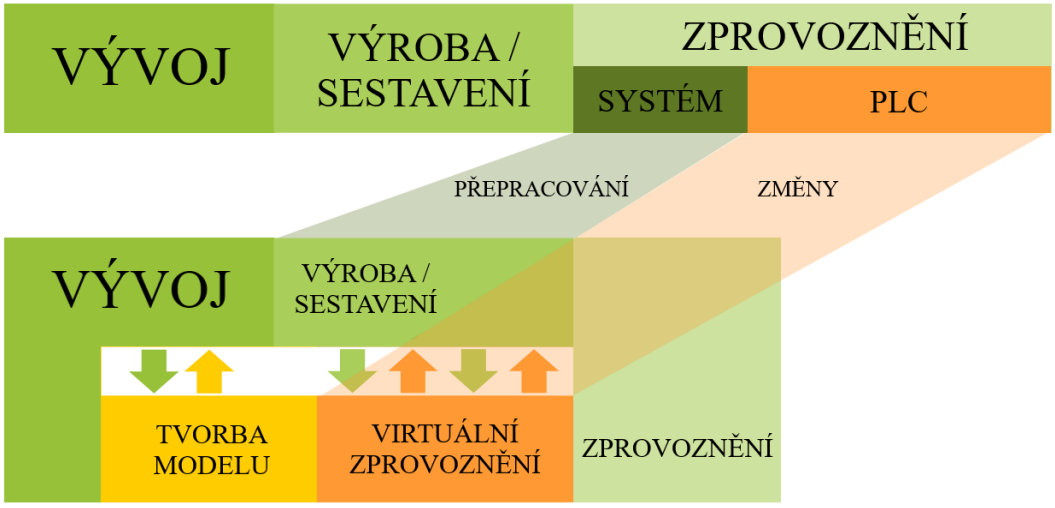
Obr. 6) Schéma řízení pomocí PLC [5]

3.4 Virtuální zprovoznění

Vývoj a stavba robotických pracovišť je komplexní problematika, protože kombinuje aspekty oborů strojírenství, elektrotechniky a programování. Mnoho firem stále využívá tradiční postup, kdy je pracoviště navrhnuto, vyrobeno, sestaveno a následně na místě zprovozněno. Právě při reálném zprovoznění, tedy poslední fázi tohoto procesu, se typicky poprvé projeví všechny chyby ve funkčnosti pracoviště. Fyzické úpravy mechanických a elektronických prvků pracoviště, opravy chyb robotických a řídících PLC programů a další odlaďování poté dokončení samotného zprovoznění značně zpožďují. [10,11]

Virtuální zprovoznění je relativně novou metodou, která spočívá ve vytvoření digitálního modelu odpovídajícího reálné podobě pracoviště, offline programování robotických operací a vytvoření řídících programů pro PLC. Tento proces může probíhat paralelně s vývojem a výrobou reálného pracoviště (viz obr. 7). Čas potřebný pro reálné zprovoznění pracoviště u zákazníka je tak možno nejen zásadně zkrátit, ale také urychlit celý proces od jeho návrhu až po dokončení, což zvyšuje ziskovost samotného projektu pro zpracovatele. Funkčnost a chování takto vytvořeného pracoviště je pak možno ověřit ve virtuálním prostředí pomocí simulace a až následně fyzicky postavit a zprovoznit reálné pracoviště u zákazníka. [10, 11]

Téměř veškeré chyby v robotických i PLC programech, které by mohly v některých případech zapříčinit i poškození reálného robota, je tak možno bezpečně odhalit a opravit již během simulace s nižšími náklady. Pracoviště je v simulaci možno provozovat při kritických podmínkách, nebo simulovat různé chybové stavy jako poškození senzoriky a následně sledovat chování řídícího programu v reakci na ně. Výhodou je také možnost průběžné prezentace simulace zákazníkovi, na jehož případné připomínky nebo požadované změny je možno snáze reagovat. Virtuální zprovoznění má oproti tradičním postupům mnoho výhod, avšak zásadní nevýhodou může být potřebná znalost této problematiky, znalost a pořízení několika využívaných softwarů, nebo vytvoření modelu pracoviště. [10, 11]



Obr. 7) Porovnání tradičního postupu a virtuálního zprovoznění [10]

Existuje několik různých metod, jak přistupovat k virtuálnímu zprovoznění a jeho testování. Základem každé metody je propojení simulačního modelu s PLC řízením. Rozdíly mezi metodami jsou pak v:

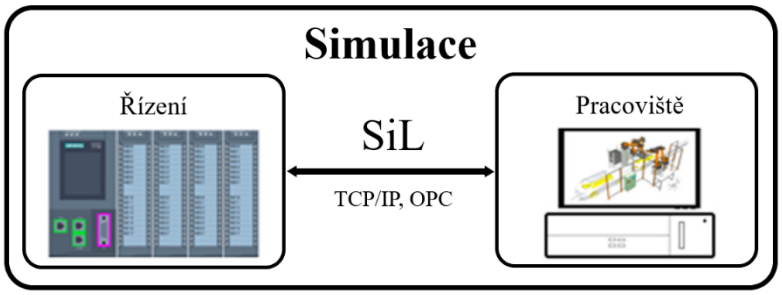
- typu a rozsahu použitých modelů,
- variantách použitých simulačních nástrojů,
- způsobu komunikace mezi systémem a simulací,
- typu a rozsahu simulovaných součástí systému. [11]

3.4.1 Model in the loop

Model in the loop (MiL), tzv. model ve smyčce, je typicky prvním krokem při testování virtuálního zprovoznění. Řídící logika PLC, HMI a jejich chování je v tomto případě reprezentováno zjednodušeným modelem ve formě blokového diagramu vytvořeného pomocí software jako je MATLAB, resp. Simulink. Takový model řízení je pak možno spustit v real-time simulaci propojený s modelem robotického pracoviště. Řídící model posílá povely pracovišti, od kterého získává zpětnou vazbu. Během simulace je sledována odezva modelu pracoviště na povely řídícího modelu a ověřena správnost návrhu řídící logiky ve srovnání s požadovaným chováním reálného pracoviště. [11, 12]

3.4.2 Software in the loop

Software in the loop (SiL), tzv. software ve smyčce, je způsob real-time simulace modelu pracoviště propojeného s reálným řídícím programem a reprezentací HMI v uzavřené smyčce (viz obr. 8). Řídící PLC včetně HMI je spuštěno ve virtuálním emulátoru, který simuluje chování reálného PLC hardwaru. Komunikace pro výměnu signálů mezi řízením a simulovaným pracovištěm probíhá přes některou z virtuálních komunikační periferií jako je OPC, TCP/IP, nebo sdílená paměť. Hlavním cílem této metody je otestovat, ověřit funkčnost a odhalit chyby řídícího programu a HMI. Výhodou této metody je možnost jejího provedení v rámci jednoho zařízení.[11, 12]

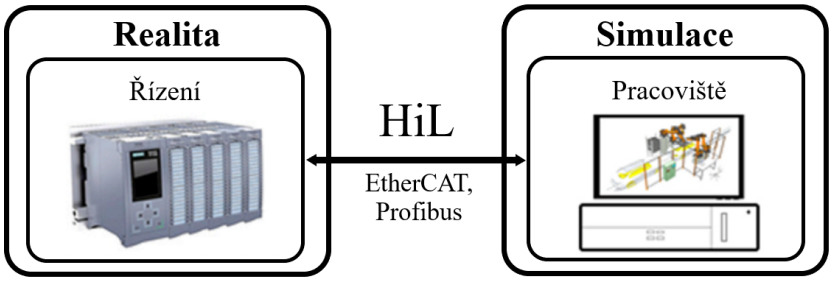


Obr. 8) Schéma konfigurace SiL [11]

3.4.3 Hardware in the loop

Hardware in the loop (HiL), tzv. hardware ve smyčce, je nejpokročilejší fáze virtuálního zprovoznění před zhotovením reálného robotického pracoviště. Tato metoda spočívá v nahrání zkompilovaných řídících programů do reálného PLC hardwaru, přičemž reálné PLC je pak propojeno se simulovaným virtuálním robotickým pracovištěm přes některou z průmyslových sběrnic (EtherCAT, Profinet, …) v uzavřené smyčce (viz obr. 9). Tento způsob testování vyžaduje vysoký výkon zařízení, na kterém je spuštěna simulace pracoviště, aby bylo zajištěno real-time zpracování všech řídících dat.[11, 12]

Výhodou této metody je možnost otestovat funkčnost řídících programů vytvořených ve virtuálním prostředí, nebo ověřit shodu chování mezi emulovaným a reálným hardwarem. HiL se často používá i pro testování parametrů PLC jako jsou minimální časové cykly, nebo využití sběrnice. Testuje se také reakce na simulované nebezpečné a poruchové stavy bez rizika poškození reálných robotů a jiných prvků. Otestovaný PLC hardware je následně možno přímo připojit k reálnému robotickému pracovišti. [11, 12]

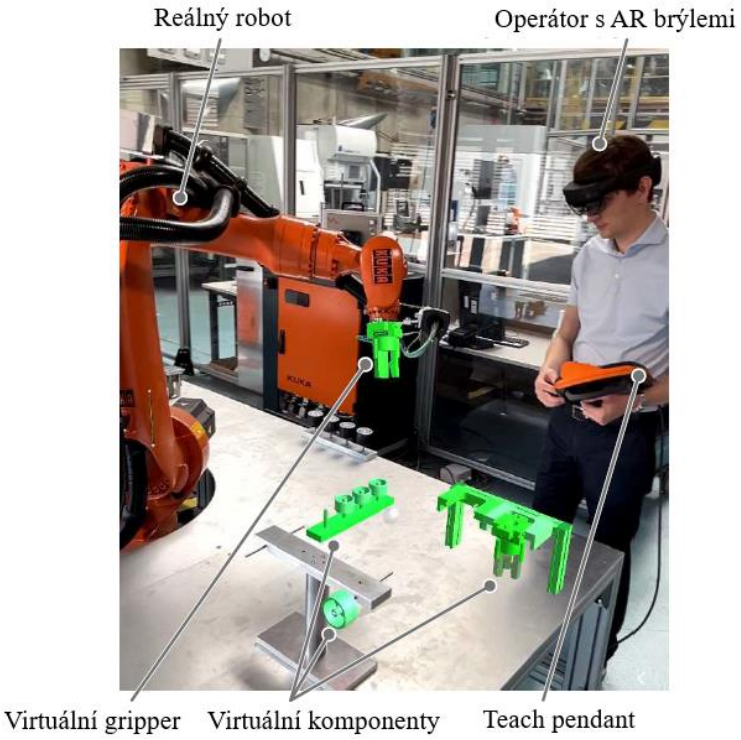


Obr. 9) Schéma konfigurace HiL [11]

3.4.4 Hybridní metody

Jako hybridní metody je možno označit přístupy, které jsou dle definice na pomezí virtuálního a reálného zprovoznění. První takovou metodou je Reality in the loop (RiL), realita ve smyčce, jejíž podstatou je propojení reálného pracoviště, nebo alespoň některých jeho částí, s virtuálním řídícím systémem. Souběžně je k tomuto procesu připojena počítačová simulace obsahující právě chybějící části pracoviště. Nabízí se tak možnost otestovat fyzické prvky pracoviště ještě před jeho úplným sestavením např. z důvodu prodlení dodání zbylých prvků.

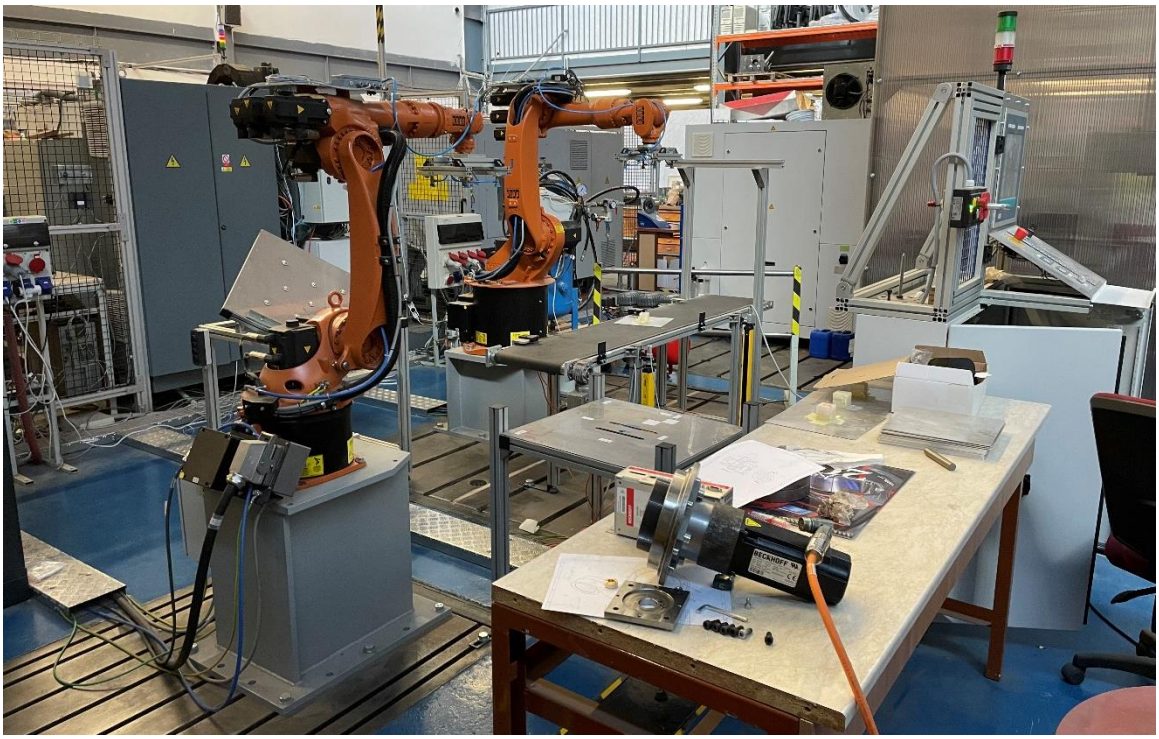
Společným limitujícím problémem všech zmíněných konfigurací virtuálního zprovoznění je spojení virtuální a reálné reprezentace součástí robotického pracoviště, z důvodu vizualizace 3D modelů digitálního dvojčete pracoviště a jejich 2D projekce na počítačové obrazovce. Řešením tohoto problému může být metoda Mixed reality in the loop simulation (MRiLS), která využívá brýle pro rozšířenou realitu (AR). Principem je vytvoření, resp. přenesení digitálního dvojčete pracoviště, nebo některých jeho částí do AR a následné propojení virtuálního řídícího systému s reálnými prvky pracoviště a AR vizualizací. Člověk provádějící toto testování je pak vybaven AR brýlemi a přímo integrován do testovacího procesu na reálném nedokončeném pracovišti, jehož chybějící části je možno pomocí těchto brýlí pozorovat (viz obr. 10). MRiLS však v průmyslu nebývá prozatím aplikována z důvodu její náročnosti a komplikacemi např. ve formě prodlevy mezi pohybem reálných prvků a AR reprezentací simulovaných prvků na ně navazujících. [11, 13, 14]



Obr. 10) Příklad využití MRiLS [14]

4 POPIS ŘEŠENÉHO PRACOVIŠTĚ

Předmětem této diplomové práce je virtuální zprovoznění výukového robotického pracoviště (viz obr. 11), které se nachází v areálu strojní fakulty Vysokého učení technického v Brně. Pracoviště je instalováno v hale C1, jejíž část zaujímají laboratoře ústavu výrobních strojů, systémů a robotiky. Pracoviště je využíváno zejména ve výuce studentů navazujícího magisterského programu na tomto ústavu v rámci cvičení, ale také k experimentům a závěrečným pracím.

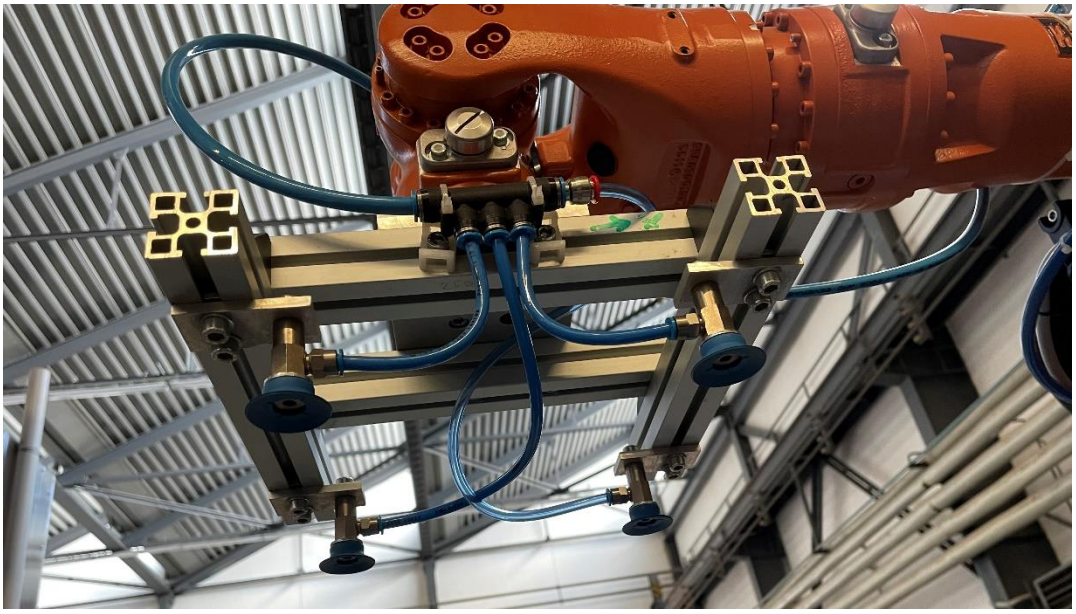


Obr. 11) Řešené robotické pracoviště

Hlavními prvky robotického pracoviště jsou dva průmyslové roboty německého výrobce KUKA AG (dále KUKA), konkrétně model KUKA KR 16-2 a model KUKA KR 5 arc. Jedná se o starší roboty, které výrobce k dnešnímu dni již nemá ve své nabídce. První z dvojice, robot KUKA KR 16-2, se vyznačuje jmenovitou nosností 16 kg, pracovní obálkou s maximálním poloměrem 1611 mm a jeho hmotnost je dle výrobce 235 kg. Umístěn je na ocelovém podstavci ve výšce 507 mm od podlahy. Druhý robot, KUKA KR 5 arc, disponuje jmenovitou nosností 5 kg, pracovní obálkou o maximálním poloměru 1530 mm a jeho hmotnost je dle výrobce 127 kg. Stejně jako předchozí robot je i tento umístěn na ocelovém podstavci, který dosahuje výšky 600 mm od podlahy pracoviště. Tento model byl původně výrobcem prodáván jako robot určený pro obloukové svařování, ale v rámci tohoto pracoviště je využíván především pro manipulační operace. [16, 17]

Dvojice robotů je osazena shodnými koncovými efektory (viz obr. 12) dle vlastního návrhu zdejšího ústavu. Konstrukce gripperů je tvořena extrudovanými hliníkovými profily od výrobce ALUTEC KK s. r. o. (dále ALUTEC), které jsou pospojovány šrouby. Hliníková konstrukce je osazena vakuovými přísavkami, které jsou obsluhovány podtlakovým ejektorem. Přísavky,

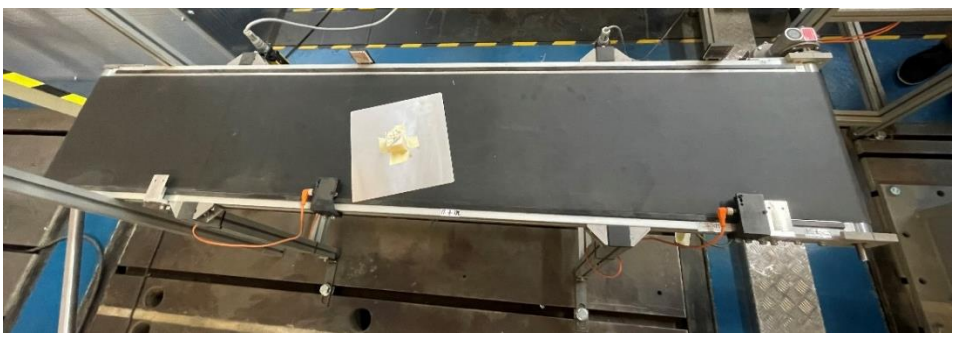
ejektor a veškeré pneumatické příslušenství jsou od výrobce Festo AG & Co. (dále Festo). Koncové efektory byly navrženy pro manipulaci hladkých čtvercových plechů o rozměrech 250 x 250 x 2 mm, přičemž umožnují manipulovat i plechy odlišných rozměrů. [15]



Obr. 12) Pohled na koncový efektor

Oba roboty jsou řízeny controllery s řídícím systémem KUKA KR C4, které jsou umístěny ve dvou skříních nacházejících se na okraji pracoviště. Rozměry každé skříně jsou 960 x 792 x 558 mm. Controllery využívají více jádrového procesoru, disponují SSD pevným diskem a rozhráním USB 3.0, GbE a DVI-I portem. Nezbytnou a samozřejmou součástí je také Teach pendant „smartPAD“ pro ruční ovládání a offline programování robotu, disponující souhlasným povelovacím zařízením a tlačítkem pro nouzové zastavení robotu. [18]

Součástí robotického pracoviště je také pásový dopravník DP50 (viz obr. 13) o délce 2000 mm a šířce 350 mm, který v současné konfiguraci pracoviště slouží k přesunu manipulovaných plechových kusů od robota KUKA KR 5 arc k robotu KUKA KR 16-2. Hnací válec dopravníku o průměru 50 mm je poháněn přes šnekovou převodovku VF30 A P63 B14 vyráběnou firmou Bonfiglioli S.P.A. asynchronním elektromotorem BN 63B 4 stejného výrobce. Pohon je řízen frekvenčním měničem Commander SK od společnosti Emerson Electric Co. Rám dopravníku je tvořen drážkovanými hliníkovými profily od výrobce ALUTEC. [15]

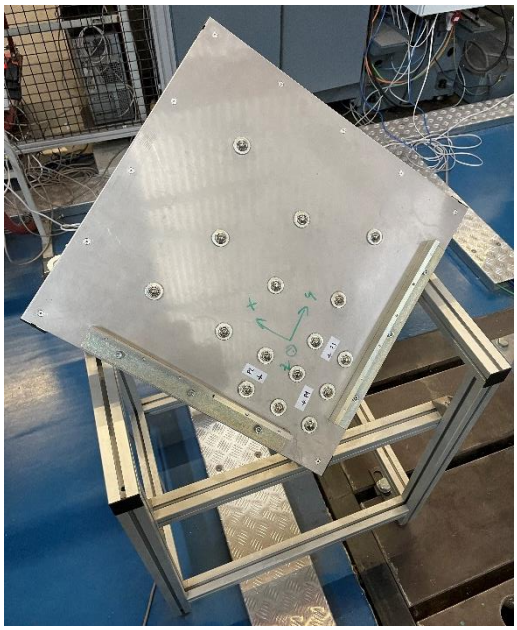


Obr. 13) Pásový dopravník DP 50

V rámci pracoviště jsou využívány dva přípravky. Prvním přípravkem je odkládací stůl (viz obr. 14), který plní funkci vstupní i výstupní pozice manipulační operace. Na odkládací ploše stolu o rozměrech 600 x 600 mm obsluha umístí plechy do stohu k pravému okraji plochy, které budou odebírány robotem, a na protější stranu budou robotem odkládány plechy, které prošly pracovištěm. Aby byla zachována dostatečná mezera mezi vstupním a výstupním stohem, maximální rozměr manipulovaných dílů nesmí přesáhnout 290 x 290 mm. V odkládací ploše je navíc vytvořeny dvě průchozí drážky pro osazení stolu indukčními snímači, které by indikovaly přítomnost plechů jak na vstupní, tak výstupní pozici.

