Mendelova univerzita v Brně

Řídicí systém modulu pro ovládání řezacího centra

Diplomová práce

Vedoucí práce:
Dr. Ing. Radovan Kukla

Bc. Vojtěch Hemala

Brno 2015

Zadani DP

Rád bych vyjádřil poděkování vedoucímu této práce, panu Dr. Ing. Radovanu Kuklovi za cenné připomínky, odborné rady a podněty k zamyšlení během vypracování diplomové práce.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Řídicí systém modulu pro ovládání řezacího centra** vypracoval/a samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*

Jsem si vědom/a, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 2. ledna 2015 _____

Abstract

Hemala, V. *Modul control system to drive cutting machine*. Diploma thesis. Brno 2015.

Diploma thesis deals with the design and implementation of control application that drives cutting machine module. Goal of application is controlling implemented module by terminal touch-panel.

The thesis begins with the theoretical analysis of cutting machine and hardware description, which involves sensors, actuators, regulating elements and programmable controller. Ensued part of this thesis makes survey within industrial communication in automatization and selection of suitable software development tool.

Implementation of application is carried out in practical part which describes source code samples and user interface visualization screens of touch-panel. Final part is dedicated to summary of results and possible extension.

Keywords

Industrial automatization, Automation Studio, B&R, Powerlink Ethernet, modular systems

Abstrakt

Hemala, V. *Řídicí systém modulu pro ovládání řezacího centra.* Diplomová práce. Brno 2015.

Diplomová práce se zabývá návrhem a implementací řídicí aplikace pro ovládání modulu řezacího centra. Cílem aplikace je ovládání implementovaného modulu z dotykového operátorského panelu.

Práce začíná teoretickým rozborem řezacího centra a popisem technického vybavení, které zahrnuje snímače, akční členy, regulační prvky a programovatelný automat. Následuje průzkum oblasti průmyslové komunikace v automatizaci a výběr vhodného vývojového prostředí.

V praktické části je provedena samotná implementace aplikace, kde jsou ukázky zdrojového kódu aplikace a vizualizační obrazovky uživatelského rozhraní dotykového panelu. Závěr práce je věnován shrnutím dosažených výsledků a možným rozšířením.

Klíčová slova

Průmyslová automatizace, Automation Studio, B&R, Powerlink Ethernet, modulární systémy

Obsah

Seznam obrázků

Seznam zkratek

CAL	CAN Application Layer
CBA	Component Based Automation
CD	Collision Detection
CFC	Continuous Function Chart
CSMA	Carrier Sense Multiple Access
DI/DO	Digital Input/Digital Output
DTC	Direct Torque control
EMC	Electro Magnetic Compatibility
EPSG	Ethernet Powerlink Standardization Group
FI	Frequence Inverter
HMI	Human-Machine Interface
IAONA	Industrial Automation Open Networking Alliance
IEC	International Electrotechnical Commission
IGBT	Insulated Gate Bipolar Transistor
ISO	International Organization for Standardization
MEMS	Micro-Electro-Mechanical Systems
OSI	Open System Interconnection
PAC	Programmable Automation Controller
PLC	Programmable Logic Controller
PNO	Profibus Nutzerorganisation
POU	Program Organisation Unit
SCADA	Supervisory Control And Data Acquisition
SCNM	Slot Communication Network Management
SDO/PDO	Service Data Object/Process Data Object
SVC	Sensorless vector control
TCP/UDP	Transmission Control Protocol/User Datagram Protocol
TFT	Thin Film Transistor
TON/TOF	Timer On Delay/Timer Off Delay

TTL	Transistor-Transistor Logic
VGA	Video Graphics Array
WYSIWYG	What You See Is What You Get

1 Úvod a cíl práce

1.1 Úvodní slovo

Každý dnes fungující podnik má za hlavní cíl vykazovat maximální zisk při minimálních nákladech. Cest k tomuto je vícero, avšak stále platí, že čím více lidí je zapojeno v komunikačním řetězci mezi objednáním a doručením výrobku, tím větší je riziko selhání lidského faktoru a tedy i zpoždění vyrobení a následně i doručení výrobku. Toto je většinou spojeno se snížením zisku.

Je tedy žádoucí, aby se možnost lidské chyby snížila na minimum. Proto se začínají lidé na některých úrovních nahrazovat automatizovanými prostředky, či nasazují se takové systémy, které výrazně pomáhají pracovníkům při jejich bezchybné činnosti.

Ve všech odvětvích stále roste podíl systémových řešení včetně automatizace. Zastavit technologii, na které jsme závislí a která je pro lidstvo skutečným přínosem, nelze. Stačí se rozhlédnout kolem sebe. Vše je alespoň z převážné většiny vytvořeno na automatických strojích. Naše boty, oblečení, domácí spotřebiče, všechny možné typy mobilních zařízení, auta atd.

V poslední době se automatizace ve výrobě rozšířila i do oblastí, kde jsou pracovní síly poměrně levné. Čínské společnosti v roce 2011 utratily za průmyslovou automatizaci 1,3 miliardy dolarů. Firma Foxconn, která pro Apple sestavuje iPady doufá, že první plně automatizovanou továrnu uvede do provozu v příštích 5 - 10 letech [Skidelsky, 2013].

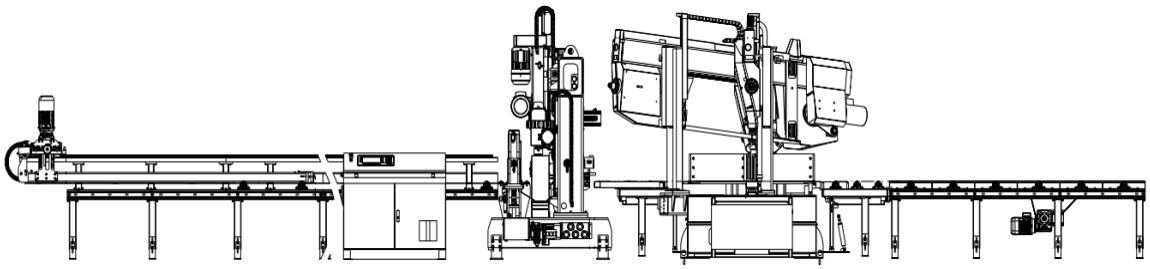
Myšlenka úplné automatizace není nová. Už počátkem 19. století uvažoval *David Ricardo*, významný britský ekonom, o možnosti nahrazení lidské práce stroji. V té době zaměstnanci rozbíjeli stroje, protože jim braly zaměstnání. Postupem času se vytvořila nová pracovní místa, která byla lépe placena, a tak se tento nešvar v podobě ničení strojů postupem času vytratil. Jenomže tento strach z automatizovaných strojů byl opodstatněný. Jistý je totiž fakt, že pracovní místa opravdu jednou dojdou. Automatizované linky a robotická pracovní síla jsou mnohem rychlejší, efektivnější a levnější než práce lidská. Roboti mohou pracovat 24 hodin denně, sedm dní v týdnu, bez přestávek na obědy, nepotřebují platit sociální a zdravotní pojištění, nejsou nemocní a ani si neříkají o větší plat.

1.2 Cíl práce

Cílem práce je návrh a následná implementace řídicího systému modulu pro ovládání řezacího centra. Řídicím systémem je myšlena komplexní aplikace, která v sobě zahrnuje jednak programovou část, s řízením veškerých akčních členu s využitím senzorického systému a také část vizualizační, která obnáší vytvoření odpovídající intuitivní uživatelské vizualizace pro vzdálené ovládání modulu z operátorského dotykového panelu.

2 Technologie řezacího centra

Řezací centrum lze stručně popsat jako multifunkční automatizovaná strojní linka pro zpracování materiálu na základě definované receptury obsluhou stroje. Multifunkční znamená, že je schopna vykonávat několik úkonu zároveň, aniž by bylo potřeba materiál nějak přenášet či přepravit na jiné pracoviště. V tomto konkrétním případě jsou to úkony: vrtání děr a řezání materiálu. Řezání a vrtání pomocí jedné strojní linky přináší vysokou univerzálnost při jejím použití, relativně vysokou přesnost vrtu/řezu a v porovnání s cenami komerčních řešení, kdy je pro stejný úkon potřeba více strojů, kabeláže, obsluhy je celá technologie dostupná za výrazně nižší náklady. Činnost linky je automatická.