Druhým přípravkem je skluz vybavený valivými hnízdy (viz obr. 15), který slouží k eliminování potenciálního chybného založení plechů obsluhou na vstupní pozici odkládacího stolu. Jakmile robot uvolní manipulovaný plech na přípravku, vlivem vlastní tíhy se díl přesune po nakloněné rovině s pomocí valivých elementů do spodní polohy, kde se zarazí o zábrany. Dojde tak k vycentrování plechu pro následné ideální uchopení koncovým efektorem robotu, protože je možno předem nadefinovat souřadnice středu dílů ve spodní poloze skluzu pro různé rozměry plechů.

Obr. 14) Odkládací stůl



Obr. 15) Skluz

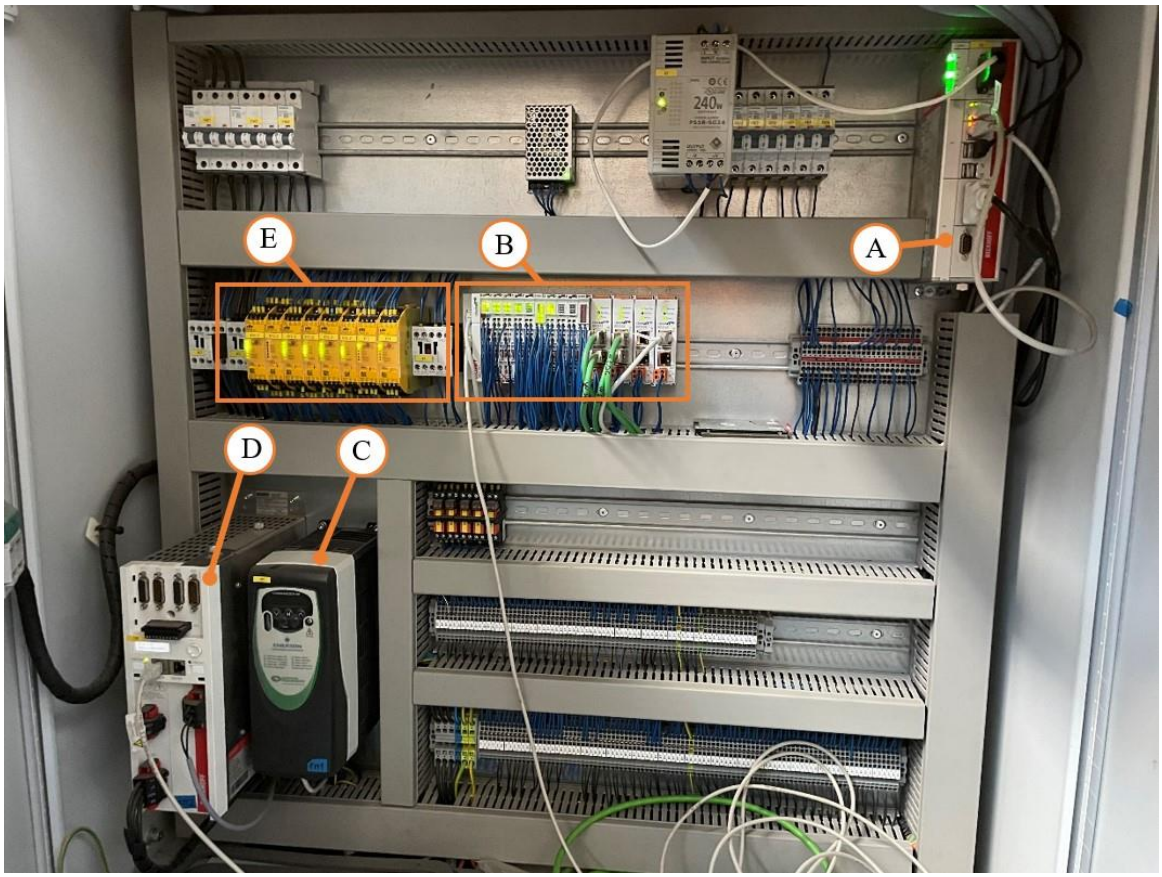
Pracoviště je také vybaveno bezpečnostními prvky, které zde slouží především pro ukázku, než pro zajištění bezpečnosti obsluhy. Prvním prvkem je bezpečnostní světelný závěs umístěný na dopravníku (viz obr. 11). Jedná se o závěs C4000 Standard, konkrétně vysílač C40S-0602FY010 a přijímač C40E-0602GY010 od výrobce SICK AG (dále jen SICK). Výška ochranného pole závěsu je 600 mm, maximální dosah až 21 m a rozlišení 20 mm – závěs by bylo tedy možno použít pro ochranu rukou obsluhy. Druhým bezpečnostním prvkem je multifunkční box zámku brány MGB od firmy EUCHNER electric s.r.o. (dále jen EUCHNER). Tento box v sobě kombinuje jištění, detekci otevření a zavření brány, dvě prosvětlená tlačítka a tlačítko nouzového zastavení. Dalšími bezpečnostními prvky jsou signalizační věž 8WD4408-0AA od společnosti Siemens AG (dále jen Siemens), která v současnosti není zapojena a tlačítko nouzového zastavení umístěné na ovládacím PLC panelu. [15]

Posledním prvkem pracoviště je ovládací panel Beckhoff CP6901-0001-0000 s rozváděčem (viz obr. 16). Vzhledem k tomu, že k řízení pracoviště bylo zvoleno PLC od firmy Beckhoff, prvním prvkem v rozváděči je industriální PC C6920-0040 (viz obr. 17) napájené 24 V zdrojem. To disponuje čtyřjádrovým procesorem Intel Core i7, 4 GB operační paměti a pevným diskem s 320 GB úložištěm. Beckhoff využívá pro komunikaci PLC s ostatními prvky vlastní průmyslovou sběrnici EtherCAT založenou na Ethernetu. Rozváděč je tedy dále osazen EtherCAT couplerem EK1100 s jednou kartou EL3064 pro čtyři analogové vstupy, jednou kartou EL4004 pro čtyři analogové výstupy, dvěma kartami EL 1008 po osmi digitálních vstupech, dvěma kartami EL2008 po osmi digitálních výstupech, dvěma kartami EL1809 po šestnácti digitálních vstupech a kartou EL2809 pro šestnáct digitálních výstupů. Coupler je navíc doplněn EtherCAT bridgy EL6692 (viz obr. 17). [15]

Obr. 16) Ovládací panel a rozváděč

Dalšími prvky nacházejícími se v rozváděči jsou měniče pro pohon dopravníku (viz obr. 17). Prvním z nich je Commander SK od společnosti Emerson Electric Co (viz. výše). Druhým měničem je servoměnič AX5203 od firmy Beckhoff. Jedná se o dvoukanálový měnič s kaskádovou řídící smyčkou vyvinutý přímo pro EtherCAT komunikaci.

Rozváděč navíc obsahuje ještě šest bezpečnostních relé PNOZ s7 a dvě bezpečnostní relé PNOZ s4 (viz. obr 17) od firmy PILZ GmbH & Co. KG (dále jen PILZ), jelikož bezpečnost pracoviště zde není řešena pomocí safety terminálů pro PLC, ale pouze mechanickým odpojením těchto relé. Každé relé je připojeno k jednomu z robotů a při odepnutí relé dojde k okamžitému zastavení robotů. [15]



Obr. 17) Rozváděč: A – industriální PC, B – EtherCAT Coupler s terminály, C – Frekvenční měnič, D – Servoměnič, E – Bezpečnostní relé

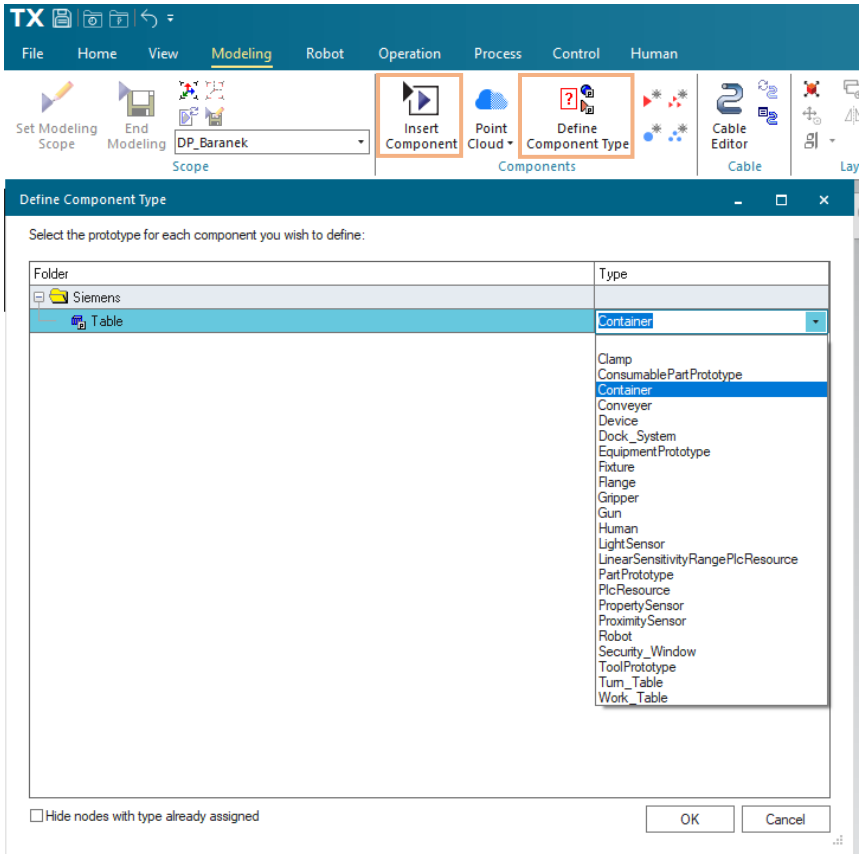
5 VIRT. ZPROVOZNĚNÍ V PROCESS SIMULATE

5.1 Vytvoření modelu pracoviště a definice komponent

Prvním krokem virtuálního zprovoznění v programu Process Simulate je vytvoření digitálního modelu pracoviště, který přesně odpovídá jeho reálné předloze. Modely robotů, dopravníku a přípravků nebylo nutné vytvářet znovu, protože mi byly poskytnuty od Ing. Mikuláše Szabari, Ph. D. v průběhu studia v rámci cvičení z předmětu „Navrhování a programování robotických pracovišť“. Tyto modely komponent nebylo třeba ani nijak upravovat a bylo možné je v této podobě importovat do prostředí programu.

Komponenty, které mají být importovaný do projektu, musí být uloženy v souboru s příponou .jt (např. Conveyor.jt) a model v tomto formátu uložen ve stejnojmenné složce s příponou .cojt (např. Conveyor.cojt). Následně je potřeba takto připravenou složku uložit do kořenového adresáře programu Procces Simulate, jehož výchozí cesta je „C:\Siemens“.

V prostředí programu je před importem komponenty nejprve potřeba definovat její typ pomocí *Define Component Type* v záložce *Modelling*. Dialogové okno vyzve k vyhledání umístění požadované komponenty v adresáři a následně je nutno vybrat některý z předdefinovaných typů součástí podle její funkce. Pro příklad při definici dopravníku se zvolí typ komponenty *Conveyor*, pro definici koncového efektoru *Gripper* atp. Takto definovaný model komponenty je poté možno importovat pomocí *Insert Component* nacházejícího se opět v záložce *Modelling* (viz obr. 18).

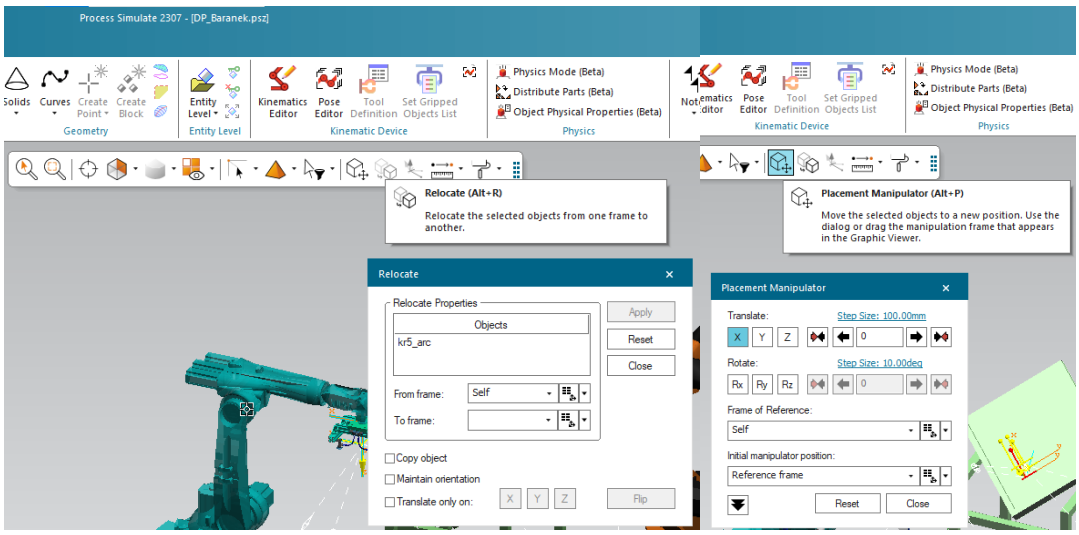


Obr. 18) Import a definice komponenty

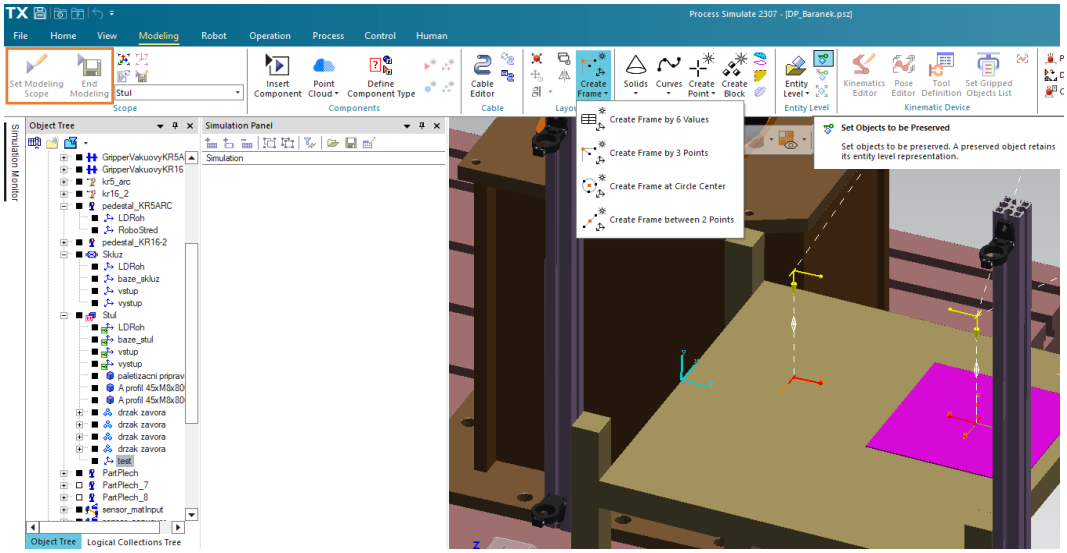
Importované komponentě je nutno definovat její polohu v prostředí programu. První importovanou součástí byla podlahová deska, nacházející se pod robotem KR 5 arc, protože k ní byly vztaženy ostatní naměřené rozměry layoutu. Přemístění objektu v rámci pracoviště je možné pomocí jeho vybrání a použití funkce Relocate, resp. Placement Manipulator, které se nacházejí v liště u horního okraje grafického prostředí (viz obr. 19), nebo je možné je vyvolat klávesou zkratkou ALT + R, respektive ALT + P. Veškeré modely byly takto rozmístěny tak, aby jejich vzájemná poloha odpovídala skutečnému rozložení pracoviště.

Modelům stolu, dopravníku a skluzu je výhodné vytvořit body, tzv. *Frame*, které jsou vázané k danému modelu a i při jeho přemístění jsou přemístěny s ním. Tyto body následně značně zjednodušší další práci jako je definice bodů trajektorie robotické operace atp. Konkrétně u dopravníku a přípravků vytvoříme body, kde se bude nacházet pomyslný střed manipulovanoného plechu v moment, kdy s ním bude robot, resp. koncový efektor interagovat. *Frame* komponenty je možné vytvořit pouze pokud se komponenta nachází v tzv. rozmodelovaném stavu. Toho je možno docílit jejím označením a kliknutím na *Set Modeling Scope* v záložce *Modeling*. Ve stejné záložce se nachází příkaz *Create Frame,* po jehož rozbalení je možno vybrat ze čtyř různých způsobů definice bodu. Aby bylo následně takto vytvořený bod možno použít, je nutno jej označit a použít příkaz *Set Objects to be Preserved*. Takto vytvořené body dále možno přesunout stejně jako jiné objekty v prostředí (viz výše). Po zadefinování všech požadovaných bodů pro danou součást je proces ukončen příkazem *End Modeling*. Tento proces je vyobrazen na obr. 20.

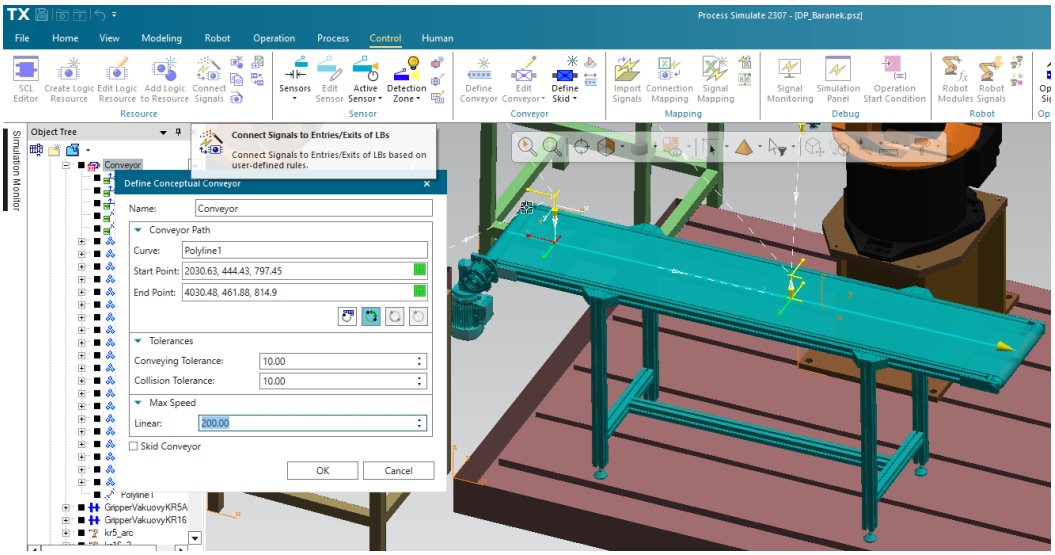
Aby byl dopravník v průběhu simulace funkční, musí se nejprve provést jeho definice, což je opět možné pouze v rozmodelovaném stavu (viz výše). Prvním krokem je vyznačení trajektorie, po které se bude dopravovaný předmět pohybovat. Trajektorii je možno vytvořit funkcí *Polyline* nacházející se v příkaze *Curves* v záložce *Modeling*. Druhým krokem je samotná definice dopravníku. Výběrem, resp. označením dopravníku se zpřístupní funkce *Define Conveyor*, která se nachází v kartě *Control* v oddíle *Conveyor*. V dialogovém okně je následně jako dráha dopravníku zvolena trajektorie vytvořená v prvním kroku, dále je možno definovat směr pohybu, nastavit tolerance, nebo určit maximální dovolenou rychlost dopravníku (viz obr. 21).



Obr. 19) Přemístění objektu



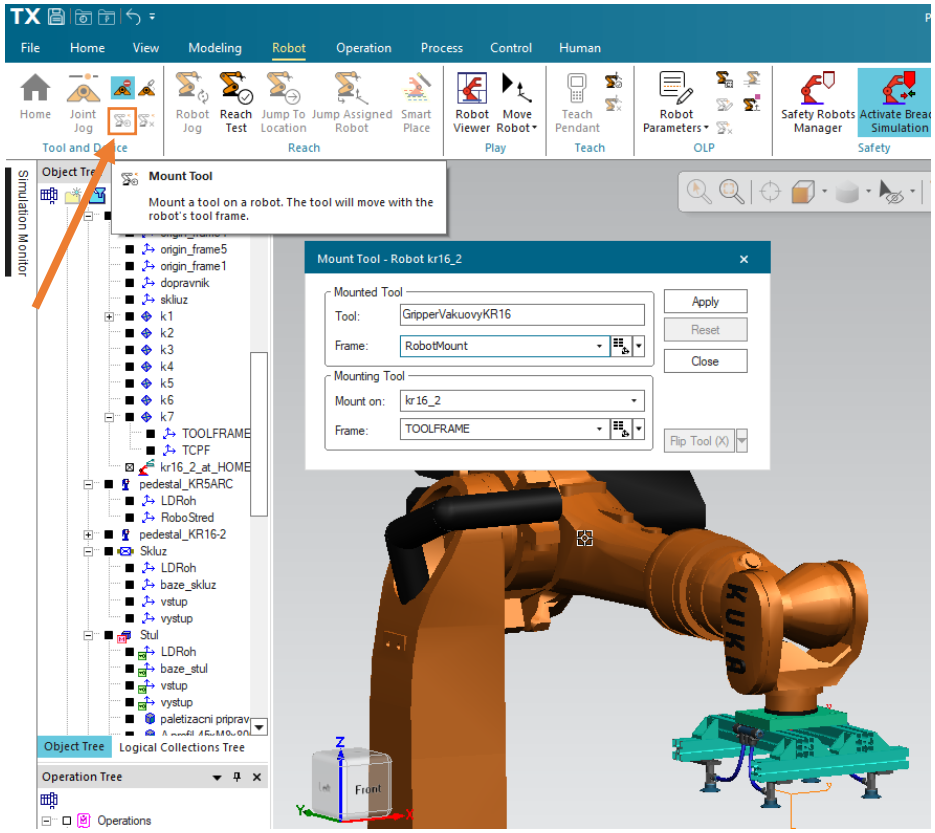
Obr. 20) Tvorba bodů komponenty



Obr. 21) Definice dopravníku

Oběma robotům je potřeba přiřadit nástroj, resp. koncový efektor. Prvním podstatným krokem pro jeho správnou funkčnost je, aby při importu modelu nástroje byl definován jako *Gripper* (viz výše). Druhou podmínkou je přítomnost minimálně dvou *Frame* bodů v modelu. Jeden bod, ve kterém bude koncový efektor spojen s přírubou robotu a druhý bod, zvaný TCP (Tool Center Point). Definici TCP bodu gripperu je vhodné věnovat zvýšenou pozornost, protože vůči jemu jsou následně počítány trajektorie robotických operací a při jeho špatné definici by mohlo dojít ke kolizi gripperu s ostatními komponenty na pracovišti. Definici těchto bodů je možné provést stejně jako u ostatních součástí (viz výše, obr. 21). V tomto případě se TCP bod u obou efektorů nachází ve středu roviny, která je kolmá na konce přísavek. Samotné pojení gripperu s robotem je provedeno označením robotu a příkazem *Mount Tool* nacházejícím

se v záložce *Robot* nebo vybráním tohoto příkazu z nabídky vyvolané pravým tlačítkem myši na robota. Po otevření dialogového okna program vyzve k vybrání konkrétního gripperu, který má být k robotu připojen, a k vybrání bodů, ve kterých má ke spojení dojít (viz obr. 22).