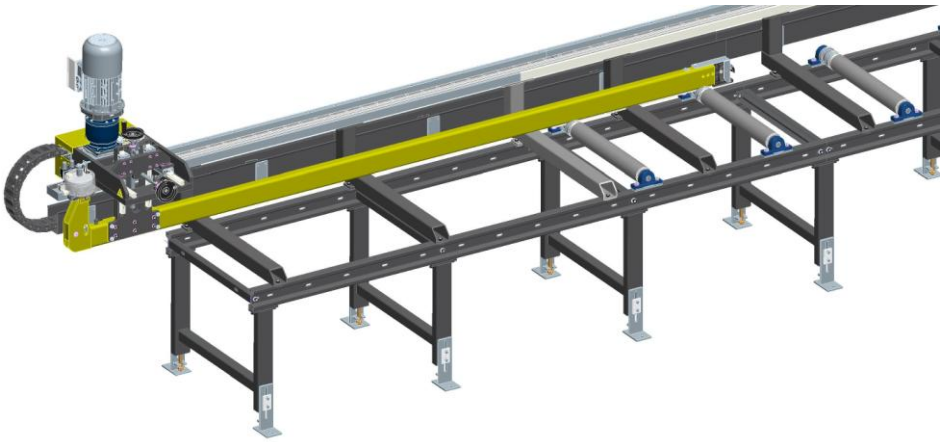


Obr. 1 Výkresová dokumentace kompletního řezacího centra

Samotné řezací centrum se skládá z několika částí – modulů (viz Obr. 1), které jsou vzájemně propojeny a během pracovního cyklu spolu neustále komunikují. Na začátku zpracovatelského řetězce se nachází vstupní válečková trať s poháněnou kleštinou, která dávkuje materiál po válečkové trati a zajišťuje jeho plynulý tok podle požadavků a daných podmínek k dalším modulům. Prostřední modul neboli vrtací centrum zajišťuje vrtání pomocí třech vrtacích jednotek. Vrtací centrum je schopno obrábět materiál ve dvou osách zároveň. Na konci celé linky se nachází pásová pila s hydraulickým natáčením, která si nechá podle receptury navézt kleštinou požadovanou délku a uřízne. Poté se již obrobený kus odveze po výstupním dopravníku na další zpracování. Před jakýmkoliv pohybem musí být všechny moduly zreferovány.

V automatickém cyklu kleština na začátku nejprve nalezne konec materiálu, upne jej do čelistí upínače a pomocí laserové závory najde jeho začátek. Poté podle povelu z nadřazeného řízení – *frameworku* s ním nadále manipuluje. Tato část – vstupní trať společně s výstupním dopravníkem jsou jediná pracoviště, do kterých může nebo spíše musí zvnějšku zasáhnout lidská ruka, a to ve smyslu navezení materiálu na válečkovou trať a jeho odstranění z výstupního dopravníku po jeho zpracování. K tomu, aby byla kleština (Obr. 2) schopna navézt požadovanou délku, potřebuje vědět, kde se materiál na válečkové trati nachází. Stejně jako člověk pro svou orientaci v prostoru nebo rozpoznávání předmětů využívá základních lidských smyslů jako zrak, čich, hmat jsou i v technice přijímající podněty z vnějšího a vnitřního prostředí velmi důležité a bez jejich přítomnosti by byli schopnosti automatizace značně omezené. Tyto podněty nám zprostředkovávají

nejrůznější senzory a snímače. Kleština pro detekci materiálu v upínacích čelistech využívá snímač indukčnostní, další indukčnostní snímač je využit pro detekci otevřených upínacích čelistí. Pro přesné odměřování pozice kleštiny po vstupní válečkové trati je použita lineární magnetická páska. Pohyb kleštiny po válečkové trati je zajištěn asynchronním třífázovým motorem řízeným měničem frekvence. Motor převádí hnací sílu pomocí ozubeného pastorku, jenž se pohybuje po ozubeném hřebeni umístěném po celé délce válečkové tratě.

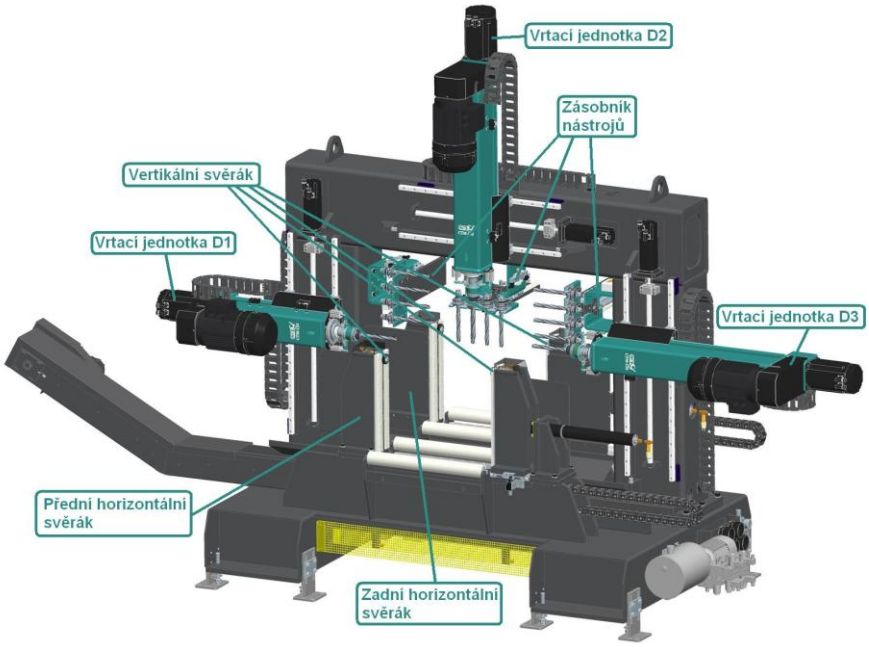


Obr. 2 Vstupní válečková trať s kleštinou

Při navážení materiálu do vrtacího centra musí být všechny vrtací jednotky ve svých výchozích pozicích, jak je ukázáno na Obr. 3. Hydraulické svěráky musí být max. otevřený, což je detekováno koncovými spínači. Každá vrtací jednotka má automatický výměník nástrojů pro až 4 vrtáky, i ty musí být ve výchozí poloze (zasunuto), což detekuji snímače na pneumatických válcích. Limitní pozice každé vrtací jednotky jsou hlídány koncovými spínači.

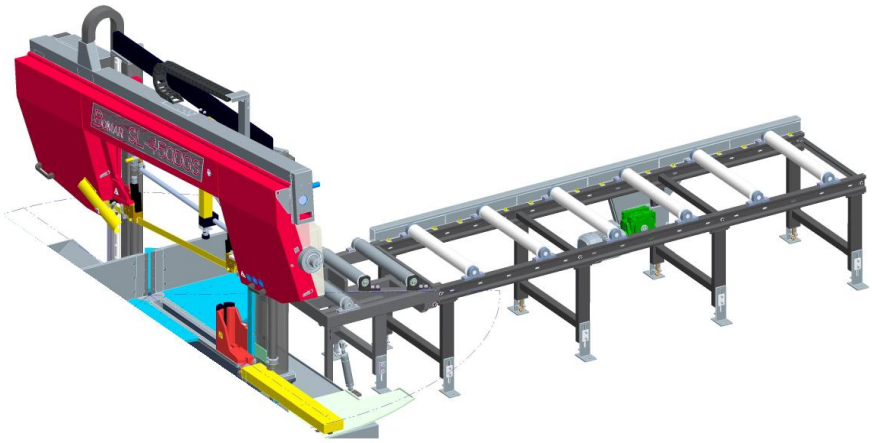
Po navezení požadované délky je materiál upnut horizontálním hydraulickým svěrákem, následně pak svěrákem vertikálním, který by měl zamezit případnému chvění materiálu při vrtu. Pevné a stálé upnutí materiálu horizontálním svěrákem zajišťuje koncový tlakový snímač. Poté již probíhá cyklus vrtání díry/děr podle předem definované receptury. Všechny vrtací jednotky by měli být naprogramovány tak, aby mohli vrtat nezávisle na sobě, tzn., že pokud vrtá v jedné hladině více vrtaček, nečekají vzájemně na sebe, až jedna dokončí cyklus, aby mohla začít jiná, ale pracují nezávisle na sobě. Takovýto postup značně urychlí celý pracovní cyklus a zefektivní vytíženost stroje. Stejný princip by měl fungovat i u výměny nástrojů, kdy vrtací jednotka po načtení nové díry z receptury si sama rozhodne, pokud má jiný nástroj ve vrtací hlavě, že jej vymění. Zásobník nástrojů je upevněn na pohyblivé pneumatické pístnici, kde koncové polohy jsou hlídané snímači na válcích. Při nastavování parametrů jednotlivých vrtáků je třeba dbát zvýšené pozornosti, poněvadž veškeré parametry – jejich délka, průměr, řezné podmínky, posuv,… jsou využívány kromě vrtání i pro řízení vzájemných kolizí mezi vrtacími jednotkami. Nesprávný či zaměněný parametr by mohl mít fatální následek při pracovním cyklu a možné poškození vrtacího centra. Vertikální