Obr. 22) Připojení efektoru k robotu

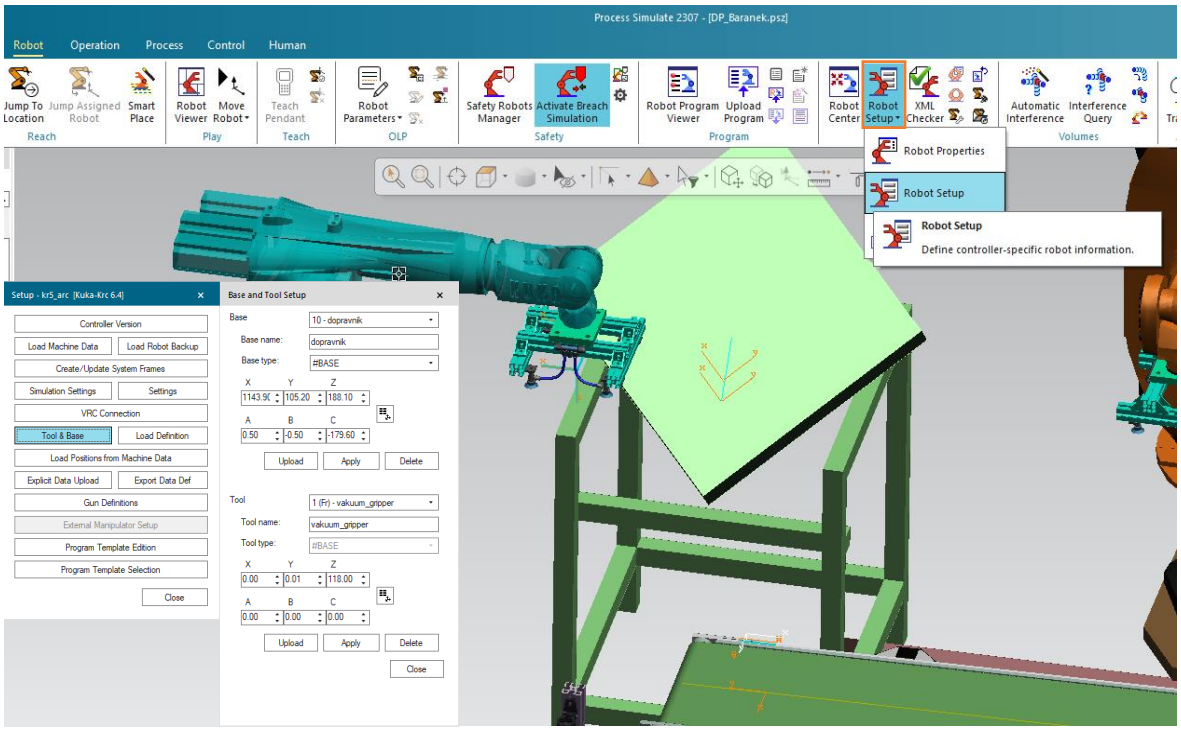
Aby bylo možné s roboty dále pracovat a vytvořit robotické programy, musí být každému robotu přiřazen robotický controller. Definice controller se provádí pomocí funkce *Controller Settings*, která je součástí nabídky *Robot Setup* v kartě *Robot*. Pokud byly roboty při importu do programu správně zadefinovány jako „Robot“ (viz výše), zobrazí se v otevřeném dialogovém oknu jejich názvy. Oběma robotům je zde možno určit controller, jeho verzi, popřípadě *Motion Planner*, který určuje to, zda-li robot ovládá samotná simulace v Process Simulate, nebo externě emulovaným virtuálním controllerem. V tomto případě je vybrán controller Kuka-Krc a jeho verze 8.3. Verze nastavená zde musí odpovídat verzi controlleru reálného pracoviště. Nastavení pro tento projekt je vidno na obr. 24. V případě, že by v dialogovém okně nebylo možné vybrat žádný controller, nebo by se v nabídce nenacházela možnost požadovaného controlleru, program Process Simulate jím nedisponuje a je potřeba jej doplnit. Balíčky controllerů všech předních výrobců robotů je možné přidat v průběhu instalace programu, nebo je nainstalovat dodatečně.

Posledním krokem přípravy před tvorbou robotických operací je nastavení souřadných bází a nástrojů oběma robotům. Souřadnice těchto bodů musí odpovídat jejich definici v robotických controllerech na reálném pracovišti, kde jsou již definovány a jejich hodnoty je možné zobrazit, nebo změnit pomocí teach pendantu. Souřadnice všech bodů v reálných

controllerech jsou vypsány v Tab. 1. Jejich definice v programu se provede po výběru jednoho z robotů funkcí Robot Setup v kartě Robot. Ve vyvolaném oknu se provede výběr možnosti Tool & Base, kde je již možno nastavit konkrétní hodnoty (viz obr. 23).

Tab 1) Reálné souřadnice bází a nástrojů

KUKA KR 5 arc	Číslo nástroje	X	Y	Z	A (Rz)	B (Ry)	C (Rx)
vakuum_gripper	1	0	0	118	0	0	0
KUKA KR 16-2	Číslo nástroje	X	Y	Z	A (Rz)	B (Ry)	C (Rx)
vakuum_gripper	1	0	0	115	0	0	0
KUKA KR 5 arc	Číslo báze	X	Y	Z	A (Rz)	B (Ry)	C (Rx)
dopravnik	10	1143,9	105,2	188,1	0,5	-0,5	-179,6
skluz	11	898,6	931,4	159,4	144,8	-30,3	145,6
stul	12	933,9	-572,1	104,1	178,8	-0,6	-1,1
KUKA KR 16-2	Číslo báze	X	Y	Z	A (Rz)	B (Ry)	C (Rx)
dopravnik	10	837,9	-1027,9	289,5	90,1	0,1	-179,5
skluz	11	11,5	-1268,4	255,4	-126,1	-30,8	144,8

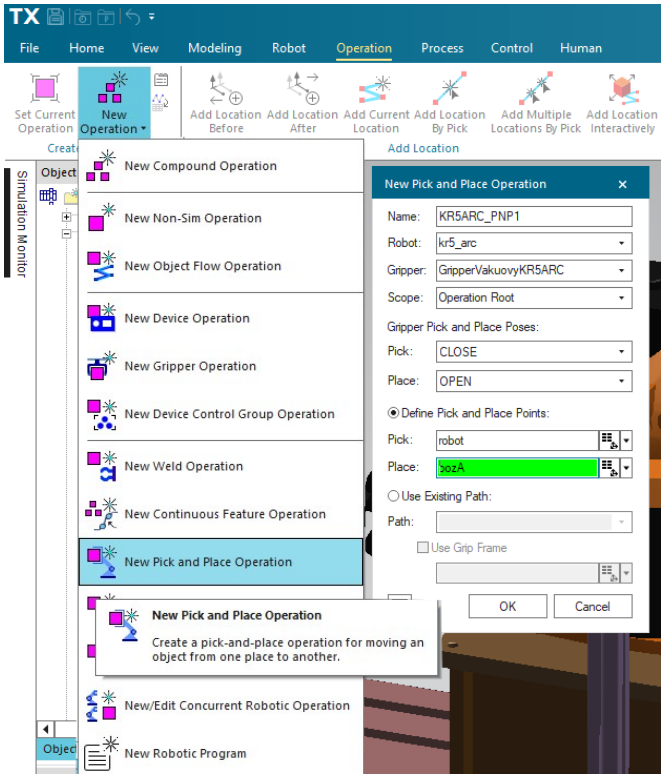


Obr. 23) Definice bází a nástrojů robotu

5.2 Robotické a simulační operace

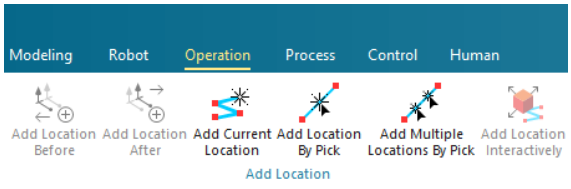
Modelovou úlohou řešenou v této práci je jednoduchá sekvence – první robot přemístí plech ze stolu na dopravník, druhý robot následně přesune plech z dopravníku na skluzový přípravek, kde se plech vycentruje a je přemístěn prvním robotem zpátky na stůl. Jedná se tedy o dvě Pick and Place operace pro robot KR 5 arc a jednu Pick and Place operaci pro robot KR 16-2.

Nová operace je vytvořena pomocí funkce *New Pick and Place Operation* nacházející se v nabídce *New Operation* v kartě *Operation*. Následně je v dialogovém okně potřeba vyplnit název vytvářené operace, přiřadit robot, který ji bude vykonávat, koncový efektor a vybrat body uchopení, resp. uvolnění součásti. Program následně vygeneruje holou operaci v panelu *Operation Tree*. Proces je znázorněn na obr. 24.



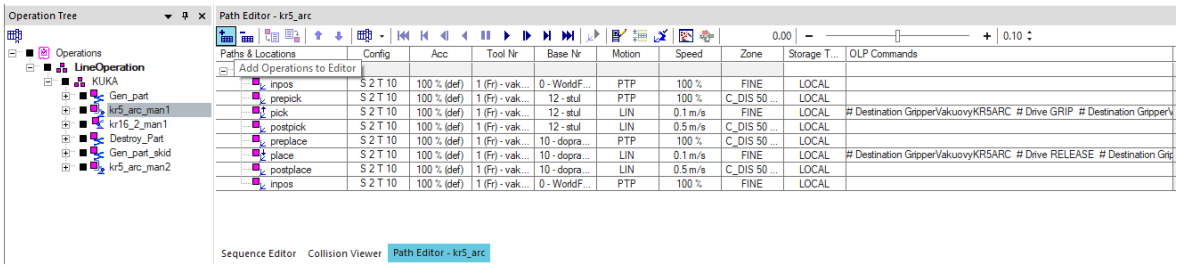
Obr. 24) Pick and Place operace

Nově vytvořenou operaci je potřeba ještě upravit a doplnit jí další parametry. První úpravou bylo přidání dalších bodů trajektorie, protože výchozí operace obsahuje pouze dva body. Přidání bodů je možné po označení operace v *Operation Tree* a výběrem některé z funkcí v kartě *Operation* v sekci *Add Location* (viz obr. 25). Byly přidány body výchozí pozice a body před, resp. po uchopení a položení manipulovaného plechu.



Obr. 25) Přidání bodu trajektorie

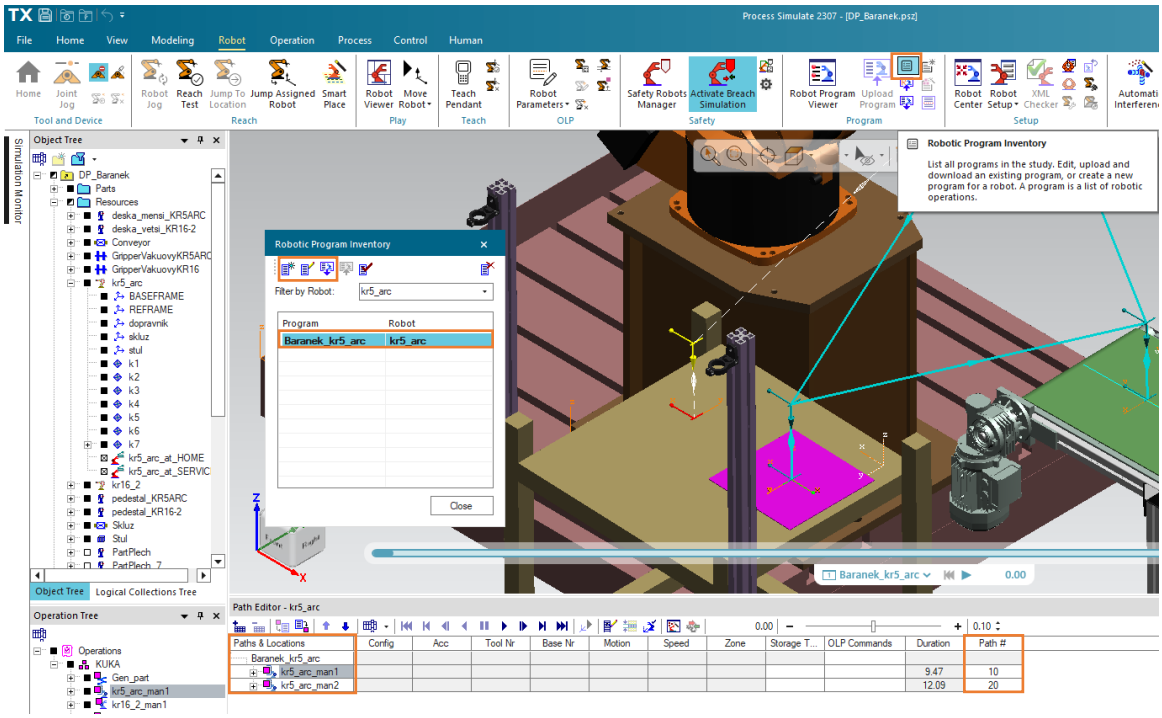
V základním programovém rozvržení se u spodního okraje obrazovky nachází panel *Path Editor*. Přetažením operace do tohoto okna, nebo kliknutím na tlačítko se symbolem „+“ při označené operaci v *Operation Tree* dojde k jejímu otevření v editoru. Zde jsou pak po řádcích vypsány jednotlivé body trajektorie operace, které je možno odebrat nebo měnit jejich posloupnost. Každému z nich je možno přiřadit definice jako je druh pohybu, rychlost, nastavení zóny, nástroje, báze, robotická konfigurace nebo OLP příkazy. Pokud některé z požadovaných atribut v editoru chybí, nebo naopak přebývají, pomocí příkazu *Customize Columns* je možné tuto nabídku upravit. Po nastavení všech parametrů každého z bodů operace je možno zapnout funkci *Auto Teach* a spustit simulaci. Tato funkce automaticky nastaví nejvhodnější robotickou konfiguraci pro dosažení každého bodu. V případě nespokojenosti lze danou konfiguraci v editoru ručně změnit. Tento proces je znázorněn na obr. 26.



Obr. 26) Path Editor

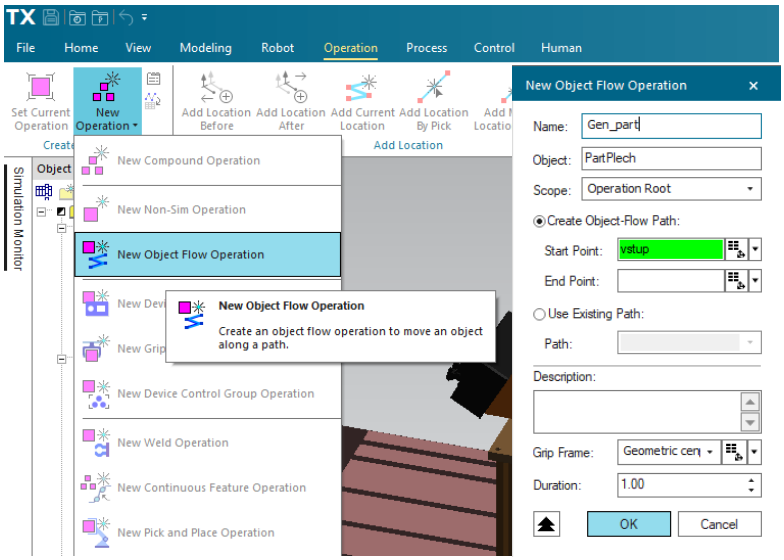
Vytvořené robotické operace je následně nutné nahrát do paměti virtuálního robotu. To je možné po vybrání příslušného robotu provést v oknu *Robotic Program Inventory*, které se nachází na hlavním panelu v záložce *Robot*, nebo v nabídce vyvolané kliknutím pravým tlačítkem myši na robot. V otevřeném oknu je nejprve příkazem *Create New Program* vytvořen nový program v paměti, do kterého lze následně přidat dříve vytvořené robotické operace. Označením vytvořeného programu se v okně přístupní funkce *Open in Path Editor*, jejímž vybráním se objeví v *Path Editoru* při spodním okraji obrazovky prázdný program.

Do tohoto prázdného programu je následně možno ze stromu *Operation Tree* přidat přetažením myší robotické operace náležící vybranému robotu. Následně je zásadní každé operaci přiřadit číslo ve sloupci „Path #“. Pod tímto číslem je program, resp. operace uložená v paměti robotu a pomocí signálu, resp. proměnné datového typu „byte“ načtena pro následné vykonání. Takto vytvořený program je následně možné vyexportovat pro reálný robot příkazem *Download to Robot* v okně *Robotic Program Inventory*. Celý postup nahrání robotických programů je znázorněn na obr. 27.



Obr. 27) Nahrání robotických programů

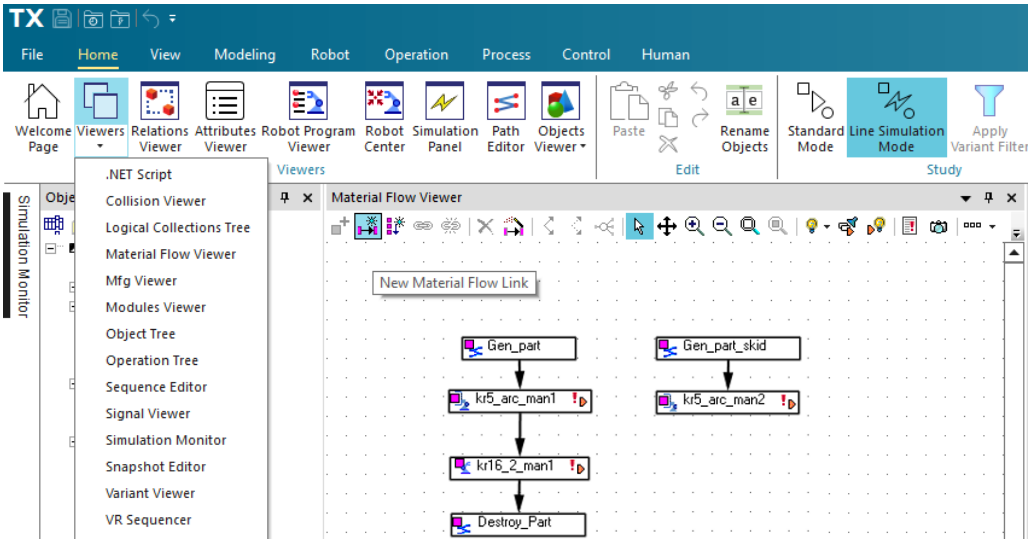
Aby bylo možné provést simulaci celé pracovní sekvence, musí být kromě robotických operací vytvořeny i operace, které budou simulovat vygenerování, resp. zničení manipulovaného dílu. Pro tento účel je vhodné použít typ operace *Object Flow Operation*. Tu je možno vytvořit pomocí příkazu *New Object Flow Operation* v nabídce *New Operation* v záložce hlavního panelu *Operation*. Následně je v rámci dialogového okna (viz obr. 28) potřeba vyplnit název operace a zvolit objekt, jehož se bude daná operace týkat. Pokud je tento druh operace využíván pro generování, resp. zničení součásti, vyplňuje se pouze jeden bod trajektorie, a to ten počáteční. Tímto bodem je poloha, ve které má být akce provedena.



Obr. 28) Object Flow operace

5.3 Materiálový tok

Vytvoření materiálového toku je klíčovým prvkem pro správný chod simulace. Zpřístupnění editoru materiálového toku je podmíněno přepnutím projektu, resp. studie, ze stavu *Standard Mode* na *Line Simulation Mode.* Tyto módy je možno měnit v kartě *Home.* Okno materiálového toku *Material viewer* se nachází v záložce *Home* pod nabídkou *Viewers.* Po otevření by se měly v tomto okně nacházet bloky reprezentující veškeré operace vytvořené v rámci projektu. Pokud tomu tak není, stačí myší přesunout ze stromu operací chybějící operaci. Propojením bloků operací mezi sebou pomocí funkce *New Material Flow Link* definuje počátek reprezentace manipulovaného materiálu v simulaci a jeho odstranění na konci řetězce propojených operací. Nastavení materiálového toku v této práci znázorňuje obr. 29.



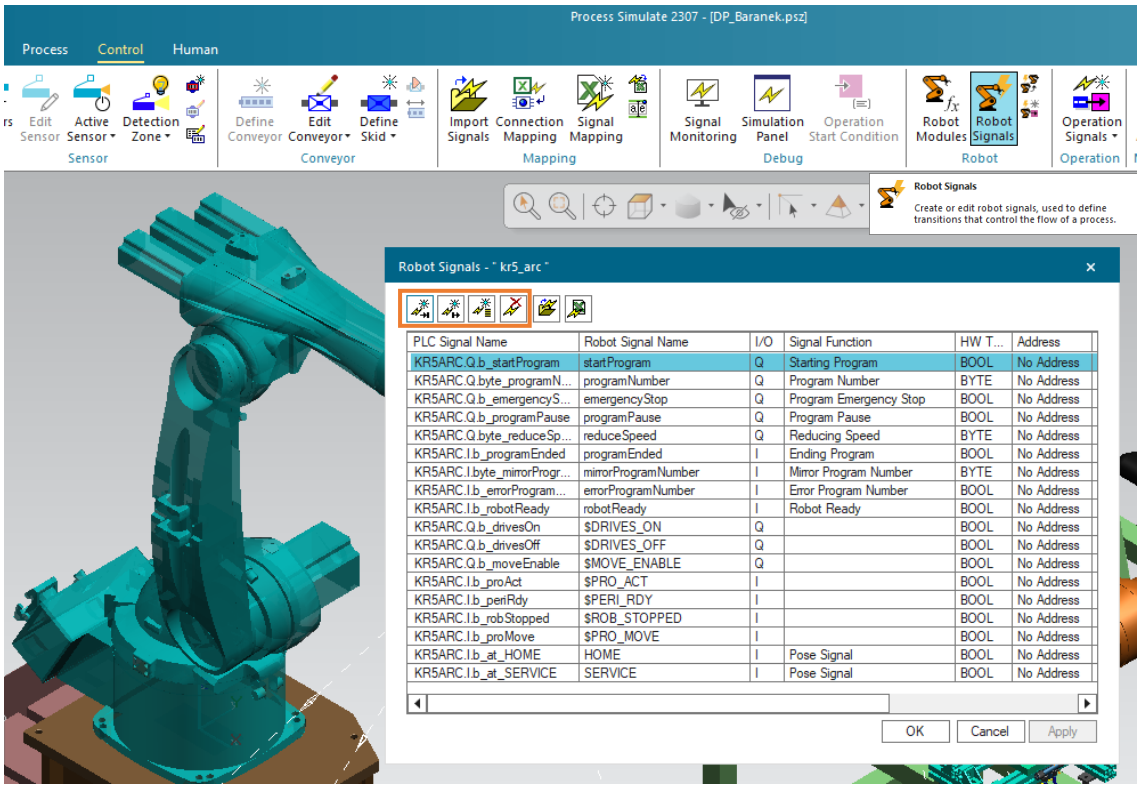
Obr. 29) Materiálový tok

5.4 Signály simulace

Stejně jako je reálné pracoviště řízeno pomocí PLC na základě signálů, bude jimi řízena i jeho virtuální simulace. Jako první byly vytvořeny robotické signály. Konvence v Process Simuate je taková, že na signály se nahlíží z pohledu PLC, což může být při tvorbě robotických signálů matoucí. Pokud má být definován výstupní signál robotu, v programu vytváříme signál vstupní a naopak.

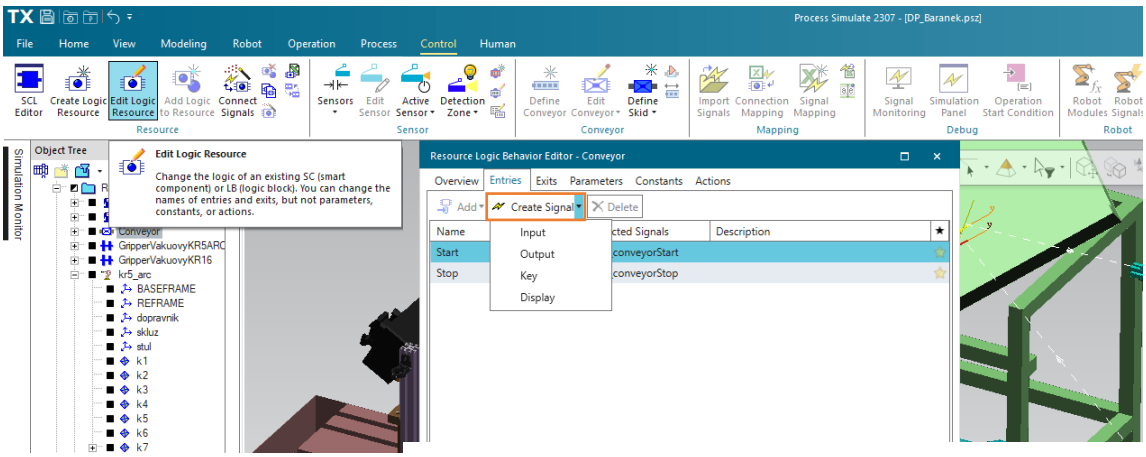
Vybráním robotu se zpřístupní funkce *Robot Signals* v kartě *Control,* jejímž použitím dojde k otevření okna pro správu robotických signálů, kde jsou pomocí možnosti *Create Default Signals* vytvořeny výchozí vstupní a výstupní signály definované v robotickém controlleru programu. Kromě výchozích signálů je možno v případě potřeby robotu vytvořit další signály pomocí *New Input Signal*,resp. *New Output Signal.*

V tomto případě byly mimo výchozích signálů navíc dodatečně vytvořeny některé signály odpovídající signálům použitelných v rámci „Automatic External“ řídícím módu reálného robotu podle oficiální příručky pro KUKA Systém Software 8.3 pro intergrátory. Oběma robotům byly nadefinovány shodné signály a jejich výčet je na obr. 30.



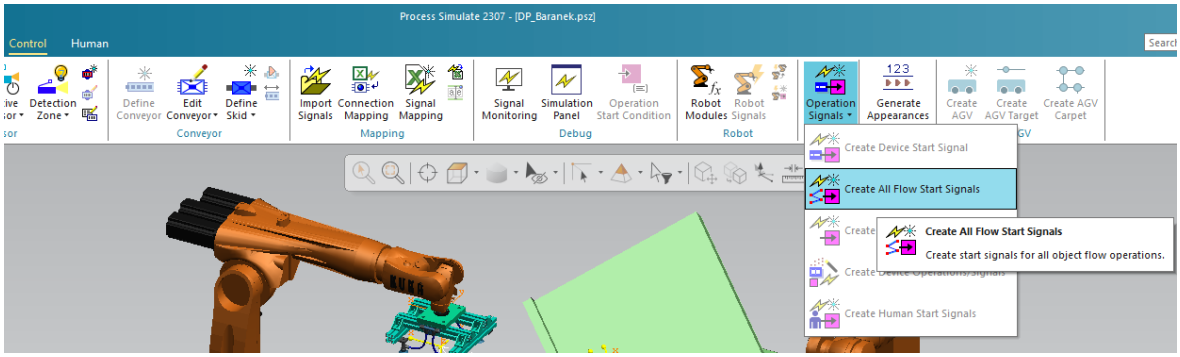
Obr. 30) Robotické signály robotu KR 5 arc

Signály byly vytvořeny i pro ovládání dopravníku. Nejprve je potřeba dopravníku definovat akce, které budou ovládány. To je možno provést označením a vybráním funkce *Edit Conveyor Logic Behaviour*, která se nachází pod tlačítkem *Edit Conveyor* v kartě *Control*. V nově otevřeném okně je možno vybrat ze čtyř řiditelných akcí dopravníku – zapnutí pohonu, vypnutí pohonu, změna rychlosti a změna směru, přičemž pro účely této simulace jsou dostačující první dvě akce. Následně byly přiřazeny signály pro ovládání těchto akcí. Funkce *Edit Logic Resource* v kartě *Control* při označeném dopravníku otevře editor logiky součásti. V záložce *Entries* jsou vypsány definované ovladatelné akce, pro které je možno vytvořit a přiřadit ovládací výstupní signál (viz obr. 31).



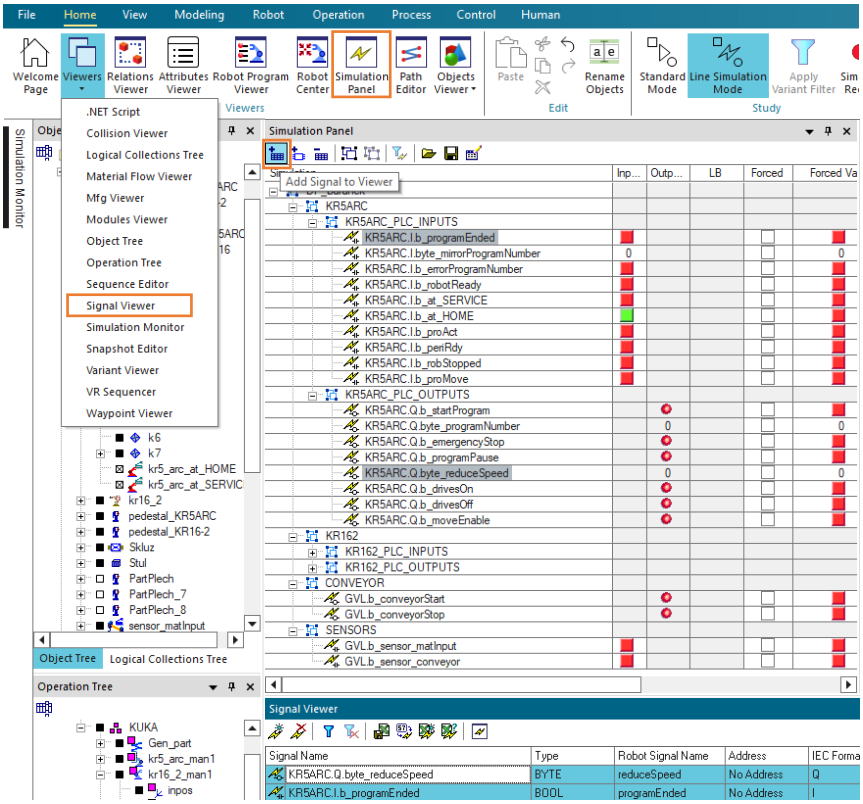
Obr. 31) Ovládací signály dopravníku

Zbylé signály simulace souvisí s použitými senzory a operacemi pro materiálový tok. Signály, které jsou spínány senzory jsou vytvořeny automaticky společně s přidáním senzoru na pracoviště. Signály pro spouštění operací materiálového toku z PLC je možno vytvořit příkazem *Create All Flow Start Signals* z nabídky *Operation Signals* v kartě *Control* (viz obr. 32).



Obr. 32) Signály Flow operací

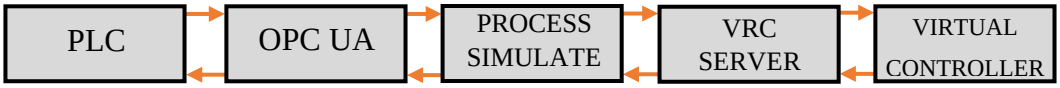
Všechny signály simulace jsou viditelné v rámci okna *Signal Viewer*, které je možno zobrazit v nabídce *Viewers* v záložce *Home*. Stav signálů je v průběhu simulace možno monitorovat pomocí funkce *Simulation Panel* v kartě *Home*, což značně usnadní následný debugging a celkovou práci se signály. Přidání signálů do tohoto okna se provede jejich označením v *Signal Vieweru* a tlačítkem *Add Signal to Viewer*. Signály je možno spojit do skupin a vytvořit strukturu pro větší přehlednost. Tento postup je názorný z obr. 33.



Obr. 33) Simulation Panel

5.5 Simulace virtuálního robotického controlleru

Ideální konfigurace komunikační smyčky pro virtuálního zprovoznění by v případě tohoto projektu měla sestávat ze vzájemné výměny proměnných v reálném čase mezi simulací v Process Simulate, PLC řízením v TwinCAT a virtuálním robotickým controllerem. Topologie ideální smyčky je na obr. 34. Virtuální robotický controller je emulátor reálného controlleru na pracovišti. Hlavní výhody zapojení virtuálního robotického controlleru do simulační smyčky jsou získání odezvy robotu na signály z řídícího PLC, zpětná vazba ve formě výstupních signálu o stavu robotu a případné chybové hlášky při špatném postupu.

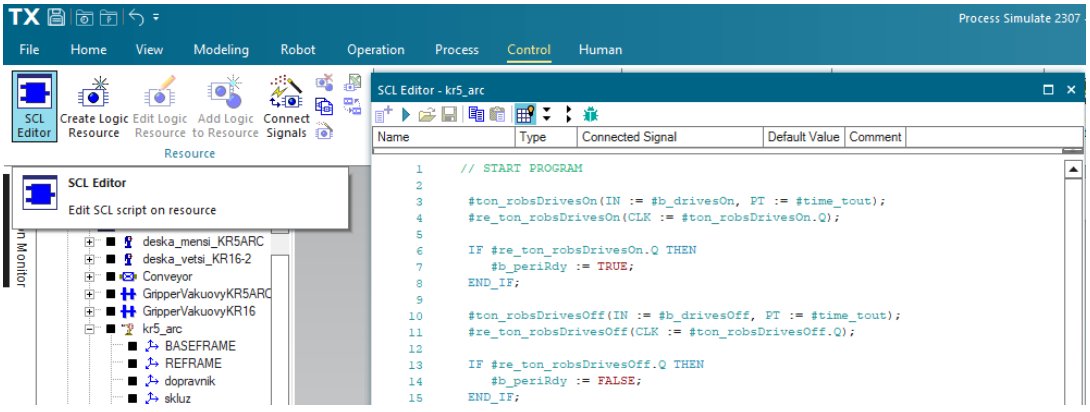


Obr. 34) Ideální smyčka pro VZ

Software pro emulování virtuálních controllerů robotů výrobce KUKA se jmenuje KUKA.OfficeLite a je možné jej propojit s programem Process Simulate pomocí tzv. *VRC Serveru*, který umožňuje komunikaci v reálném čase. Společnost KUKA v době řešení této diplomové práce nenabízí bezplatné edukační licence na využívání programu pro studenty. Z tohoto důvodu nebylo bohužel možné pro simulaci využít originální virtuální controller. [19]

Aby bylo možné během simulace získávat zpětnou vazbu o stavu robotů (např. stav motorů), bylo nutné v Process Simulate vytvořit zjednodušenou náhradu virtuálního controlleru. Ovládání výstupních signálů robotu je v rámci signálově řízené simulace možné pouze dvěma způsoby – vytvořením logických bloků, nebo vytvoření SCL (Structured Control Language) skriptu pro každý robot.

Kvůli jednoduššímu provedení a větší variabilitě byla vybrána možnost vytvoření SCL skriptů. Prostředí pro programování sdílí některé prvky a konvence s prostředím TIA Portal od stejné společnosti. Tyto skripty jsou během simulace cyklicky vyhodnocovány podobně jako PLC. Okno editoru pro tvorbu programového skriptu je možné otevřít označením robotu a následném vybráním funkce *SCL Editor* v záložce *Control* na hlavním panelu. Robot se při psaní skriptu musí nacházet v rozmodelovaném stavu (viz např. kap. 5.1). Postup otevření okna SCL Editor a ukázka části skriptu jednoho z robotů je na obr. 35.



Obr. 35) SCL Editor

6 KOMUNIKACE MEZI PROCESS SIMULATE A PLC

Vytvoření komunikace mezi simulačním programem Process Simulate a prostředím emulátoru PLC TwinCAT 3, který slouží i k programování řízení, je základní předpoklad pro výměnu řídících signálů. Toto propojení umožňuje v reálném čase testovat odezvu simulace, reprezentující chování reálného pracoviště, na řídící pokyny a stejně tak v obráceném smyslu získávat zpětnou vazbu o stavu pracoviště pro PLC.

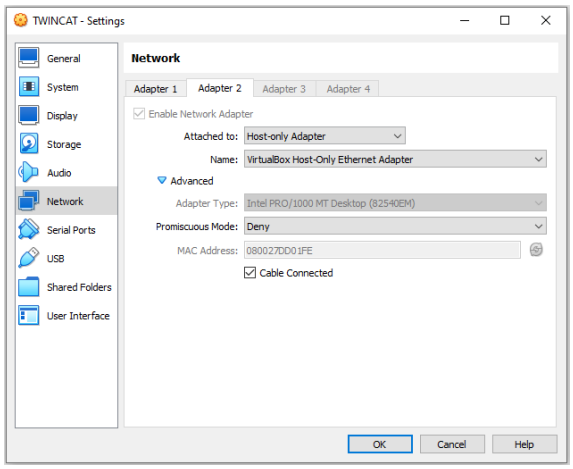
Process Simulate umožňuje připojení externího PLC několika způsoby. Process Simulate je vyvíjený společností Siemens, z tohoto důvodu je komunikace s PLC tohoto výrobce implementována přímo v programu. Nativní podpora vytvoření komunikace s programem TwinCAT neexistuje, a proto bylo využito komunikace prostřednictvím OPC UA.

OPC UA je zkratkou pro Open Platform Communications Unified Architecture. Jedná se o komunikační standard pro spolehlivou a zabezpečenou výměnu dat využívanou především na poli automatizace na principu server – client. Funguje nezávisle na platformě a je možno jej používat na většině výpočetního hardware včetně PLC nebo microcontrolllerů. Realtime komunikace využívá binární protokol *opc.tcp://Server*, který bude využit i pro propojení Process Simulate s TwinCAT. [20]

6.1 Příprava pro vytvoření serveru

OPC UA server bývá typicky hostován PLC a stejný přístup byl zvolen i v případě této práce. Společnost Beckhoff v rámci svých funkcí rozšiřujících balíčků „Connectivity“ pro TwinCAT 3 nabízí zdarma ke stažení balíček „TF6100 | TwinCAT 3 OPC UA“, který je třeba pro další postup nainstalovat.

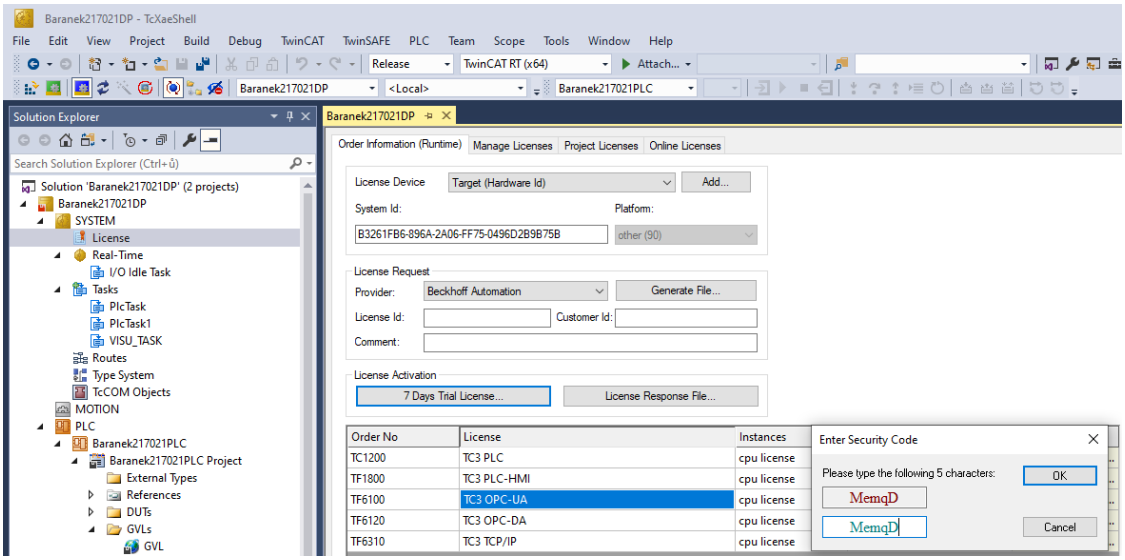
Pokud je pro práci se softwarem TwinCAT využívána virtuální instance počítače pomocí některého z virtualizačních nástrojů jako je Oracle VM VirtualBox (využit i pro tuto práci), je vhodné se nejprve ujistit, zda-li je možné s virtuálním zařízením komunikovat z hostujícího počítače. Jednou z možností je využití příkazového řádku ve Windows a příkazu *ping* ve spojení s IP adresou virtuálního počítače. Pokud by zařízení nebylo dosažitelné a je využíván VirtualBox, je problém možno vyřešit přidáním nového síťového adaptéru v nastavení virtuálního zařízení (viz obr. 36).



Obr. 36) Přidání síťového adaptéru

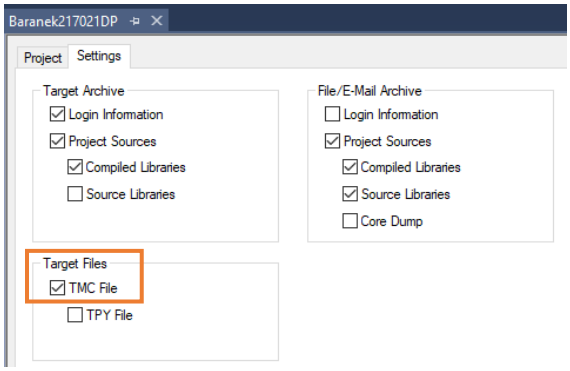
6.2 Vytvoření OPC UA serveru

Prvním krokem po založení nového TwinCAT projektu s PLC je aktivování licence pro rozšiřující balíček TF6100 s OPC UA funkcemi. Licence projektu je možno spravovat kliknutím na *License* v *Solution Exploreru* na levém kraji obrazovky. Následně je potřeba v záložce *Manage Licenses* vyhledat licenci „TC3 OPC-UA“ a zaškrtnout políčko *cpu license*. Licenci je následně nutné aktivovat v kartě *Order Information (Runtime)* jejím vybráním a zvolením sedmidenní zkušební varianty (viz obr. 37). Licence je možné po uplynutí sedmi dnů opět zdarma obnovit stejným způsobem. Po aktivování licenci je doporučeno restartovat zařízení.



Obr. 37) Aktivování licencí

Následuje nastavení PLC projektu. Nejdříve je vhodné aktivovat možnost *Autostart Boot Project* kliknutím pravým tlačítkem na PLC projekt v *Solution Exploreru*. V záložce *Settings* je dále třeba povolit přístup k TMC (TwinCAT Module Class) souborům (viz obr. 38), které obsahují informace potřebné k tomu, aby bylo možno číst proměnné OPC serverem. Karta *Project* PLC projektu umožňuje zobrazit a potažmo změnit port PLC. Výchozí číslo portu je 851 a může zůstat ponecháno. Pokud by však bylo v rámci řešení vytvořeno více PLC projektů, musí mít unikátní číslo portu, aby mohli být adresovány serverem.

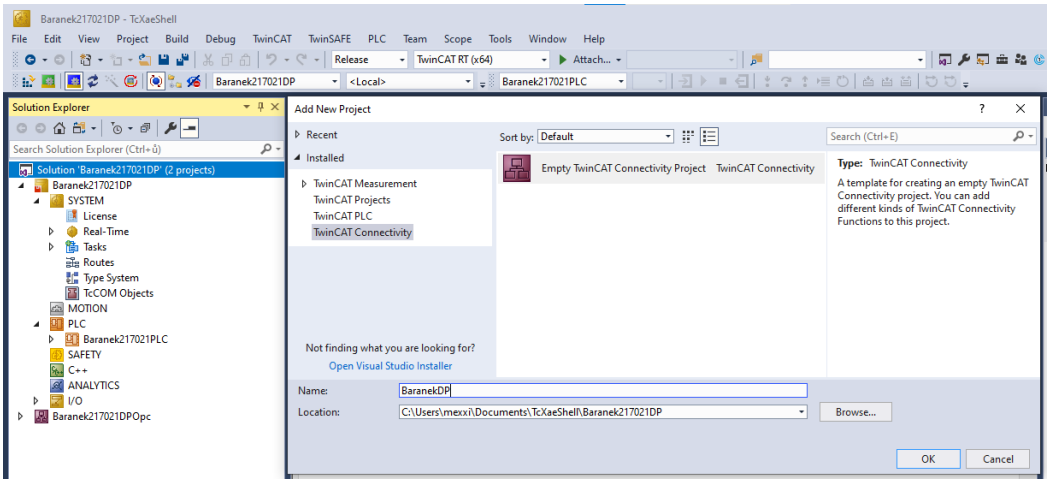


Obr. 38) Nastavení PLC projektu

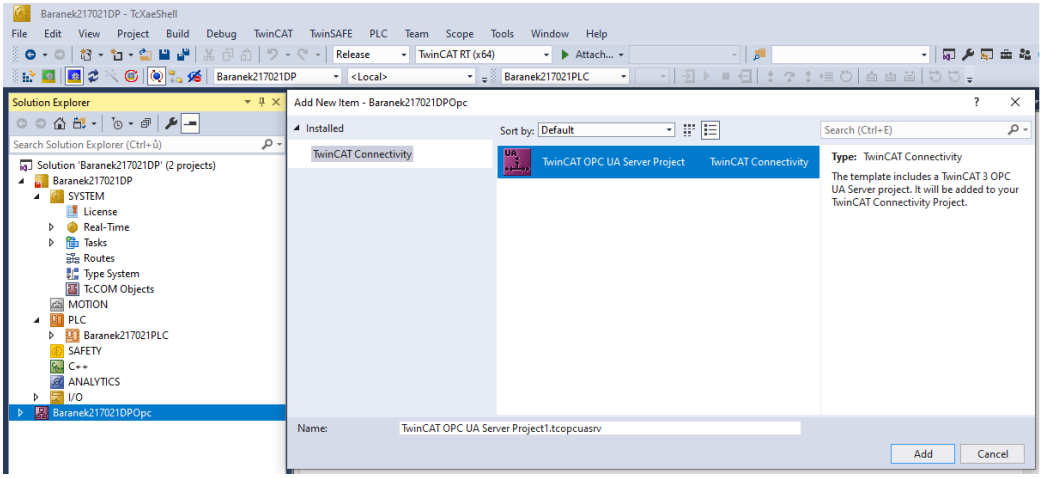
Samotné vytvoření serveru je možné dvěma způsoby – externě, nebo v prostředí TwinCAT. Součástí „TF6100“ balíčku (viz kap. 6.1) je totiž samostatná aplikace pro vytvoření serveru mimo prostředí TwinCAT, která disponuje vícero funkcemi a možnostmi konfigurace serveru. Tato varianta je vhodnější spíše pro rozsáhlejší projekty o více zařízeních. Pro účely této práce je víc než dostačující varianta vytvoření OPC UA serveru pomocí rozšíření přímo v TwinCAT.

Prvním krokem je vytvoření nového projektu typu *TwinCAT Connectivity* v rámci stejné *Solution*, ve které se nachází dříve vytvořený PLC projekt. Ten je vytvořen kliknutím pravým tlačítkem na aktuální *Solution* v *Solution Exploreru* a vybráním možnosti *Add New Project.* V nově otevřeném okně je nutno v levém sloupci vybrat *TwinCAT Connectivity* a vybere prázdný projekt tohoto typu (viz obr. 39). Projekt je vhodné pro pořádek v kořenových souborech uložit do stejného adresáře, ve kterém je uložen PLC projekt.

Takto vytvořenému projektu je nyní možno přidat OPC UA server. Vytvoření serveru je možné kliknutím pravým tlačítkem myši na connectivity projekt v *Solution Exploreru*, z nabídky je vybrána možnost *Add* a následně *New Item*. V následně otevřeném oknu se vybere položka *TwinCAT OPC-UA Server Project* (viz obr. 40).