a horizontální pohyb vrtacích jednotek je zajištěn servopohony, kde každá jednotka je vybavena právě dvěma (pro pohyb v obou souřadných osách). Samotný rotační pohyb vrtáku, který je upnut ve vřetenu vykonává třífázový asynchronní motor řízený měničem frekvence. Celkem tedy šest servopohonů a tři motory řízené třemi měniči frekvence. Chlazení a zároveň mazání vrtáků probíhá pomocí mikronizéru, který je řízen ventilem a pomocí dopravního vedení je chladicí emulze dopravena až k vrtací hlavě.



Obr. 3 Popis jednotlivých částí vrtacího centra

Po vyvrtání všech děr a splnění dalších omezujících podmínek pro ukončení vrtacího cyklu, je povolen horizontální svěrák, poté i svěrák vertikální a obrobený materiál je kleštinou odvezen k pásové pile (viz Obr. 4) na uřezání definované délky. I zde platí, že pila před jakýmkoliv pohybem musí být zreferována a při navážení materiálu do stroje je nutné, aby rameno bylo ve své horní poloze a upínací svěrák maximálně otevřený. Před započetím řezání musí proběhnout kontrola napnutí pásu. Napínání pásu je řešeno hydraulickým válcem a probíhá vždy při rozběhu čerpadla agregátu. Při jeho vypnutí se pás pily vlivem poklesu tlaku v hadicích po chvíli opětovné povolí. Úhel ramene je snímán inkrementálním snímačem s nulovou značkou umístěným v ose natáčení ramene. Zvedání ramene pily, přesun svěráku pily, aretace natáčení ramene jsou řízený hydraulickým ventilem. Pokud je řezaný kus dostatečně dlouhý tak, že jeho poměrná část leží na výstupním dopravníku, tak se po uříznutí kleština vrací zpět do výchozí polohy a výstupní dopravní odveze hotový kus k dalšímu zpracování. Avšak pokud je řezaná délka malá, je nutné kleštinou vyvézt tento kus za tzv. mrtvé pásmo, jenž je ohraničené vzdáleností mezi pásem pily a prvním poháněným válečkem výstupního dopravníku.



Obr. 4 Natáčecí pásová pila s výstupním dopravníkem

2.1 Analýza senzorického systému

2.1.1 Význam, vývoj a určení senzorů

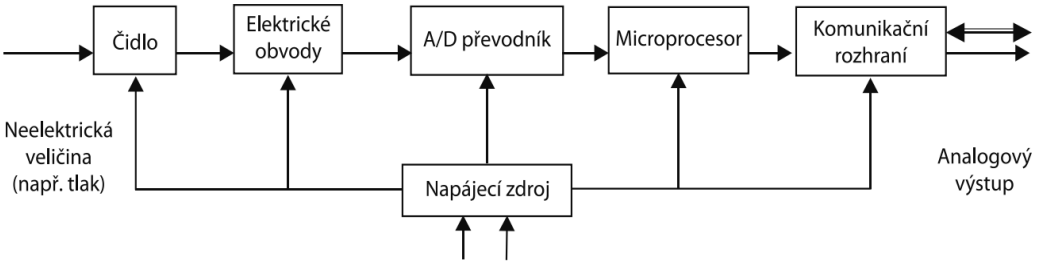
Systémy automatického řízení a informatiky se neustále rozvijí a aplikují v průmyslu, ve službách, ve zdravotnictví i v domácnostech. Tyto systémy mají své funkce založené na kvalitních vstupních datech a informacích, tj. na kvalitním podsystému měření nebo snímání. Proto jsou senzory důležitou součástí většiny moderních automatizovaných soustav a zařízení. V průmyslové automatizované výrobě je obvykle nutné detekovat přítomnost objektu, předmětu, kapaliny v určitých místech stroje nebo linky a podle toho řídit a časovat práci různých částí stroje. K tomu je však nutné vhodně zvolit typ senzoru tak, aby byl schopen opakovaně zajistit spolehlivou detekci. Volbu je tak nutné provést v závislosti nejen na fyzikálních vlastnostech detekovaných objektů, ale i pracovního prostředí.