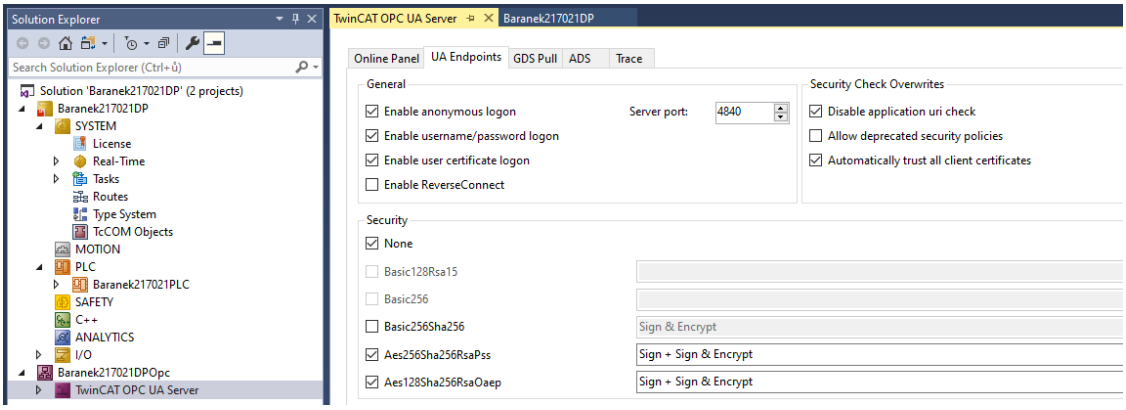
Obr. 39) Vytvoření Connectivity projektu



Obr. 40) Vytvoření server projektu

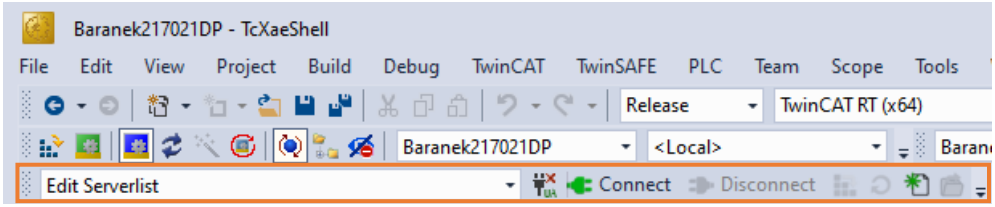
Nově vytvořený OPC UA Server projekt je nyní potřeba nastavit. Prvním krokem je nastavení tzv. serverových endpointů. Endpoint je fyzický adresa dostupná v síti, přes kterou se může client připojit na server. Dvojklikem myši na projekt serveru v *Solution Exploreru* dojde k otevření jeho nastavení. V záložce *UA Endpoints* je potřeba povolit možnost anonymního přihlášení na server *Enable Anonymous logon* a povolit nezabezpečený přístup v oddílu *Security* zaškrtnutím *None* (viz obr. 41). Nachází se zde i serverový port, který je možno změnit, ale v tomto případě byl ponechán na hodnotě *4840*.

Pokud by byl server provozován pro přenos citlivých dat, nebo k řízení reálného pracoviště, kde by potenciální napadení serveru útočníkem mohlo způsobit škody, nezabezpečený přístup by nebylo vhodné povolovat. Process Simulate však dokáže komunikovat prostřednictvím OPC UA protokolu pouze přes nezabezpečený přístup, navíc při řešení této práce nehrozí žádné zneužití přístupu na server a zabezpečení by práci zbytečně komplikovalo.

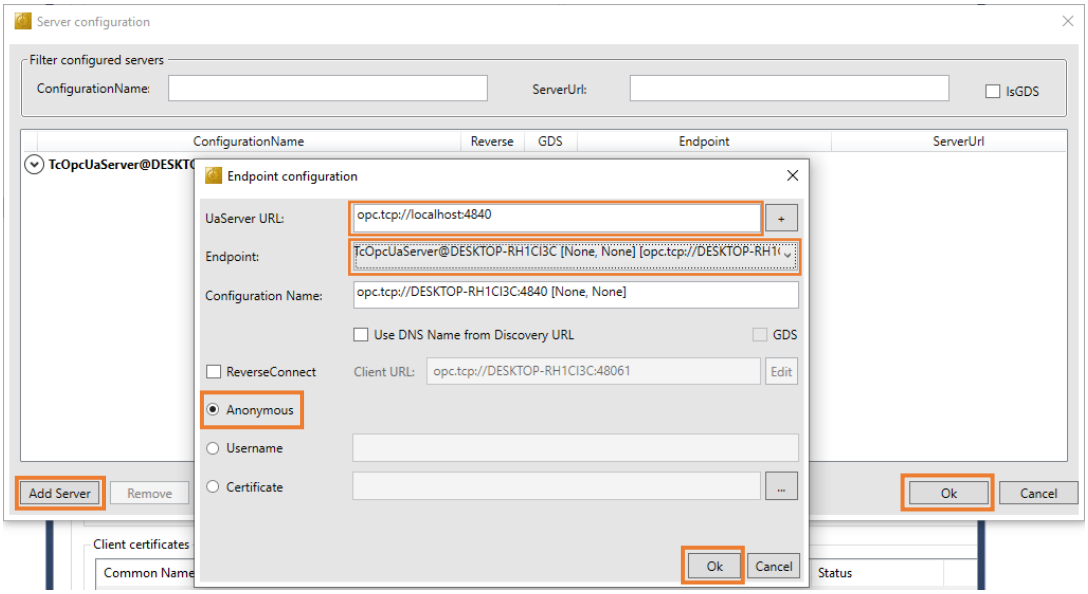


Obr. 41) Nastavení UA Endpoints

Nyní po dokončení těchto nastavení je potřeba aktivovat konfiguraci PLC a zmáčknout *Login*. V hlavní nabídce programu v kartě *View*, následně *Toolbars* a kliknutím na *TwinCAT OPC UA Configurator* je na hlavní lištu přidán panel nástroje pro konfiguraci serveru (viz obr. 42). Kliknutím na *Edit serverlist* dojde k rozbalení seznamu uložených serverů, který bude zatím prázdný. Opětovným kliknutím na *Edit serverlist* dojde k vyvolání okna, kde kliknutím na *Add Server* bude vyvoláno okno pro přidání nového serveru, resp. endpointu na seznam. Pole *UaServer URL* by mělo mít již automaticky vyplněnou URL adresu serveru. Pokud tomu tak není, je nutné zadat adresu *opc.tcp://localhost:4840*. Po vyplnění adresy je potřeba vybrat konkrétní endpoint se zabezpečením *[None, None]*, ujistit se, že je zvoleno *Anonymous* přihlášení a potvrdit konfiguraci. Následně je vybrán nově přidaný endpoint na seznamu a výběr potvrzen. Postup vysvětluje obr. 43.

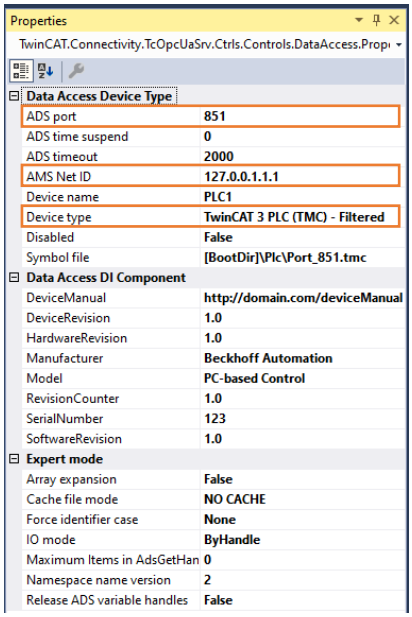


Obr. 42) TwinCAT OPC UA Configurator



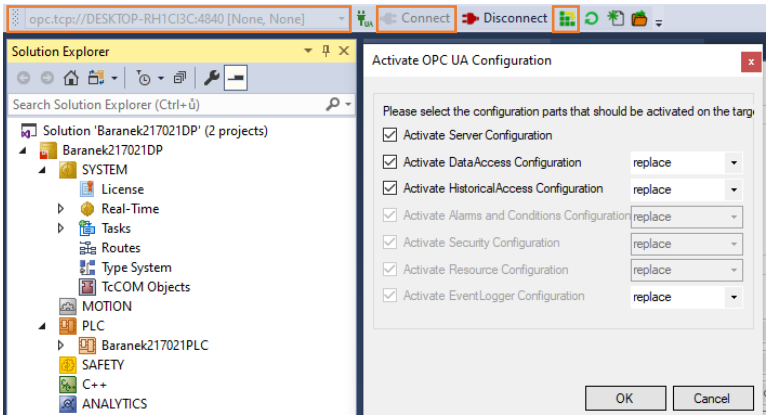
Obr. 43) Přidání endpointu na seznam

Následně je třeba přidat tzv. *device* do projektu OPC UA Serveru. Kliknutím na projekt v *Solution Exploreru* dojde k jeho rozbalení. Pro účely této práce, kdy je záměrem vyměňovat data (proměnné) v reálném čase, bylo vytvořeno *Data Access Device.* To se provede pravým kliknutím na *Data Access* a použitím příkazu *Add Device.* Ve vyvolaném oknu je možno nastavit několik parametrů. Těmi zásadními jsou *Ads Port, AMS Net ID* a *Device Type*. Hodnota *Ads portu* musí odpovídat portu PLC projektu, což je ve výchozím nastavení hodnota *851*. Stejně tak *AMS Net ID* není potřeba měnit. Pokud je *Device Type* nastaven jako *Filtered*, tak na OPC Server budou zapisovány jen ty globální proměnné použité v PLC, které budou mít přiřazen speciální atribut při jejich definici, což je v tomto případě vhodnější varianta. Finální konfigurace je vidět na obr. 44.



Obr. 44) Konfigurace Data Access Device

Posledním krokem tvorby OPC UA server je aktivace jeho konfigurace. Nejprve je potřeba v panelu OPC UA konfigurátoru připojit server. To je možno provést vybráním endpointu a kliknutím na *Connect*. Následně po úspěšném připojení dojde k zpřístupnění dalších funkcí panelu včetně *Activate OPC UA Server Config*. Výběrem toho příkazu je vyvoláno okno s možnostmi aktivace, v němž je vhodné ponechat všechny možnosti zatrhnuté (viz obr. 45) pro přepsání všech starých parametrů i pro budoucí změny konfigurace.



Obr. 45) Aktivace OPC UA konfigurace

6.3 Definice proměnných v TwinCAT 3

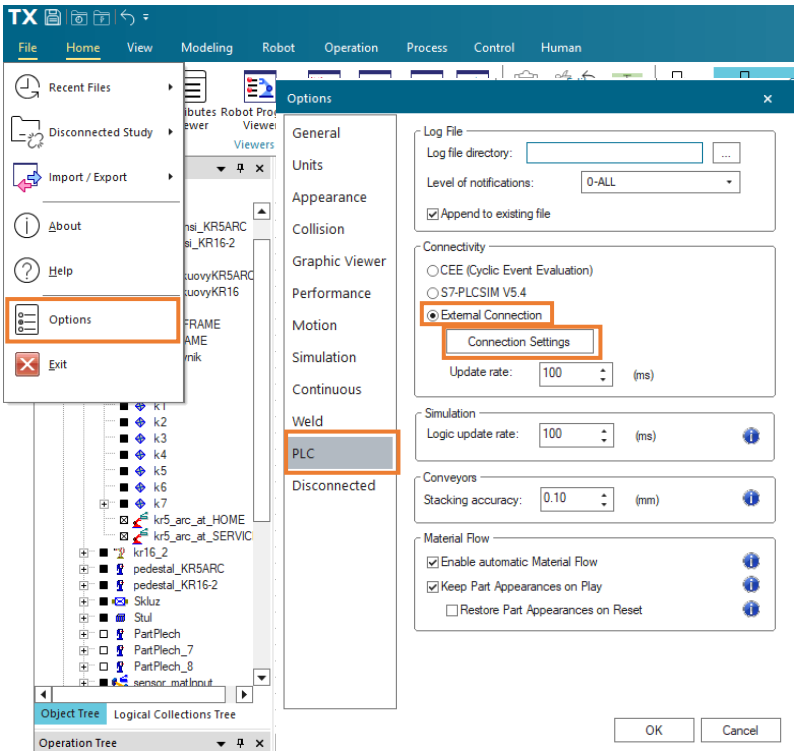
Proměnné, které mají být součástí OPC UA komunikace musí být definované jako globální, tedy v *Global Variable List*. Pro každý robot bylo využito datového typu *Structure* k definici zvlášť vstupních a zvlášť výstupních proměnných robotických signálů. Tyto struktury jsou následně volány v *Global Variable List* pojmenované podle označení robotů. Každá takto definovaná proměnná má díky strukturování před jejím názvem navíc prefix, který specifikuje, ke kterému robotu je vázána a zda-li se jedná o vstupní, nebo výstupní signál robotu. Všechny proměnné, které mají být dostupné na OPC UA serveru musí mít speciální identifikátor *{attribute ´OPC.UA.DA´ := ´1´}*. Výhodou této komunikace navíc je, že proměnným není potřeba přiřazovat adresy, protože jejich propojení je uskutečněno na základě jejich názvů. Definice části proměnných účastnících se komunikace je na obr. 46.

```
5    {attribute 'OPC.UA.DA':='1'}
6    b_programEnded : BOOL;
7    {attribute 'OPC.UA.DA':='1'}
8    byte_mirrorProgramNumber : BYTE;
9    {attribute 'OPC.UA.DA':='1'}
10   b_errorProgramNumber : BOOL;
11   {attribute 'OPC.UA.DA':='1'}
12   b_robotReady : BOOL;
13   {attribute 'OPC.UA.DA':='1'}
14   b_at_HOME : BOOL;
15   {attribute 'OPC.UA.DA':='1'}
16   b_at_SERVICE : BOOL;
17   {attribute 'OPC.UA.DA':='1'}
18   b_proAct : BOOL; //CYCLE ON
19   {attribute 'OPC.UA.DA':='1'}
20   b_periRdy : BOOL; //TRUE = drives are switched on
21   {attribute 'OPC.UA.DA':='1'}
22   b_robStopped : BOOL;    //TRUE if robot is standstill
23   {attribute 'OPC.UA.DA':='1'}
24   b_proMove: BOOL;    //TRUE if robot is axis are moving
```

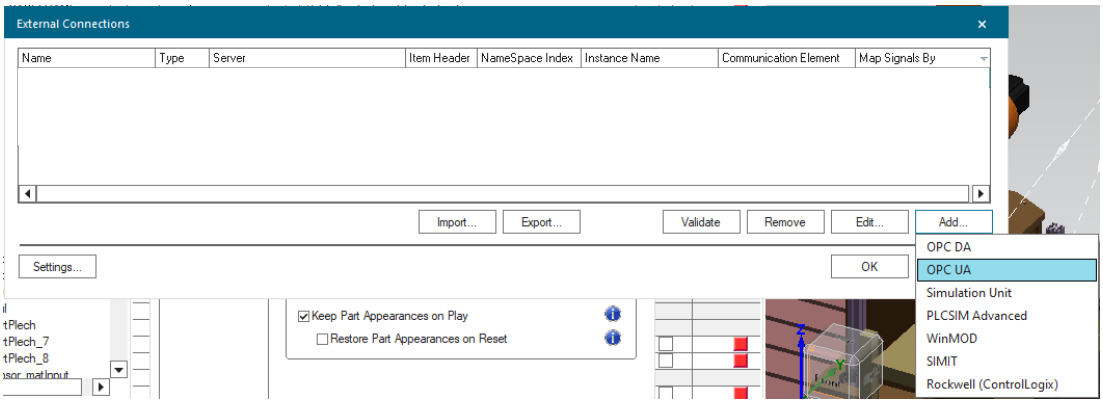
Obr. 46) Definice proměnných pro OPC UA

6.4 Propojení PLC a Process Simulate

Pro připojení Process Simulate k externímu řízení simulace je potřeba nejprve otevřít nastavení programu, které se nachází na hlavním panelu v kartě *File*. V jejím rozbalovacím menu se vybere *Options*. Otevřené okno nastavení má po levé straně v nabídce možnost *PLC*. V záložce *PLC* je potřeba v oddílů *Connectivity* přepnout řízení simulace z *CEE* (cyklická událostně řízená) na *External Connection* (viz obr. 47). Následně po kliknutí na tlačítko *Connection Settings* se zobrazí okno *External Connections* pro samotnou konfiguraci externích PLC připojení. V okně *External Connections* je pomocí tlačítka *Add…* možno přidat nové připojení z nabídky, ve které je zvoleno *OPC UA* (viz obr. 48).

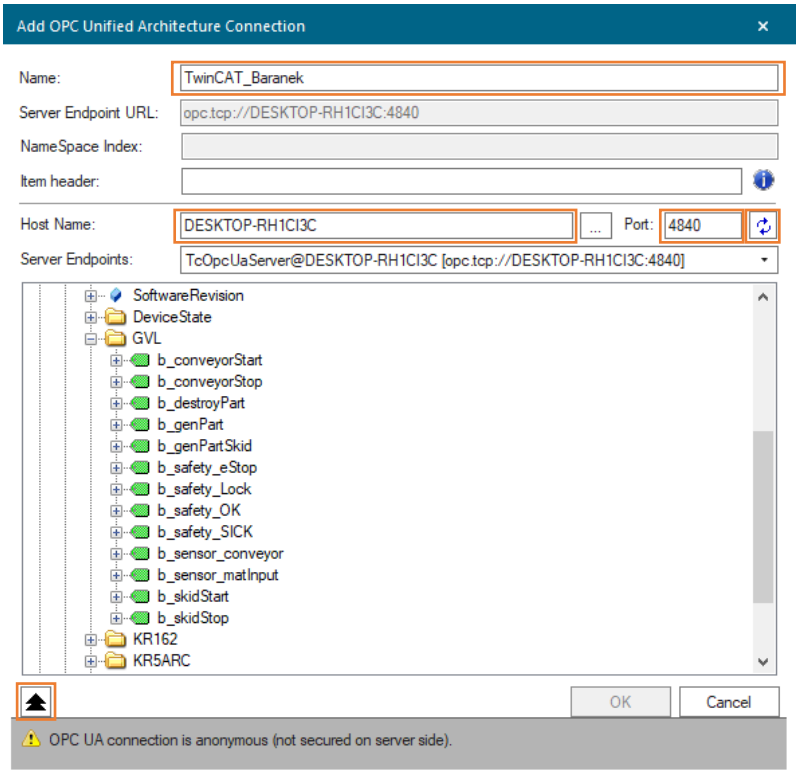


Obr. 47) Nastavení externího připojení



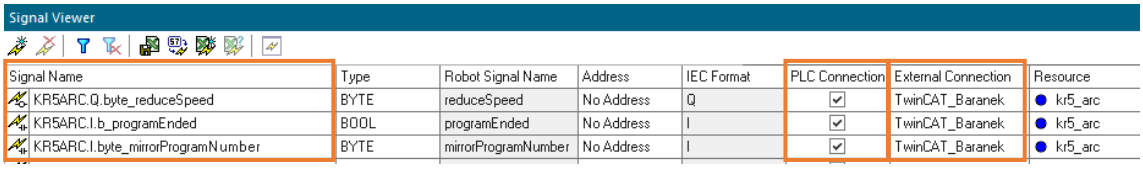
Obr. 48) Přidání nového připojení

Nakonfigurování nového připojení v okně *Add OPC Unified Architecture Connection* sestává nejprve z rozšíření okna pomocí černých šipek v levém dolním rohu, následně je požadováno vyplnění jména tohoto připojení (může být libovolné), vyplnění adresy OPC UA serveru a případně jeho portu, pokud je odlišný od výchozího. Kliknutím na tlačítko se symbolem modrých šipek dojde k připojení k serveru. Program automaticky vyplní ostatní parametry okna a v jeho spodní části dojde k zobrazení obsahu serveru. Postup je vidět na obr. 49.



Obr. 49) Konfigurace připojení

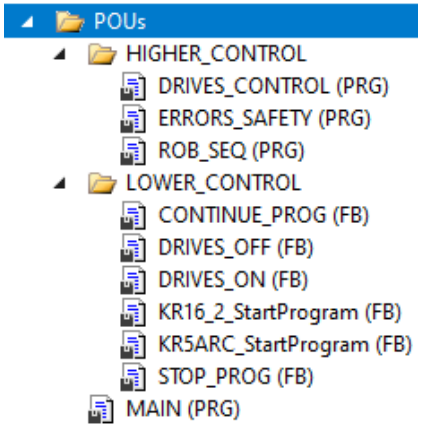
Proměnné z PLC jsou v Process Simulate reprezentovány signály vytvořenými v kap. 5.4. Aby došlo k správnému napárování signálů k proměnným na OPC UA serveru, je potřeba nejprve provést jejich konfiguraci v okně *Signal Viewer*. Pokud je toto okno skryto, je možné jej opět zobrazit stejnojmenným tlačítkem v nabídce *Viewers* pod kartou *View* nebo kartou *Home*. Prvním předpokladem pro správnou komunikaci je shodný název signálu *Signal Name*, dále je u každého signálu nutno zatrhnout kolonku *PLC Connection* a ve sloupci *External Connection* vybrat z nabídky název externího připojení, ke kterému má signál patřit (viz obr. 50). Pokud by byl byť jen jeden ze všech signálů v Signal Vieweru nakonfigurován špatně, nebude fungovat přenos ani jednoho signálu. Výpis všech signálů simulace je v příloze 1.



Obr. 50) Konfigurace signálů pro komunikaci

7 PLC ŘÍZENÍ

Řídící programy pro PLC byly v TwinCAT rozděleny do dvou pomyslných skupin nazvaných „LOWER CONTROL“ (pro nižší úroveň řízení) a „HIGHER CONTROL“ (pro vyšší úroveň řízení). Takovéto rozvržení má mimo jiné výhodu ve větší přehlednosti a čitelnosti programu. Struktura rozvržení je vidět na obr. 51. Veškeré programování bylo provedeno v jazyku Structured Text (ST).



Obr. 51) Struktura řídících programů

Skupina nižší úrovně sestává ze šesti funkčních bloků, přičemž každý z nich je určen pro vykonání konkrétního úkonu spojeného s roboty, jako je např. spuštění robotického programu. Vykonání „akční“ části programu obsaženého ve funkčním bloku proběhne pouze v moment, kdy je externě zavolán některým z programů vyšší úrovně řízení nebo tlačítkem z HMI. Detailnějšímu vysvětlení funkčních bloků se věnuje kapitola 7.1. Skupina vyššího řízení je tvořena třemi programy:

- „DRIVES_CONTROL“,
- „ERRORS_SAFETY“,
- „ROB_SEQ“.

První jmenovaný obsahuje stavový automat, který řeší sekvenci zapínání a vypínání robotických pohonů obou robotů. Ovládán je pomocí HMI a odkazuje se na funkční bloky „DRIVES_ON“ a „DRIVES_OFF“. Druhý program obsahuje podmínkovou logickou strukturu řešící bezpečnost pracoviště, „acknowledgment“ a resetování potenciálních errorů. Poslední a zároveň nejobsáhlejší z programů obsahuje především stavový automat řídící celou sekvenci a logiku informující HMI o jejím stavu. Odkazuje se na funkční bloky „CONTINUE_PROG“, „STOP PROGRAM“, „KR16_2_StartProgram“ a „KR5ARC_StartProgram“. Celou logiku programu „ROB_SEQ“ znázorňuje vývojový diagram v příloze 2.

7.1 Popis funkčních bloků

7.1.1 Funkční blok zapnutí pohonů

Funkční blok zapnutí pohonů „DRIVES_ON“ je volán z nadřazeného programu „DRIVES_CONTROL“. Funkčnímu bloku je definována jedna vstupní proměnná „b_execute“ a dvě výstupní proměnné „b_done“ a „b_busy“. Proměnná „b_done“ indikuje, zda-li byla „akční“ část funkčního bloku úspěšně provedena a proměnná „b_busy“ značí, že funkční blok je „zaneprázdněn“. Program se skládá ze čtyř podmínek (viz obr. 52):

- První podmínka je vykonána, pokud je zaznamenána náběžná hrana na vstupní proměnné „b_execute“.
- Druhá podmínka je vykonána, pokud PLC dostane zpětnou vazbu od robotického controlleru, že pohony obou robotů byly zapnuty.
- Třetí podmínka je vykonána, pokud je zaznamenána náběžná hrana na „time-out“ časovači.
- Čtvrtá podmínka je vykonána, pokud je v HMI zmáčknuto tlačítko „RESET“.

DRIVES_ON

```
VAR_INPUT
    b_execute : BOOL;
END_VAR
VAR_OUTPUT
    b_done : BOOL;
    b_busy : BOOL;
END_VAR
VAR
    re_execute : R_TRIG;
    b_hold_tout : BOOL;
    ton_tout : TON;
    re_ton_tout : R_TRIG;
    time_tout : TIME := T#5000ms;
END_VAR
```

100

```
re_execute(CLK:= b_execute, Q=> );
ton_tout(IN:= b_hold_tout, PT:= time_tout);
re_ton_tout(CLK:= ton_tout.Q);

IF re_execute.Q THEN
    b_hold_tout := TRUE; //PROMĚNNÁ PRO ZAPNUTÍ TIME-OUT ČASOVAČE
    b_done := FALSE;
    b_busy := TRUE;
    KR162.Q.b_drivesOn := TRUE; //SIGNÁL PRO ZAPNUTÍ POHONŮ KUKA KR16-2
    KR5ARC.Q.b_drivesOn := TRUE; //SIGNÁL PRO ZAPNUTÍ POHONŮ KUKA KR 5 arc
END_IF

IF KR5ARC.I.b_periRdy AND KR162.I.b_periRdy THEN
    b_hold_tout := FALSE; //VYRESETOVÁNÍ ČASOVAČE
    KR162.Q.b_drivesOn := FALSE; //SIGNÁL PRO ZAPNUTÍ POHONŮ KUKA KR16-2
    KR5ARC.Q.b_drivesOn := FALSE; //SIGNÁL PRO ZAPNUTÍ POHONŮ KUKA KR 5 arc
    b_busy := FALSE;
    b_done := TRUE;
    GVL.led_drivesOn := TRUE; //ZAPNUTÍ HMI SIGNALIZACE ZAPNUTÝCH POHONŮ
    GVL.led_drivesOff := FALSE; //VYPNUTÍ HMI SIGNALIZZACE VYPNUTÝCH POHONŮ
END_IF

IF re_ton_tout.Q THEN
    b_hold_tout := FALSE; //VYRESETOVÁNÍ ČASOVAČE
    KR162.Q.b_drivesOn := FALSE; //SIGNÁL PRO ZAPNUTÍ POHONŮ KUKA KR16-2
    KR5ARC.Q.b_drivesOn := FALSE; //SIGNÁL PRO ZAPNUTÍ POHONŮ KUKA KR 5 arc
    b_busy := FALSE;
    GVL.b_error := TRUE; //PROMĚNNÁ INDIKUJÍCÍ CHYBU
    GVL.s_errorMessage := 'DRIVES_DIDNT_SWITCH_ON'; //VYPSÁNÍ KONKRÉTNÍ CHYBY
END_IF

IF GVL.b_reset THEN
    b_hold_tout := FALSE;   //VYRESETOVÁNÍ ČASOVAČE
    KR162.Q.b_drivesOn := FALSE;    //SIGNÁL PRO ZAPNUTÍ POHONŮ KUKA KR16-2
    KR162.Q.b_drivesOn := FALSE;    //SIGNÁL PRO ZAPNUTÍ POHONŮ KUKA KR 5 arc
    b_busy := FALSE;
    b_done := FALSE;
END_IF
```

Obr. 52) Funkční blok „DRIVES_ON“

7.1.2 Funkční blok vypnutí pohonů

Funkční blok zapnutí pohonů „DRIVES_OFF“ je volán z nadřazeného programu „DRIVES_CONTROL“. Funkčnímu bloku je definována jedna vstupní proměnná „b_execute“ a dvě výstupní proměnné „b_done“ a „b_busy“. Proměnná „b_done“ indikuje, zda-li byla „akční“ část funkčního bloku úspěšně provedena a proměnná „b_busy“ značí, že funkční blok je „zaneprázdněn“. Program se skládá ze čtyř podmínek (viz obr. 53):

- První podmínka je vykonána, pokud je zaznamenána náběžná hrana na vstupní proměnné „b_execute“.
- Druhá podmínka je vykonána, pokud PLC dostane zpětnou vazbu od robotického controlleru, že pohony obou robotů byly vypnuty.
- Třetí podmínka je vykonána, pokud je zaznamenána náběžná hrana na „time-out“ časovači.
- Čtvrtá podmínka je vykonána, pokud je v HMI zmáčknuto tlačítko „RESET“.

DRIVES_OFF

```
FUNCTION_BLOCK DRIVES_OFF
VAR_INPUT
    b_execute : BOOL;
END_VAR
VAR_OUTPUT
    b_busy : BOOL;
    b_done : BOOL;
END_VAR
VAR
    re_execute : R_TRIG;
    b_hold_tout : BOOL;
    ton_tout : TON;
    re_ton_tout : R_TRIG;
    time_tout : TIME := T#5000ms;
END_VAR
```

```
re_execute(CLK:= b_execute);
ton_tout(IN:= b_hold_tout, PT:= time_tout);
re_ton_tout(CLK:= ton_tout.Q);

IF re_execute.Q THEN
    b_hold_tout := TRUE; //PROMĚNNÁ PRO ZAPNUTÍ TIME-OUT ČASOVAČE
    b_done := FALSE;
    b_busy := TRUE;
    KR162.Q.b_drivesOff := TRUE; //SIGNÁL PRO VYPNUTÍ POHONŮ KUKA KR16-2
    KR5ARC.Q.b_drivesOff := TRUE; //SIGNÁL PRO VYPNUTÍ POHONŮ KUKA KR 5 arc
END_IF

IF NOT KR5ARC.I.b_periRdy AND NOT KR162.I.b_periRdy THEN
    b_hold_tout := FALSE; //VYRESETOVÁNÍ ČASOVAČE
    KR162.Q.b_drivesOff := FALSE; //SIGNÁL PRO VYPNUTÍ POHONŮ KUKA KR16-2
    KR5ARC.Q.b_drivesOff := FALSE; //SIGNÁL PRO VYPNUTÍ POHONŮ KUKA KR 5 arc
    b_busy := FALSE;
    b_done := TRUE;
    GVL.led_drivesOff := TRUE; //ZAPNUTÍ HMI SIGNALIZACE VYPNUTÝCH POHONŮ
    GVL.led_drivesOn := FALSE; //VYPNUTÍ HMI SIGNALIZACE ZAPNUTÝCH POHONŮ
END_IF

IF re_ton_tout.Q THEN
    b_hold_tout := FALSE; //VYRESETOVÁNÍ ČASOVAČE
    KR162.Q.b_drivesOn := FALSE; //SIGNÁL PRO VYPNUTÍ POHONŮ KUKA KR16-2
    KR5ARC.Q.b_drivesOn  := FALSE; //SIGNÁL PRO VYPNUTÍ POHONŮ KUKA KR 5 arc
    b_busy := FALSE;
    GVL.b_error := TRUE; //PROMĚNNÁ INDIKUJÍCÍ CHYU
    GVL.s_errorMessage := 'DRIVES_DIDNT_SWITCH_OFF'; //VYPSÁNÍ KONKRÉTNÍ CHYBY V HMI
END_IF

IF GVL.b_reset THEN
    b_hold_tout := FALSE; //VYRESETOVÁNÍ ČASOVAČE
    KR162.Q.b_drivesOff := FALSE; //SIGNÁL PRO VYPNUTÍ POHONŮ KUKA KR16-2
    KR5ARC.Q.b_drivesOff  := FALSE; //VYPNUTÍ HMI SIGNALIZACE ZAPNUTÝCH POHONŮ
    b_busy := FALSE;
    b_done := FALSE;
END_IF
```

Obr. 53) Funkční blok „DRIVES_OFF“

7.1.3 Funkční blok zahájení programu

Funkční blok pro zahájení vykonávání robotického programu byl vytvořen zvlášť pro každý robot. Strukturou programu se nijak neliší, rozdílné jsou jenom proměnné robotických signálů, které ovládají. Popsán bude proto pouze jeden z nich – „KR16_2_StartProgram“. Funkčnímu bloku je definována jedna vstupní proměnná „b_execute“ a dvě výstupní proměnné „b_done“ a „b_busy“. Proměnná „b_done“ indikuje, zda-li byla „akční“ část funkčního bloku úspěšně provedena a proměnná „b_busy“ značí, že funkční blok je „zaneprázdněn“. Program se skládá ze čtyř podmínek (viz obr. 54):

- První podmínka je vykonána, pokud je zaznamenána náběžná hrana na vstupní proměnné „b_execute“ a pokud není druhý robot v pohybu ani nevykonává robotický program, aby nedošlo ke kolizi, pokud by se nacházel v kolizní zóně (u skluzu).
- Druhá podmínka je vykonána, pokud se robot začne hýbat a vykonávat program.
- Třetí podmínka je vykonána, pokud je zaznamenána náběžná hrana na „time-out“ časovači.
- Čtvrtá podmínka je vykonána, pokud je v HMI zmáčknuto tlačítko „RESET“.

KR16_2_StartProgram

```
VAR_INPUT
    b_execute : BOOL;
END_VAR
VAR_OUTPUT
    b_busy : BOOL;
    b_done : BOOL;
END_VAR
VAR
    re_execute : R_TRIG;
    b_hold_tout : BOOL;
    ton_tout : TON;
    re_ton_tout : R_TRIG;
    time_tout : TIME := T#2000ms;
END_VAR
```

```
re_execute(CLK:= b_execute);

ton_tout(IN:= b_hold_tout, PT:= time_tout);

re_ton_tout(CLK:= ton_tout.Q);

IF re_execute.Q AND NOT KR5ARC.I.b_proMove AND NOT KR5ARC.I.b_proAct THEN
    b_hold_tout := TRUE; //PROMĚNNÁ PRO ZAPNUTÍ TIME-OUT ČASOVAČE
    b_done := FALSE;
    b_busy := TRUE;
    KR162.Q.b_startProgram := TRUE; //SIGNAL PRO SPUŠTĚNÍ ROBOTICKÉHO PROGRAMU
END_IF

IF KR162.I.b_proAct AND KR162.I.b_proMove THEN
    b_hold_tout := FALSE;
    KR162.Q.b_startProgram := FALSE; //SIGNAL PRO SPUŠTĚNÍ ROBOTICKÉHO PROGRAMU
    b_busy := FALSE;
    b_done := TRUE;
END_IF

IF re_ton_tout.Q THEN
    b_hold_tout := FALSE; //VYRESETOVÁNÍ ČASOVAČE
    KR162.Q.b_startProgram := FALSE; //SIGNAL PRO SPUŠTĚNÍ ROBOTICKÉHO PROGRAMU
    b_busy := FALSE;
    GVL.b_error := TRUE; //PROMĚNNÁ INDIKUJÍCÍ CHYBU
    GVL.s_errorMessage := 'PROG_NOT_STARTED'; //VYPSÁNÍ KONKRÉTNÍ CHYBY
END_IF

IF GVL.b_reset THEN
    b_hold_tout := FALSE; //VYRESETOVÁNÍ ČASOVAČE
    KR162.Q.b_startProgram := FALSE; //SIGNAL PRO SPUŠTĚNÍ ROBOTICKÉHO PROGRAMU
    b_busy := FALSE;
    b_done := FALSE;
END_IF
```

Obr. 54) Funkční blok „KR16_2_StartProgram“

7.1.4 Funkční blok pozastavení programu

Funkční blok pro pozastavení vykonávání robotického programu „STOP_PROG“ je volán z nadřazeného programu „ROB_SEQ“, nebo exekuován tlačítkem z HMI. Funkčnímu bloku jsou definovány stejné vstupní a výstupní proměnné, jako předchozím blokům. Program se skládá z pěti podmínek (viz obr. 55):

- První podmínka je vykonána, pokud je zaznamenána náběžná hrana na vstupní proměnné „b_execute“, nebo zmáčknuto tlačítko v HMI a pokud robot KR 5 arc není zastaven.
- Druhá podmínka je vykonána, pokud je zaznamenána náběžná hrana na vstupní proměnné „b_execute“, nebo zmáčknuto tlačítko v HMI a pokud robot KR 16-2 není zastaven.
- Třetí podmínka je vykonána, pokud je zpětná vazba, že jsou oba roboty zastaveny.
- Čtvrtá podmínka je vykonána, pokud je zaznamenána náběžná hrana na „time-out“ časovači.
- Pátá podmínka je vykonána, pokud je v HMI zmáčknuto tlačítko „RESET“.

```
    b_hold_tout : BOOL;
    ton_tout : TON;
    re_ton_tout : R_TRIG;
    time_tout : TIME := T#2000ms;
END_VAR
```

```
re_btn_execute(CLK:= GVL.btn_stopProg);
re_execute(CLK:= b_execute);
ton_tout(IN:= b_hold_tout, PT:= time_tout);
re_ton_tout(CLK:= ton_tout.Q);
GVL.b_robotsStopped := NOT KR162.I.b_proAct AND NOT KR5ARC.I.b_proAct       //OBA ROBOTY ZASTAVENY = TRUE
                       AND KR162.I.b_robStopped AND KR5ARC.I.b_robStopped;

IF re_execute.Q OR re_btn_execute.Q AND NOT KR5ARC.I.b_robStopped THEN
    b_hold_tout := TRUE; //PROMĚNNÁ PRO ZAPNUTÍ TIME-OUT ČASOVAČE
    b_done := FALSE;
    b_busy := TRUE;
    KR5ARC.Q.b_moveEnable := FALSE; //SIGNAL PRO ZABLOKOVÁNÍ POHYBU KR 5 arc
    KR5ARC.Q.b_programPause := TRUE; //JEN PRO PS SIMULACI. SIGNAL POZASTAVENI PROGRAMU KR 5 arc
    GVL.led_seqStopped := TRUE; //ZAPNUTI HMI SIGNALIZACE POZASTAVENE SEKVENCE
END_IF

IF re_execute.Q OR re_btn_execute.Q AND NOT KR162.I.b_robStopped THEN
    b_hold_tout := TRUE;    //PROMĚNNÁ PRO ZAPNUTÍ TIME-OUT ČASOVAČE
    b_done := FALSE;
    b_busy := TRUE;
    KR162.Q.b_moveEnable := FALSE;  //SIGNAL PRO ZABLOKOVÁNÍ POHYBU KR 16-2
    KR162.Q.b_programPause := TRUE; //JEN PRO PS SIMULACI. SIGNAL POZASTAVENI PROGRAMU KR16-2
    GVL.led_seqStopped := TRUE; //ZAPNUTI HMI SIGNALIZACE POZASTAVENE SEKVENCE
END_IF

IF GVL.b_robotsStopped THEN
    b_execute := FALSE;
    b_hold_tout := FALSE;   //VYRESETOVÁNÍ ČASOVAČE
    b_busy := FALSE;
    b_done := TRUE;
END_IF


IF re_ton_tout.Q THEN
    b_execute := FALSE;
    b_hold_tout := FALSE;   //VYRESETOVÁNÍ ČASOVAČE
    b_busy := FALSE;
    GVL.b_error := TRUE;    //PROMĚNNÁ INDIKUJÍCÍ CHYBU
    GVL.s_errorMessage := 'ROBOTS NOT STOPPED'; //VYPSÁNÍ KONKRÉTNÍ CHYBY V HMI
END_IF

IF GVL.b_reset THEN
    b_execute := FALSE;
    b_hold_tout := FALSE;   //VYRESETOVÁNÍ ČASOVAČE
    b_busy := FALSE;
    b_done := FALSE;
    KR162.Q.b_programPause := FALSE; //JEN PRO PS SIMULACI. SIGNAL POZASTAVENI PROGRAMU KR16-2
    KR5ARC.Q.b_programPause := FALSE; //JEN PRO PS SIMULACI. SIGNAL POZASTAVENI PROGRAMU KR 5 arc
    GVL.led_seqStopped := FALSE;    //VYPNUTI HMI SIGNALIZACE POZASTAVENE SEKVENCE
END_IF
```

Obr. 55) Funkční blok „STOP_PROG“

7.1.5 Funkční blok pokračování programu

Funkční blok pro pokračování vykonávání robotického programu „CONTINUE_PROG“ je volán z nadřazeného programu „ROB_SEQ“, nebo exekuován tlačítkem z HMI. Funkčnímu bloku jsou definovány stejné vstupní a výstupní proměnné, jako předchozím blokům. Program se skládá z pěti podmínek (viz obr. 56):

- První podmínka je vykonána, pokud je zmáčknuto tlačítko v HMI a pokud je signál pro pozastavení KR 5 arc „TRUE“.
- První podmínka je vykonána, pokud je zmáčknuto tlačítko v HMI a pokud je signál pro pozastavení KR 16-2 „TRUE“.
- Třetí podmínka je vykonána, pokud je zpětná vazba, že ani jeden z robotů není zastaven.
- Čtvrtá podmínka je vykonána, pokud je zaznamenána náběžná hrana na „time-out“ časovači.
- Pátá podmínka je vykonána, pokud je v HMI zmáčknuto tlačítko „RESET“.

CONTINUE_PROG

```
FUNCTION_BLOCK CONTINUE_PROG
VAR_INPUT
    b_execute : BOOL;
END_VAR
VAR_OUTPUT
    b_busy : BOOL;
    b_done : BOOL;
END_VAR
VAR
    re_execute : R_TRIG;
    b_hold_tout : BOOL;
    ton_tout : TON;
    re_ton_tout : R_TRIG;
    time_tout : TIME := T#2000ms;
END_VAR
```

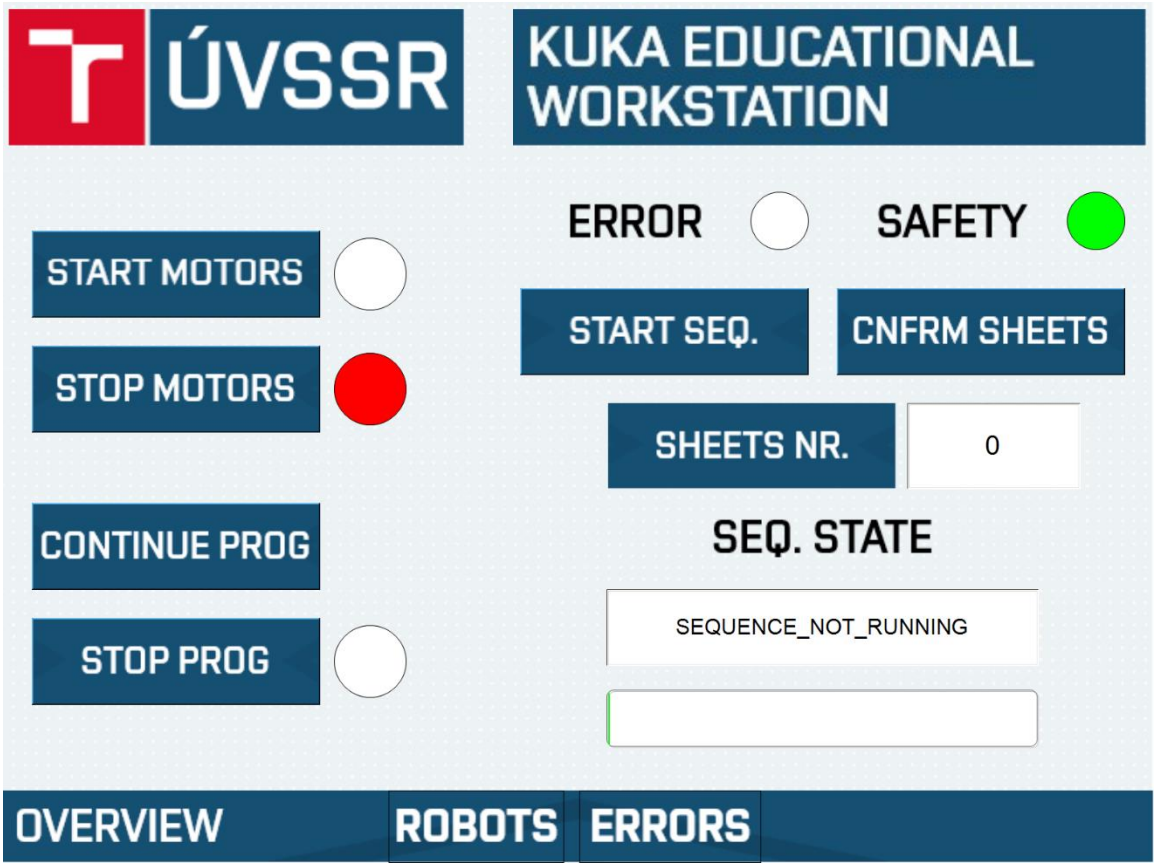
```
re_execute(CLK:= GVL.btn_continueProg);
ton_tout(IN:= b_hold_tout, PT:= time_tout);
re_ton_tout(CLK:= ton_tout.Q);

IF re_execute.Q AND KR5ARC.Q.b_programPause THEN
    b_hold_tout := TRUE; //PROMĚNNÁ PRO ZAPNUTÍ TIME-OUT ČASOVAČE
    b_done := FALSE;
    b_busy := TRUE;
    KR5ARC.Q.b_moveEnable := TRUE; //SIGNÁL PRO ODBLOKOVÁNÍ POHYBU KR 5 arc
    KR5ARC.Q.b_programPause := FALSE; //JEN PRO PS SIMULACI. SIGNAL ODPAUZOVÁNÍ PROGRAMU KR 5 arc
END_IF

IF re_execute.Q AND KR162.Q.b_programPause THEN
    b_hold_tout := TRUE; //PROMĚNNÁ PRO ZAPNUTÍ TIME-OUT ČASOVAČE
    b_done := FALSE;
    b_busy := TRUE;
    KR162.Q.b_moveEnable := TRUE; //SIGNÁL PRO ODBKLOKOVÁNÍ POHYBU KR 16-2
    KR162.Q.b_programPause := FALSE; //JEN PRO PS SIMULACI. SIGNAL ODPAUZOVÁNÍ PROGRAMU KR 16-2
END_IF

IF NOT KR5ARC.I.b_robStopped OR NOT KR162.I.b_robStopped THEN
    b_hold_tout := FALSE;   //VYRESETOVÁNÍ ČASOVAČE
    b_busy := FALSE;
    b_done := TRUE;
    GVL.led_seqStopped := FALSE; //VYPNUTI HMI SIGNALIZACE POZASTAVENI SEKVENCE
END_IF

IF re_ton_tout.Q THEN
    b_hold_tout := FALSE;   //VYRESETOVÁNÍ ČASOVAČE
    b_busy := FALSE;
    GVL.b_error := TRUE;    //PROMĚNNÁ INDIKUJÍCÍ CHYBU
    GVL.s_errorMessage := 'PROGRAM NOT CONTINUED'; //VYPSÁNÍ KONKRÉTNÍ CHYBY V HMI
END_IF

IF GVL.b_reset THEN
    b_hold_tout := FALSE;   //VYRESETOVÁNÍ ČASOVAČE
    b_busy := FALSE;
    b_done := FALSE;
    KR162.Q.b_programPause := FALSE; //JEN PRO PS SIMULACI. SIGNAL ODPAUZOVÁNÍ PROGRAMU KR 16-2
    KR5ARC.Q.b_programPause := FALSE; //JEN PRO PS SIMULACI. SIGNAL ODPAUZOVÁNÍ PROGRAMU KR 5 arc
END_IF
```

Obr. 56) Funkční blok „CONTINUE_PROG“

7.2 Human Machine Interface

Vizualizace HMI je byla tvořena cíleně pro řídící panel Beckhoff CP6901-0001-0000, kterým je vybaveno reálné pracoviště. Velikost tohoto displeje je 12“ a rozlišení 800 x 600 px, což bylo vzato v potaz při návrhu. HMI je tvořeno třemi vizualizačními obrazovkami nazvanými „OVERWIEV“, „ERRORS“ a „ROBOTS STATUS“.