Vývoj měřicí a senzorové techniky probíhal v generačních vlnách, které lze např. charakterizovat takto [ZEHNULA 1996]:

1. generace – klasické snímače bez normalizovaného výstupního elektrického signálu (např. termočlánek, tenzometr),
2. generace – mikro a opto-elektronické snímače s normalizovaným proudovým výstupním signálem (např. tlakový Si-senzor, optické vláknové senzory),
3. generace – inteligentní (SMART) senzory (viz Obr. 5), videoprocesorové subsystémy

Senzor je funkční prvek tvořící vstupní blok měřicího řetězce, který je v přímém styku s měřeným prostředím. Pojem senzor je ekvivalentní pojmu snímač nebo detektor. Citlivá část senzoru se občas označuje jako čidlo. Senzor jako primární zdroj informací snímá sledovanou fyzikální, chemickou nebo biologickou veličinu a dle určitého definovaného principu ji transformuje na měřicí veličinu – nejčastěji na veličinu elektrickou. Dále existují senzory, u nichž je neelektrická veličina přímo transformována na číslicový signál. Rychle postupující

vývoj mikroelektroniky napomohl rozšíření systémové schopnosti senzorů. Ty se postupně mění na tzv. inteligentní a kompaktní měřicí systémy s vestavěnými funkcemi zpracování signálu a specifickými možnostmi komunikace, jako např. rozhraní umožňující přenos naměřených dat bezdrátovou technologií [ĎAĎO, KREIDL 2000].



Obr. 5 Schéma inteligentního senzoru
[zdroj: http://coptel.coptkm.cz/reposit.php?action=2&id=9211]

Senzory můžeme dělit podle:

- **měřené veličiny** na senzory teploty, tlaku, průtoku, mechanických veličin (posunutí, polohy, rychlosti, zrychlení, síly, mechanického napětí aj.), senzory elektrických a magnetických veličin aj.
- **fyzikálního principu** na senzory odporové, indukčnostní, kapacitní, magnetické, piezoelektrické, optoelektronické, optické vláknové aj.
- **stylu senzoru** s měřeným prostředím na bezdotykové a dotykové (proximitní a taktilní)
- **tvaru výstupní veličiny** na spojité (analogové) a diskrétní (nespojité)

2.1.2 Metody pro zmenšení chyb senzorů

V praxi nejsou žádné měření, žádná měřicí metoda ani žádný přístroj absolutně přesné. Nejrůznější negativní vlivy, které se v reálném měřicím procesu vyskytují, se projeví odchylkou mezi naměřenou a skutečnou hodnotou sledované veličiny. Výsledek měření se tak vždy pohybuje v jistém tolerančním poli kolem skutečné hodnoty, ale téměř nikdy nenastává ideální ztotožnění obou hodnot.

Senzory při reakci na vnější vstupní podnět dávají výstupní signál zatížený také vnitřními a vnějšími parazitními vlivy. Vnitřní chyby senzorů jsou dány vlastními systematickými a náhodnými procesy senzoru a jeho vnitřním rušením. Vnějším vlivem je jednak vazba na měřený proces, jednak vazba výstupu na další obvody vyhodnocování, působení interferenčního a elektromagnetického rušení.
Vstupní vazba senzoru na proces je dána kvalitou přenosu měřeného podnětu na senzor a případnou zpětnou vazbou působení senzoru na proces. Např. senzor pro měření teploty v prostředí musí mít zajištěn co nejdokonalejší přestup tepla do senzoru. Zpětná vazba senzoru na proces může být tepelné působení senzoru na snímanou plochu při snímání elektromagnetického záření.