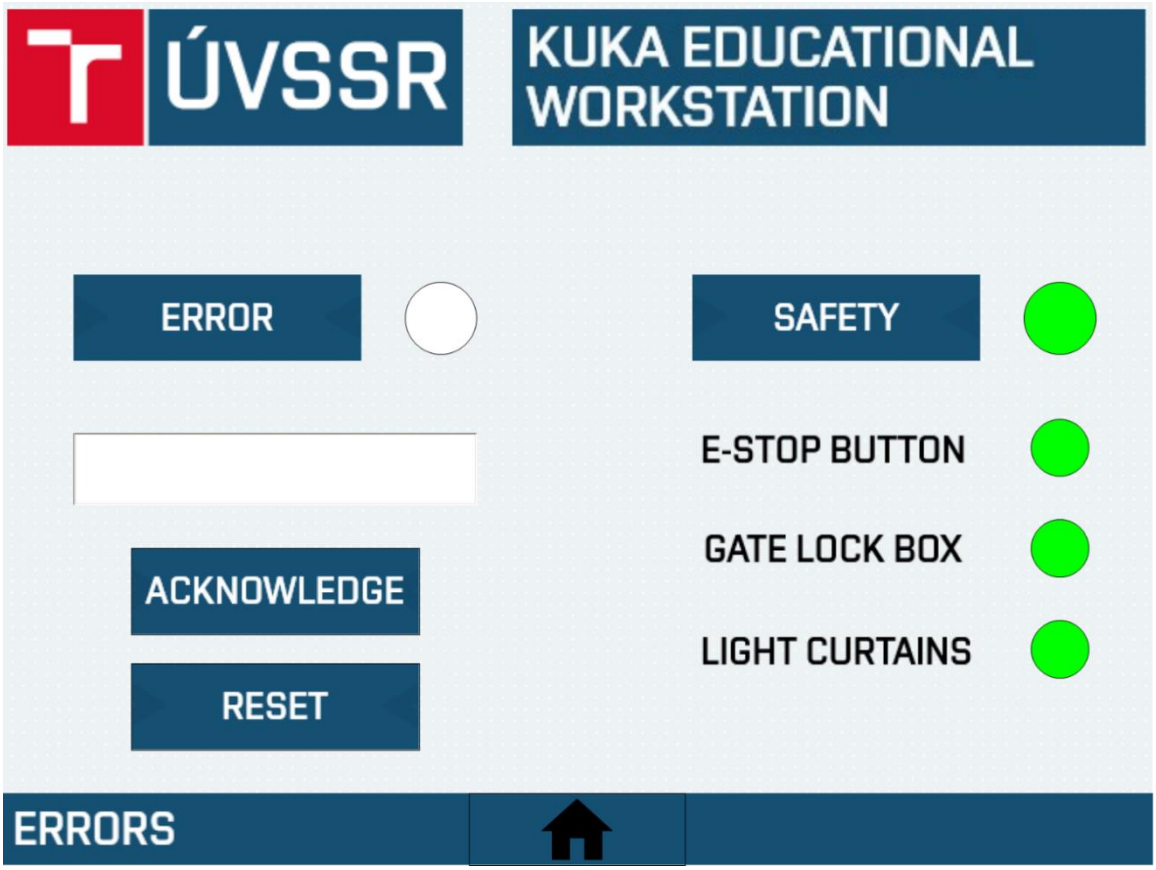
Hlavní a zároveň výchozí obrazovka „OVERVIEW“ (viz obr. 57), která je zobrazena ihned po spuštění, obsahuje všechny prvky potřebné k ovládání vytvořené operace pro pracoviště a základní signalizace stavu. Tlačítka pro ovládání robotů zahrnují zapnutí pohonů, vypnutí pohonů, přerušení vykonávání robotického programu a pokračování jeho vykonávání po přerušení. Pravá strana obrazovky obsahuje tlačítko pro spuštění sekvence, pole pro vyplnění počtu zamýšlených plechů k manipulaci, resp. kolikrát se bude sekvence opakovat a tlačítko potvrzení tohoto počtu. Pod nápisem „SEQ. STATE“ je textové pole informující o aktuálním vykonávaném kroku sekvence a animovaný „progress bar“ symbolizující procentuální pokrok sekvence. Na obrazovce je také signalizace chyby a signalizace stavu safety prvků pracoviště. Přepnutí na další dostupné obrazovky je možné tlačítky na spodním panelu.



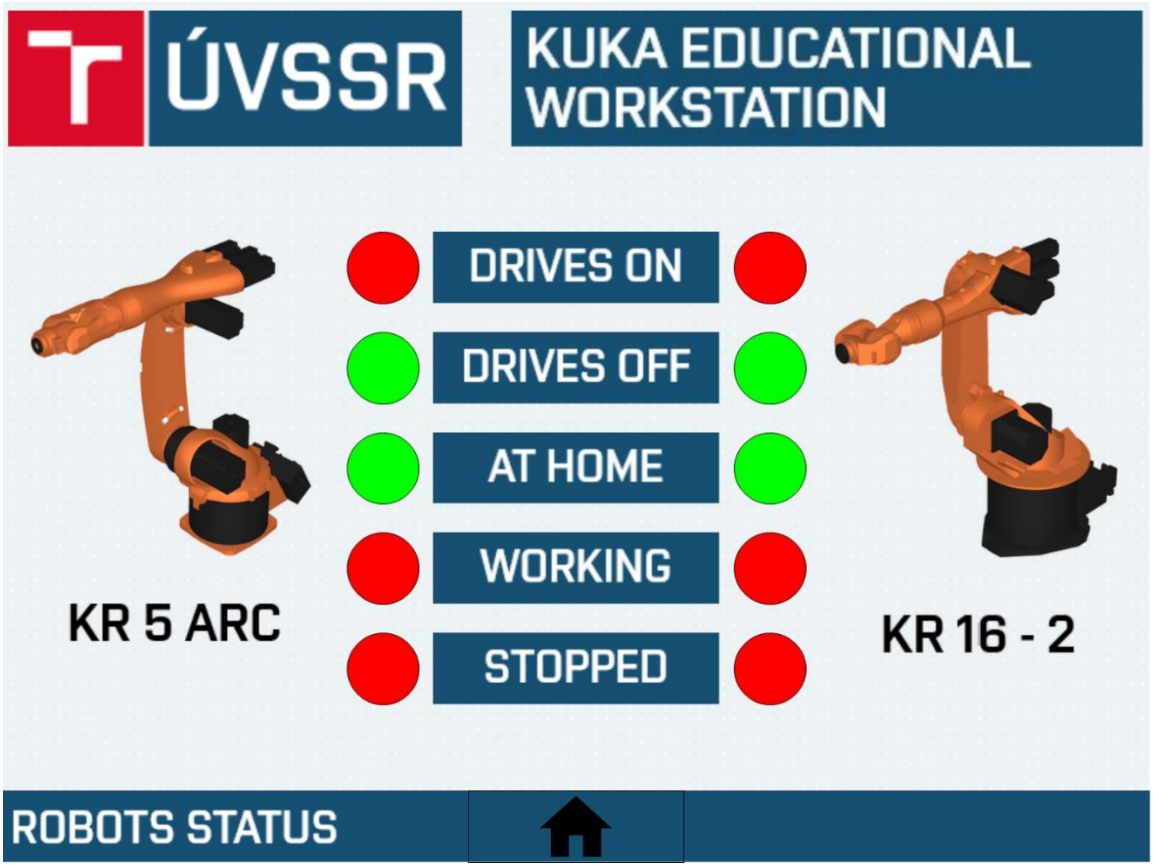
Obr. 57) Výchozí obrazovka „OVERVIEW“

Druhá obrazovkou HMI zvaná „ERRORS“ (obr. 58) je rozdělena na dvě poloviny. Levá polovina obsahuje signalizační světlo chyby, pod kterým se nachází textové pole, ve kterém je v případě výskytu chyby zobrazena její příčina a dále dvě tlačítka. Tlačítko „ACKNOWLEDGE“ plní funkci potvrzení přečtení důvodu chyby a tlačítko „RESET“ pro odstranění chyby a navrácení řídích programů do výchozího stavu. Při výskytu chyby je nejprve vyžadování její potvrzení prvním tlačítkem, které následně zpřístupní možnost resetu. Na levé straně obrazovky je signalizace jednotlivých safety prvků dostupných na pracovišti. Navrácení na výchozí obrazovku je možné tlačítkem s piktogramem domku na spodním panelu.

Třetí obrazovka HMI zvaná „ROBOTS STATUS“ (viz obr. 59) plní pouze kontrolní funkci. Ukazuje aktuální stav pěti vybraných signálů zpětné vazby robotického controlleru jak pro robot KR 5 arc i pro robot KR 16-2. Návrat na výchozí obrazovku je opět možný tlačítkem na spodním panelu.



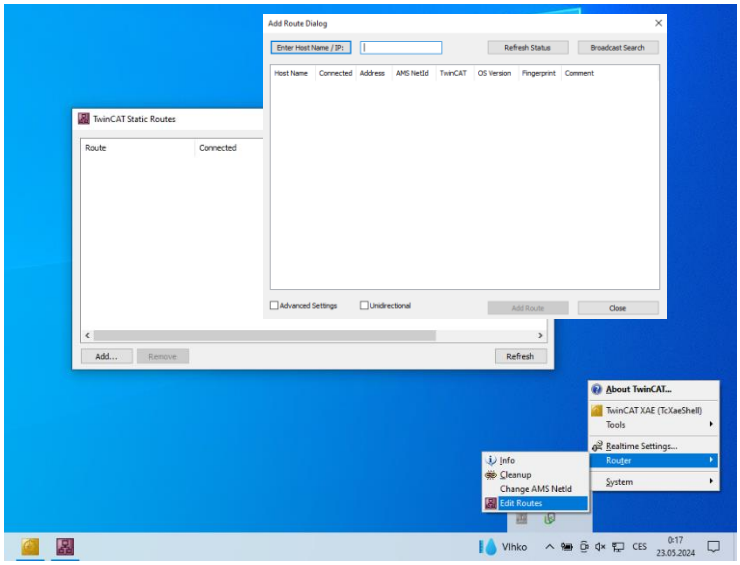
Obr. 58) Obrazovka „ERRORS“



Obr. 59) Obrazovka „ROBOTS STATUS“

8 REÁLNÉ ZPROVOZNĚNÍ

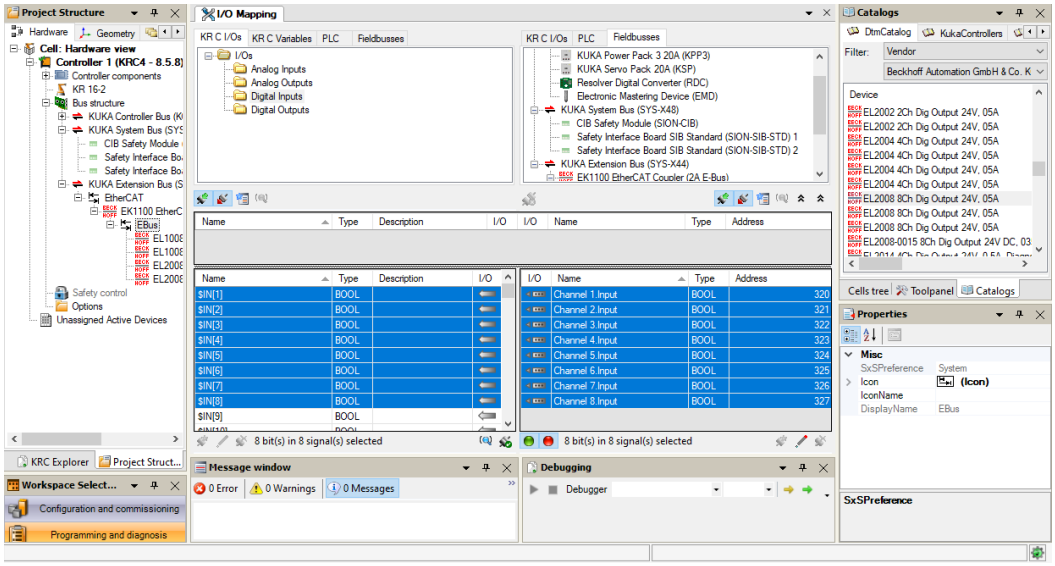
Prvním krokem reálného zprovoznění je nahrání buildu řízení vytvořeného v prostředí softwaru TwinCAT do reálného PLC controlleru na pracovišti. Nejprve je nutné propojit počítač, na kterém probíhal vývoj s IPC datovým ethernet kabelem. Jakmile se obě zařízení nachází v rámci jedné sítě, TwinCAT je schopný pomocí automatického vyhledávání *Broadcast search* (viz obr. 60) kontroler najít a vytvořit spojení. Následně je v TwinCAT možné změnit target device z <Local> na reálný controller. Rozbalením položky *I/O* v *Solution Exploreru* a kliknutím pravým tlačítkem na *Devices*, následně funkcí *Scan* je možno projektu připojit všechny I/O terminály přítomné na EtherCAT sběrnici, na které je následně potřeba namapovat proměnné.



Obr. 60) Propojení TwinCAT s PLC

Roboty KUKA je možné řídit nadřazeným PLC, pokud se robot nachází v „Automatic External Mode“. Dalším krokem přípravy je tedy namapování signálů KUKA controlleru, které jsou v tomto módu využívány, na adresy odpovídajícím adresám vstupních a výstupních terminálů. Mapování se provádí v prostředí softwaru WorkVisual (viz obr. 61). Z controlerů robotů je možné po připojení přes ethernetový kabel stáhnout zálohu aktuální používané verze konfigurace do počítače. Nová konfigurace s namapovanými proměnnými je následně zpětně nahrána do robotických controllerů a aktivována. [21]

Posledním krokem je nahrání robotických programů vyexportovaných z Process Simulate do robotických controllerů. Ty je možno stáhnout z PC na USB disk, z kterého je následně možné je nakopírovat do paměti controlleru. Následně je nutné v controlleru nakonfigurovat program „cell.src“, který se používá pro volání programů, a aktivovat jej. [21]



Obr. 61) Mapování signálů ve WorkVisual

Pro nakonfigurování programu „cell.src“ je potřeba být přepnutý v uživatelské skupině „Expert“. Program sestává ze „switch/case“ struktury, kdy každý „case“ stavového automatu reprezentuje program, který má být volán z paměti robotu. Výchozí podoba programu „cell.src“ je vidět na obr. 62. Počet stavů je možno upravit podle počtu programů, které pro daný robot budou volány. Robot KR 5 arc bude volat dva programy, proto stačí ponechat pouze dva stavy „CASE“. Pro robot KR16-2 byl vytvořen pouze jeden robotický program a z toho důvodu bude ponechán pouze jeden stav. Číslo stavu musí odpovídat číslu přirazenému robotického programu v kapitole 5.2, takže např. pro robot KR 5 arc budou stavy pojmenovány „CASE 10“ a „CASE 20“. Posledním krokem je přepsání „;EXAMPLEx“ na název programu, který bude pod daným číslem volán. [21]

```
1  DEF  CELL ( )
     ...
6    INIT
7    BASISTECH INI
8    CHECK HOME
9    PTP HOME  Vel= 100 % DEFAULT
10    AUTOEXT INI
11    LOOP
12      P00 (#EXT_PGNO,#PGNO_GET,DMY[],0 )
13      SWITCH  PGNO ; Select with Programnumber
14
15      CASE 1
16        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
17        ;EXAMPLE1 ( ) ; Call User-Program
18
19      CASE 2
20        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
21        ;EXAMPLE2 ( ) ; Call User-Program
22
23      CASE 3
24        P00 (#EXT_PGNO,#PGNO_ACKN,DMY[],0 )
25        ;EXAMPLE3 ( ) ; Call User-Program
26
27      DEFAULT
28        P00 (#EXT_PGNO,#PGNO_FAULT,DMY[],0 )
29      ENDSWITCH
30    ENDLOOP
31  END
```

Obr. 62) Struktura programu „cell.src“ [21]

9 ZHODNOCENÍ A DISKUZE

Proces virtuálního zprovoznění vyžaduje několik různých softwarů, což je zřejmé i z této práce. Vezmeme-li v potaz kolik je na trhu značek robotů, druhů řídících systémů a množství různých součástí robotických pracovišť, dostaneme nepřeberné množství kombinací prvků, kterými může být pracoviště tvořeno. Nejen potřebná úroveň odbornosti pracovníků vývoje, programátorů a integrátorů, ale i velký počet potřebného licencovaného softwaru často s vysokou cenovkou jsou důvody, proč se obzvlášť menší firmy virtuálnímu zprovoznění můžou vyhýbat.

Diplomová práce popisuje postup virtuálního a reálného zprovoznění robotického pracoviště. Virtuální zprovozňování robotizovaných linek a robotických pracovišť je dlouhodobým trendem v průmyslu, protože při správné aplikaci lze velmi snížit vstupní náklady při spojené s návrhem a integrací nového výrobního systému.

Jako nedostatek je možno považovat absence virtuálního controlleru robotů v rámci simulační smyčky. Software KUKA.OfficeLite potřebný pro virtualizaci controlleru je licencovaný a v době řešení práce jsem já, ani ústav licencí nedisponoval. Tato licence nebyla bohužel poskytnuta ani při požadavku na české zastoupení firmy. Controller tak musel být nahrazen programovým skriptem v Process Simulate, aby bylo možné získávat zpětnou vazbu od robotů. Zpětná vazba tak odpovídá realitě jenom v omezené míře.

Během vypracování práce byly splněny všechny požadavky, které byly vedoucím uloženy. Hlavní přínos práce vidím v možnosti jejího využití při vzdělávání budoucích studentů oboru Výrobní stroje, systémy a roboty na ÚVSSR. Práce by mohla posloužit jako podklad pro výuku některého z předmětů věnovaného robotice, nebo jako předloha pro zadání semestrálních projektů apod.

10 ZÁVĚR

První část této diplomové práce byla věnována přehledu současného stavu poznání, tedy rešerši problematiky spojené s virtuálním zprovoznění robotického pracoviště. Pojednává o současném trendu v automatizačním průmyslu, způsobech programování robotů a softwarů k programování využívaných. Byly popsány způsoby řízení robotického pracoviště controllerem robotu, pomocí nadřazeného PLC a řízení s využitím komunikačního protokolu OPC UA.

Druhá část práce byla určena popisu pracoviště, které bylo v rámci práce řešeno. Předmětem bylo provést zprovoznění výukového pracoviště v dílnách Ústavu výrobních strojů, systémů a robotiky nacházejícího se v hale C1 strojní fakulty Vysokého učení technického v Brně. V této části práce byly tedy popsány všechny podstatné součásti pracoviště jako jsou roboty, pásový dopravník, přípravky a řídící panel s rozváděčem. Popis komponent byl navíc doplněn fotografiemi.

V práci byla dále řešena příprava simulace chodu virtuálního pracoviště v programu Tecnomatix Process Simulate. Tato kapitola dokumentuje postup procesu vytvoření 3D modelu layoutu pracoviště, definování jednotlivých prvků, tvorbu robotických programů, zpracování materiálového toku simulace, konfiguraci řídících signálů a naprogramování skriptu pro získávání zpětné vazby od obou robotů během simulace. Nahrazení chybějícího virtuálního robotického controlleru programovým skriptem bohužel snížilo míru podobnosti simulace s realitou, avšak v době řešení práce neexistovala jiná možnost získávání zpětné vazby od robotů.

Následující část práce byla věnována zprovoznění komunikace mezi programem Process Simulate a programem TwinCAT 3, který je vývojovým prostředím pro programování řídícího PLC hardware společnosti Beckhoff. Komunikace byla zprostředkována protokolem OPC UA. V práci bylo podrobně popsáno vytvoření serveru pomocí doplňkového balíčku TF6100 pro TwinCAT 3, definice PLC proměnných pro přenos a postup propojení simulace se serverem.

Poslední část virtuálního zprovoznění byla věnována řídícím programům a tvorbě HMI. Podrobně popsány byly vytvořené funkční bloky nižší úrovně řízení pro ovládání dílčích funkcí robotů a stručně popsány programy vyšší úrovně řízení s odkazem na přílohu, ve které je pomocí vývojového diagramu popsán hlavní program řídící pracovní sekvenci. Dále byly popsány tři vytvořené obrazovky HMI.

Poslední kapitola práce byla věnována reálnému zprovoznění pracoviště. V jednotlivých krocích byl popsán postup připojení TwinCAT k reálnému PLC a nahrání konfigurace vytvořeného řízení, následně proces mapování proměnných robotického controlleru a postup zprovoznění automatického režimu robotu pro řízení z nadřazeného PLC.

11 SEZNAM POUŽITÝCH ZDROJŮ

[1] GRAU, Antoni; INDRI, Marina; LO BELLO, Lucia a SAUTER, Thilo. Robots in Industry: The Past, Present, and Future of a Growing Collaboration With Humans. Online. *IEEE industrial electronics magazine*. 2021, roč. 15, č. 1, s. 50-61. ISSN 1932-4529. Dostupné z: https://doi.org/10.1109/MIE.2020.3008136. [cit. 2024-01-28].

[2] International federation of robotics. Online. International federation of robotics. 2023. Dostupné z: https://ifr.org/. [cit. 2024-01-27].

[3] HEIMANN, Oliver a GUHL, Jan. Industrial Robot Programming Methods: A Scoping Review. Online. In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2020, s. 696-703. ISBN 172818956X. ISSN 1946-0740. Dostupné z: https://doi.org/10.1109/ETFA46521.2020.9211997. [cit. 2024-01-28].

[4] SANCHEZ RESTREPO, Susana; RAIOLA, Gennaro; CHEVALIER, Pauline; LAMY, Xavier a SIDOBRE, Daniel. Iterative virtual guides programming for human-robot comanipulation. Online. In: 2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM). IEEE, 2017, s. 219-226. ISBN 1509059989. Dostupné z: https://doi.org/10.1109/AIM.2017.8014021. [cit. 2024-05-24].

[5] BLECHA, P., Z. KOLÍBAL, R. KNOFLÍČEK, A. POCHYLÝ, T. KUBELA, R. BLECHA a T. BŘEZINA. Mechatronika: Modul 10: Robotika [online]. Brno: EU - Projekt č. DE/08/LLP-LdV/TOI/147110, 2008 [cit. 2023-05-29]. Dostupné z: https://docplayer.cz/8054951-Mechatronika-modul-10-robotika.html

[6] SICILIANO, Bruno a Oussama. KHATIB. Springer handbook of robotics. Berlin: Springer, 2008. ISBN 978-3-540-23957-4.

[7] KOLÍBAL,Z. a kol. Roboty a robotizované výrobní technologie. VUTIUM Brno, 2016, ISBN 978-80-214-4828-5.

[8] ABB RobotStudio. Online. ABB. Dostupné z: https://new.abb.com/products/robotics/robotstudio. [cit. 2024-03-21].

[9] KUKA.Sim. Online. KUKA Robotics Corporation. Dostupné z: https://www.kuka.com/en-us/products/robotics-systems/software/simulation-planning-optimization/kuka_sim. [cit. 2024-03-21].

[10] LECHLER, Tobias; FISCHER, Eva; METZNER, Maximilian; MAYR, Andreas a FRANKE, Jörg. Virtual Commissioning – Scientific review and exploratory use cases in advanced production systems. Online. Procedia CIRP. 2019, roč. 81, s. 1125-1130. ISSN 2212-8271. Dostupné z: https://doi.org/10.1016/j.procir.2019.03.278. [cit. 2024-02-14].

[11] STRIFFLER, Nikolai a VOIGT, Tobias. Concepts and trends of virtual commissioning – A comprehensive review. Online. Journal of manufacturing systems. 2023, roč. 71, s. 664-680. ISSN 0278-6125. Dostupné z: https://doi.org/10.1016/j.jmsy.2023.10.013. [cit. 2024-02-14].

[12] JACKSON, Chad. Hybrid digital twin for virtual commissioning. Online. Virtual commissioning. 2020. Dostupné z: https://virtualcommissioning.com/hybrid-digital-twin-for-virtual-commissioning/. [cit. 2024-02-15].