Vliv výstupních vazeb lze nejčastěji pozorovat u parametru zátěžného odporu výstupní jednotky anebo u rušení na spojovacím vedení. Zpětným působení vyhodnocovací jednotky může být např. ohřev teplotního odporového senzoru od měřicího proudu [Hruška, 2010].

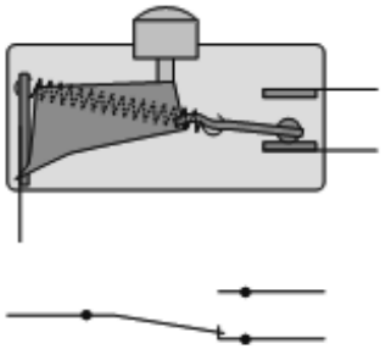
V praxi jsou používány metody, které umožní zmenšit vznik chyb snímání senzorů. Mezi nejznámější metody můžeme zařadit:

- kompenzační/diferenční/zpětnovazební zapojení senzorů
- linearizační zapojení
- automatickou kalibraci
- korekce dynamické chyby.

2.1.3 Odporové snímače polohy

Odporové senzory patří mezi dotykové absolutní senzory. Jako čidlo je používán potenciometr – regulovatelný odporový napěťový dělič. Vlastnosti snímače jsou ovlivňovány hodnotou *TKR*[1], životností, rozlišovací schopností, třídou přesnosti a linearitou. Měřená neelektrická veličina je spojitě převedena na změnu odporu a ta je vyhodnocena. Vstupní podnět způsobuje u odporových senzorů změnu elektrického odporu. Takovým vnějším podnětem je např. poloha předmětu [Beneš, 2014].

Kleština tyto senzory využívá pro určení fyzických limitních poloh na obou stranách válečkové trati a rovněž jako referenční koncový senzor, na který kleština referuje. V technické praxi se tomuto senzoru říká koncový snímač, protože jasně vytyčuje koncovou polohu při polohování pohonů, kdy po sepnutí přepínače se přestaví výstup podle zapojení, jak je ukázáno na Obr 6. Jedná se tedy o snímače se skokovou změnou odporu – převádějí změnu polohy sledovaného objektu na skokovou změnu odporu způsobenou přepínáním kontaktů. Výstupní signál je logického typu (sepnuto - vypnuto).



Obr. 6 Koncový snímač s mechanicky ovládaným mžikovým přepínačem [zdroj: Beneš, 2014, strana 76]

[1] Teplotní koeficient odporu

[2] Oscilační obvod, jehož parametry jsou indukčnost L a kapacita C

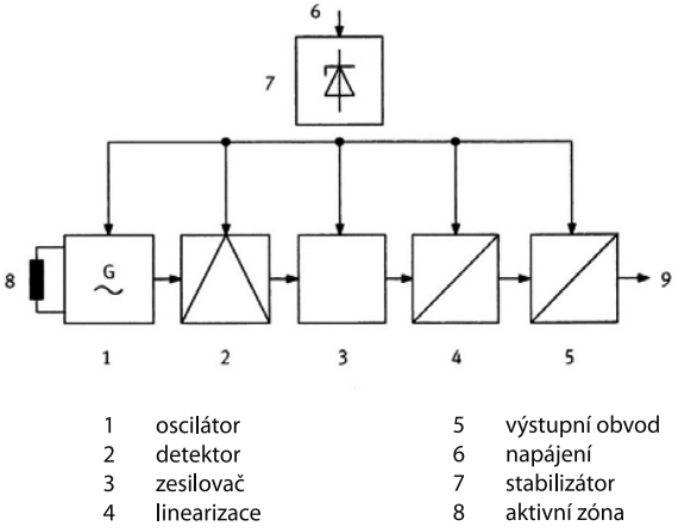
2.1.4 Indukčnostní senzory

Tyto senzory jsou pasivní. Měřená veličina je převáděna na změnu indukčnosti L přes změnu magnetického toku nebo vzájemné indukčnosti M. Indukčnost je připojena do měřicího obvodu se střídavým napájením, nejběžněji můstkového nebo rezonančního.

2.1.4.1 Senzory s malou vzduchovou mezerou

Někdy též nazývány jako bezdotykové senzory polohy jsou velmi rozšířené pro měření posuvů, poloh a dalších veličin. Jedná se o pasivní, binární nebo analogové senzory, reagující pouze na kov. Funkční schéma je zobrazeno na Obr. 7.