[13] STOCKLEIN, Jorg; GEIGER, Christian a PAELKE, Volker. Mixed reality in the loop - design process for interactive mechatronical systems. Online. In: *2010 IEEE Virtual Reality Conference (VR)*. IEEE, 2010, s. 303-304. ISBN 9781424462377. ISSN 1087-8270. Dostupné z: https://doi.org/10.1109/VR.2010.5444755. [cit. 2024-02-25].

[14] SCHNIERLE, Marc a RÖCK, Sascha. Latency and sampling compensation in mixed-reality-in-the-loop simulations of production systems. Online. Production engineering (Berlin, Germany). 2023, roč. 17, č. 3-4, s. 341-353. ISSN 0944-6524. Dostupné z: https://doi.org/10.1007/s11740-022-01175-2. [cit. 2024-02-25].

[15] PIECH, Zbyněk. Návrh oplocení a analýza rizik robotického pracoviště. Vysoké učení technické v Brně. Fakulta strojního inženýrství, 2022.

[16] KUKA KR 5 arc User Manual. Online. In: . Dostupné z: https://manualzz.com/doc/38563031/kuka-kr-5-arc. [cit. 2024-03-20].

[17] KUKA AG. KUKA KR 16-2 Brochure. Online. Dostupné z: https://roblab.ka.fu.tuke.sk/960-kuka-kr-16-2-robot-adatlap.pdf. [cit. 2024-03-11].

[18] KUKA AG. Robot Controller KR C4. Online. Dostupné z: https://www.kuka.com/en-de/products/robot-systems/robot-controllers/kr%C2%A0c4. [cit. 2024-03-11].

[19] KUKA AG. KUKA.OfficeLite. Online. Dostupné z: https://www.kuka.com/cs-cz/produkty,slu%C5%BEby/robotick%C3%A9syst%C3%A9my/software/pl%C3%A1nov%C3%A1n%C3%AD-projektov%C3%A1n%C3%AD-servisbezpe%C4%8Dnost/kuka_officelite. [cit. 2024-05-14].

[20] OPC FOUNDATION. OPC Technologies. Online. OPC Foundation. Dostupné z: https://opcfoundation.org/about/opc-technologies/opc-ua/. [cit. 2024-05-15].

[21] KUKA System Software 8.3: Operating and Programming Instructions for System Integrators. Online. In: . 2015. Dostupné z: http://www.wtech.com.tw/public/download/manual/kuka/krc4/KUKA%20KSS-8.3-Programming-Manual-for-SI.pdf. [cit. 2024-05-15].

12 SEZNAM ZKRATEK, OBRÁZKŮ A TABULEK

12.1 Seznam zkratek

PLC	Programmable Logic Controller
OLP	Off-Line Programming
3D	Three dimensional
CAD	Computer Aided Design
VR	Virtuální realita
VZ	Virtuální zprovoznění
KRL	KUKA Robot Language
OPC	Open Platform Communcations
OPC UA	Open Platform Communications
HMI	Human-Machine Interface
TCP/IP	Transmission Control Protocol/Internet Protocol
IP	Internet_Protocol
AR	Augmented Reality
IPC	Industriální počítač
VRC	Virtual Robot Controller
SCL	Structured Control Language
ÚVSSR	Ústav výrobních strojů, systémů a robotiky

12.2 Seznam tabulek

12.3 Seznam obrázků

13 SEZNAM PŘÍLOH

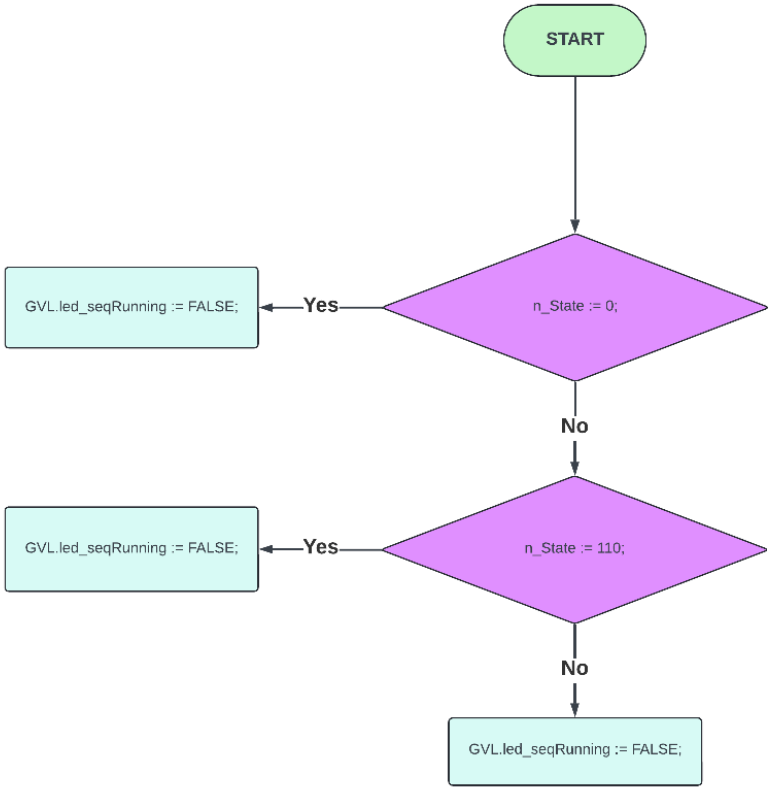
PŘÍLOHY

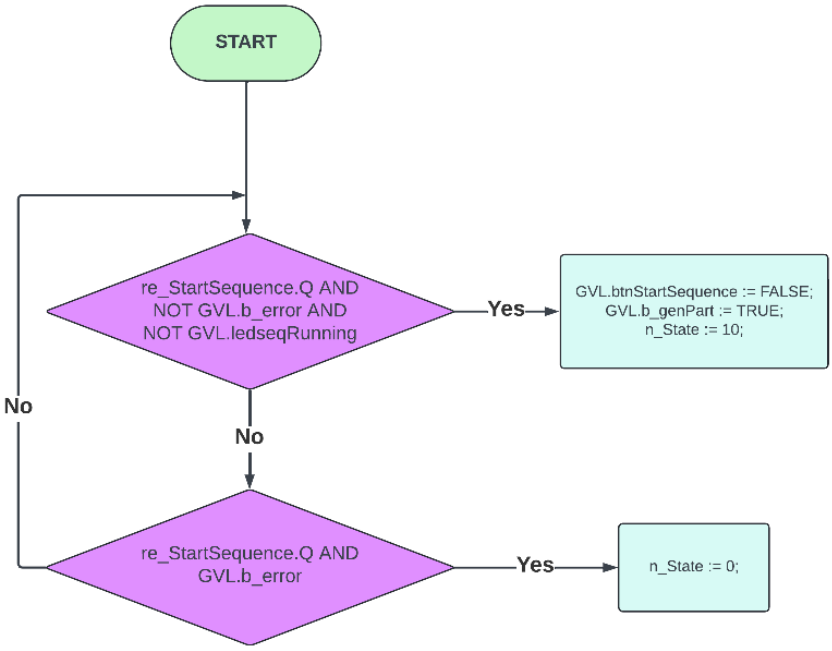
1. Signály použité v PS simulaci

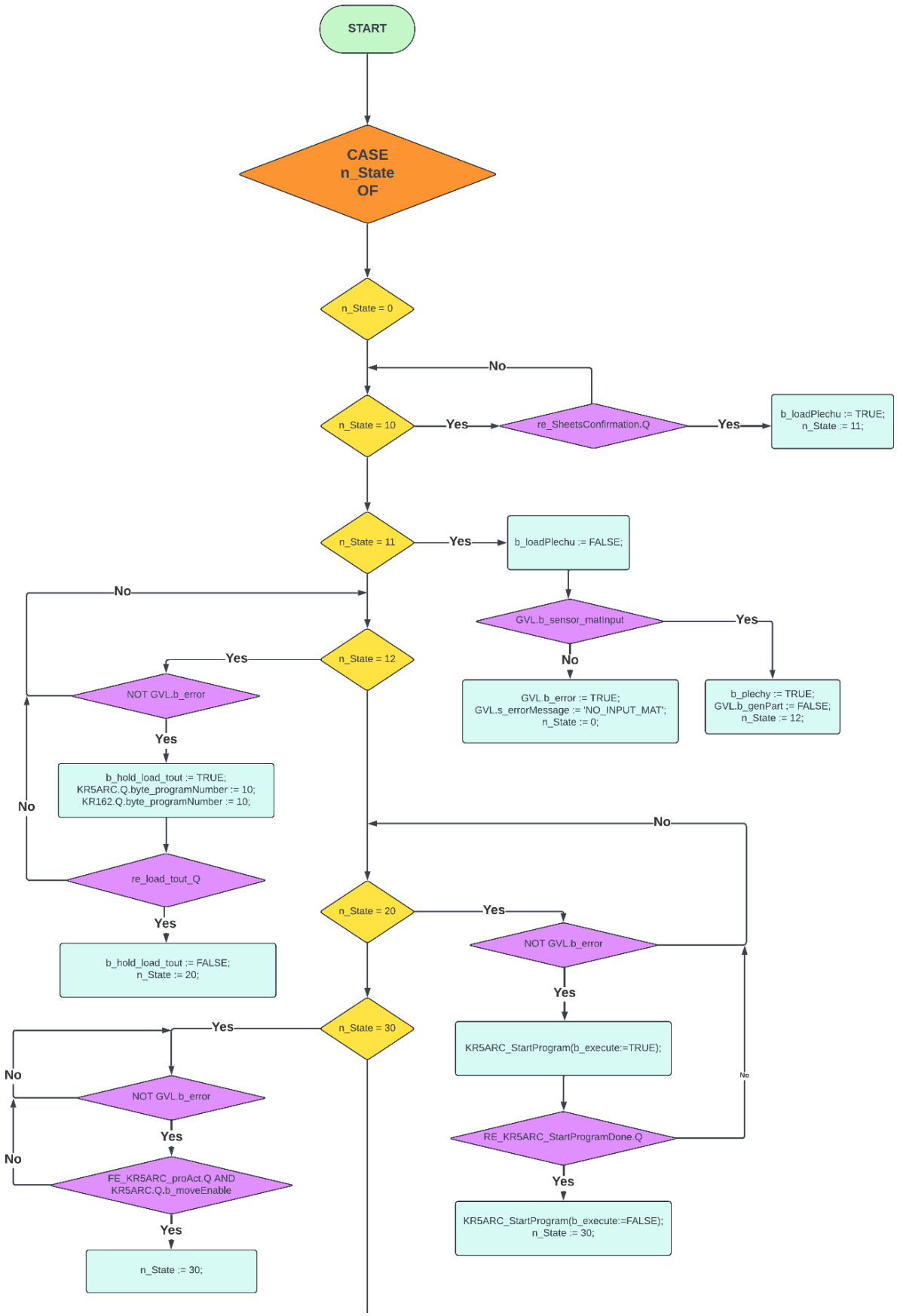
Signal Name	Type	Robot Signal Name	IEC Format	Resource
GVL.b_genPart	BOOL		Q	
GVL.b_destroyPart	BOOL		Q	
GVL.b_genPartSkid	BOOL		Q	
GVL.b_sensor_conveyor	BOOL		I	sensor_conveyor
GVL.b_sensor_matInput	BOOL		I	sensor_matInput
GVL.b_conveyorStart	BOOL		Q	Conveyor
GVL.b_conveyorStop	BOOL		Q	Conveyor
KR5ARC.Q.b_drivesOn	BOOL	$DRIVES_ON	Q	kr5_arc
KR5ARC.Q.b_drivesOff	BOOL	$DRIVES_OFF	Q	kr5_arc
KR5ARC.Q.b_moveEnable	BOOL	$MOVE_ENABLE	Q	kr5_arc
KR5ARC.I.b_proAct	BOOL	$PRO_ACT	I	kr5_arc
KR5ARC.I.b_periRdy	BOOL	$PERI_RDY	I	kr5_arc
KR5ARC.I.b_robStopped	BOOL	$ROB_STOPPED	I	kr5_arc
KR5ARC.I.b_proMove	BOOL	$PRO_MOVE	I	kr5_arc
KR5ARC.Q.b_startProgram	BOOL	startProgram	Q	kr5_arc
KR5ARC.Q.byte_programNumber	BYTE	programNumber	Q	kr5_arc
KR5ARC.Q.b_emergencyStop	BOOL	emergencyStop	Q	kr5_arc
KR5ARC.Q.b_programPause	BOOL	programPause	Q	kr5_arc
KR5ARC.Q.byte_reduceSpeed	BYTE	reduceSpeed	Q	kr5_arc
KR5ARC.I.b_programEnded	BOOL	programEnded	I	kr5_arc
KR5ARC.I.byte_mirrorProgramNumber	BYTE	mirrorProgramNumber	I	kr5_arc
KR5ARC.I.b_errorProgramNumber	BOOL	errorProgramNumber	I	kr5_arc
KR5ARC.I.b_robotReady	BOOL	robotReady	I	kr5_arc
KR5ARC.I.b_at_SERVICE	BOOL	SERVICE	I	kr5_arc
KR5ARC.I.b_at_HOME	BOOL	HOME	I	kr5_arc
KR162.Q.b_drivesOn	BOOL	$DRIVES_ON	Q	kr16_2
KR162.Q.b_drivesOff	BOOL	$DRIVES_OFF	Q	kr16_2
KR162.Q.b_moveEnable	BOOL	$MOVE_ENABLE	Q	kr16_2
KR162.I.b_at_SERVICE	BOOL	SERVICE	I	kr16_2
KR162.I.b_proAct	BOOL	$PRO_ACT	I	kr16_2
KR162.I.b_periRdy	BOOL	$PERI_RDY	I	kr16_2
KR162.I.b_robStopped	BOOL	$ROB_STOPPED	I	kr16_2
KR162.I.b_proMove	BOOL	$PRO_MOVE	I	kr16_2
KR162.Q.b_startProgram	BOOL	startProgram	Q	kr16_2
KR162.Q.byte_programNumber	BYTE	programNumber	Q	kr16_2
KR162.Q.b_emergencyStop	BOOL	emergencyStop	Q	kr16_2
KR162.Q.b_programPause	BOOL	programPause	Q	kr16_2
KR162.Q.byte_reduceSpeed	BYTE	reduceSpeed	Q	kr16_2
KR162.I.b_programEnded	BOOL	programEnded	I	kr16_2
KR162.I.byte_mirrorProgramNumber	BYTE	mirrorProgramNumber	I	kr16_2

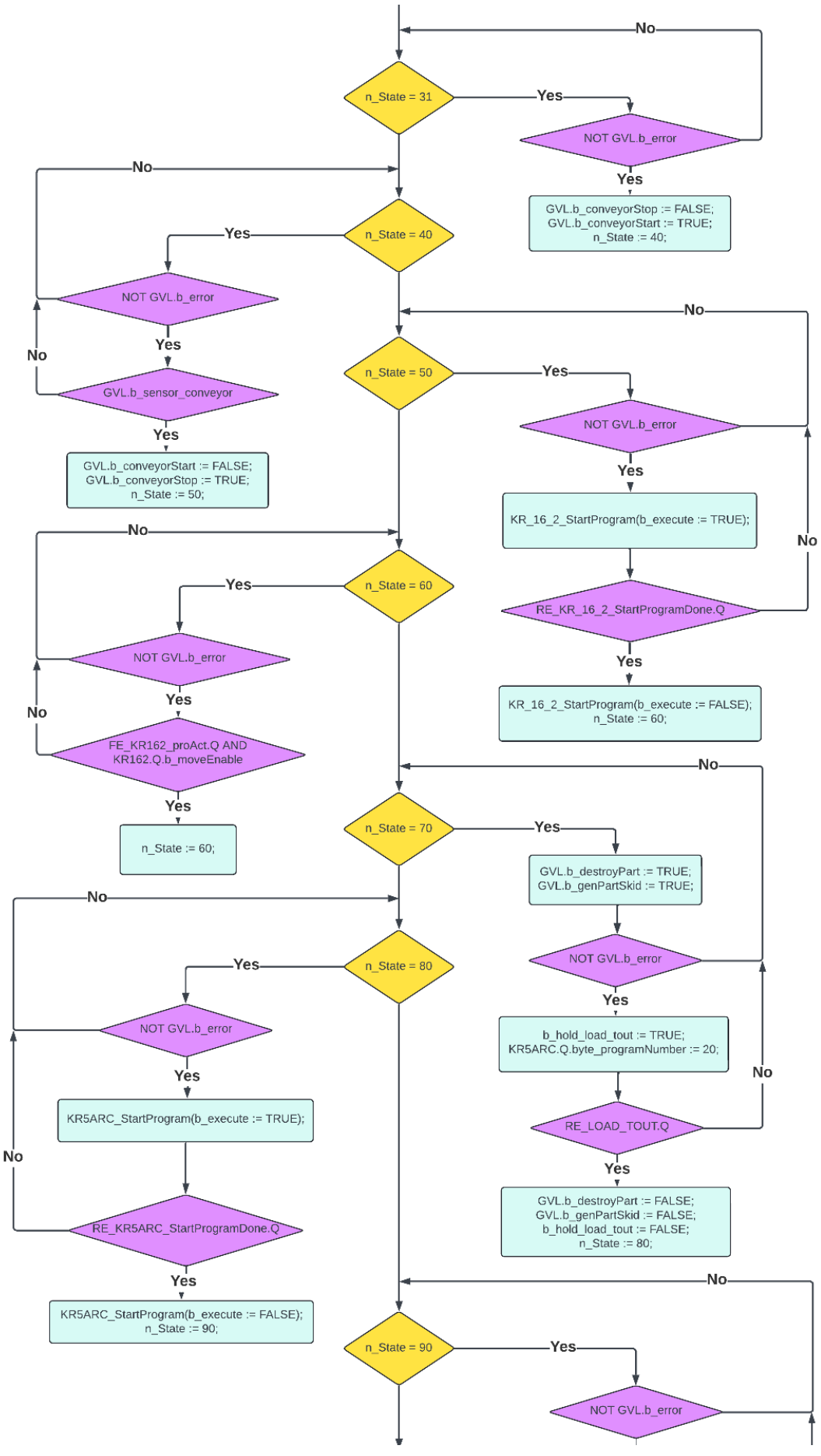
KR162.I.b_errorProgramNumber	BOOL	errorProgramNumber	I	kr16_2
KR162.I.b_robotReady	BOOL	robotReady	I	kr16_2
KR162.I.b_at_HOME	BOOL	HOME	I	kr16_2

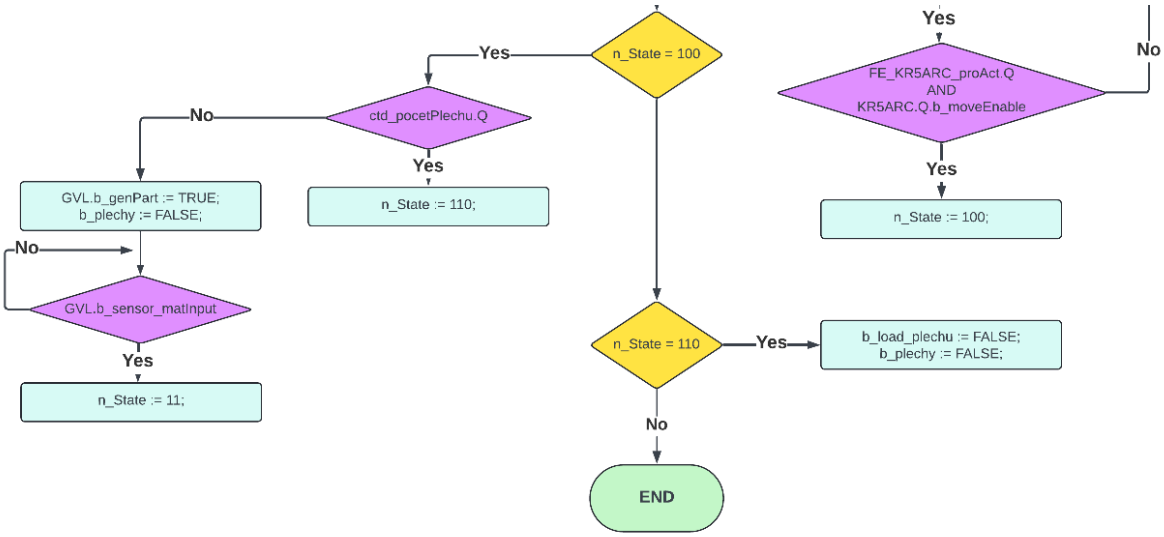
2. Program „ROB_SEQ“






START
CASE n_State OF
n_State = 0
No
n_State = 10
Yes
re_SheetsConfirmation.Q
Yes
b_loadPlechu := TRUE;
n_State := 11;
n_State = 11
Yes
b_loadPlechu := FALSE;
No
GVL.b_sensor_matInput
Yes
No
GVL.b_error := TRUE;
GVL.s_errorMessage := 'NO_INPUT_MAT';
n_State := 0;
b_plechy := TRUE;
GVL.b_genPart := FALSE;
n_State := 12;
n_State = 12
Yes
NOT GVL.b_error
Yes
b_hold_load_tout := TRUE;
KR5ARC.Q.byte_programNumber := 10;
KR162.Q.byte_programNumber := 10;
No
re_load_tout_Q
Yes
b_hold_load_tout := FALSE;
n_State := 20;
No
n_State = 20
Yes
NOT GVL.b_error
Yes
KR5ARC_StartProgram(b_execute:=TRUE);
RE_KR5ARC_StartProgramDone.Q
No
Yes
KR5ARC_StartProgram(b_execute:=FALSE);
n_State := 30;
n_State = 30
Yes
No
NOT GVL.b_error
Yes
No
FE_KR5ARC_proAct.Q AND
KR5ARC.Q.b_moveEnable
Yes
n_State := 30;


n_State = 31
Yes
No
NOT GVL.b_error
Yes
GVL.b_conveyorStop := FALSE;
GVL.b_conveyorStart := TRUE;
n_State := 40;
n_State = 40
Yes
No
NOT GVL.b_error
Yes
No
GVL.b_sensor_conveyor
Yes
GVL.b_conveyorStart := FALSE;
GVL.b_conveyorStop := TRUE;
n_State := 50;
n_State = 50
Yes
No
NOT GVL.b_error
Yes
KR_16_2_StartProgram(b_execute := TRUE);
RE_KR_16_2_StartProgramDone.Q
No
Yes
KR_16_2_StartProgram(b_execute := FALSE);
n_State := 60;
n_State = 60
Yes
No
NOT GVL.b_error
Yes
No
FE_KR162_proAct.Q AND
KR162.Q.b_moveEnable
Yes
n_State := 60;
n_State = 70
Yes
No
GVL.b_destroyPart := TRUE;
GVL.b_genPartSkid := TRUE;
NOT GVL.b_error
Yes
No
b_hold_load_tout := TRUE;
KR5ARC.Q.byte_programNumber := 20;
RE_LOAD_TOUT.Q
Yes
No
GVL.b_destroyPart := FALSE;
GVL.b_genPartSkid := FALSE;
b_hold_load_tout := FALSE;
n_State := 80;
n_State = 80
Yes
No
NOT GVL.b_error
Yes
No
KR5ARC_StartProgram(b_execute := TRUE);
RE_KR5ARC_StartProgramDone.Q
Yes
KR5ARC_StartProgram(b_execute := FALSE);
n_State := 90;
n_State = 90
Yes
No
NOT GVL.b_error


n_State = 100
Yes
ctd_pocetPlechu.Q
No
Yes
n_State := 110;
GVL.b_genPart := TRUE;
b_plechy := FALSE;
No
GVL.b_sensor_matInput
Yes
n_State := 11;
Yes
FE_KR5ARC_proAct.Q
AND
KR5ARC.Q.b_moveEnable
No
Yes
n_State := 100;
n_State = 110
Yes
b_load_plechu := FALSE;
b_plechy := FALSE;
No
END