Princip měření spočívá ve změně vzduchové mezery mezi čidlem a snímaným objektem. Základem senzoru je trvale pracující oscilátor, nejčastěji *LC*[2], jehož kmitočet je běžně 0,1 až 1 MHz Cívkou senzoru prochází střídavý proud a kolem cívky, čidla se vytváří magnetické pole. Pokud se v tomto poli nachází elektricky vodivý kovový materiál, tak se do něho podle Faradayova zákona naindukují vířivé proudy. Podle Lenzova pravidla je pole generované vířivými proudy protiběžné v porovnání s polem generovaným cívkou. Tím se zmenšuje amplituda oscilací. Pokud dále přibližujeme vodivý předmět, sníží se amplituda natolik, že její snížení vyhodnotí klopný obvod a změní svůj stav. Tím také změní svůj stav výstupní obvod, který podle zapojení sepne nebo rozepne spínač [Beneš, 2014].



Obr. 7 Funkční schéma indukčnostního senzoru [zdroj: Beneš, 2014, strana 69]

Kleština používá tyto senzory k detekci přítomnosti materiálu v upínacích čelistech kleštiny při hledání konce materiálu. Logická 1 na výstupu senzoru znamená přítomnost materiálu. Další využití indukčnostního senzoru je k detekci maximálního otevření upínacích čelistí kleštiny.

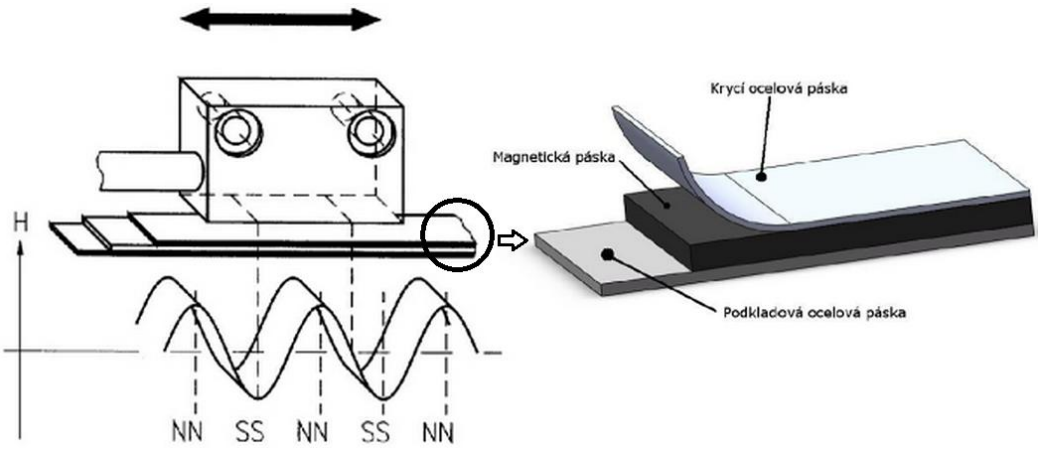
[2] Oscilační obvod, jehož parametry jsou indukčnost L a kapacita C

2.1.5 Lineárně magnetické snímače

Pro velice přesné měření lineárních posuvů a pohybu se velmi hodí lineární magnetické snímače. U těchto senzorů je *vlnovod*[3] vyroben ze speciální slitiny niklu a oceli s vnějším průměrem 0,7 mm a vnitřním 0,5 mm. Měděný vodič je vedený vnitřkem po celé délce pásky. Začátek měření je inicializován krátkým proudovým impulzem. Tento proud vytváří magnetické pole, které se otáčí okolo vlnovodu. Permanentní magnet, jenž je uložen ve snímací hlavě v bodě měření, je použit jako ukazatel polohy, jehož magnetické siločáry jsou kolmé k elektromagnetickému poli (viz Obr. 8).

V místě vlnovodu, kde se obě pole protnou, se vytvoří vlivem magnetostrikčního jevu velmi malá elastická deformace, která se šíří vlnovodem oběma směry ve formě mechanické vlny. Rychlost šíření této vlny na vlnovodu je 2 830 m/s a je prakticky nezávislá na vlivech okolního prostředí. Část vlny, která dosáhne ke vzdálenému konci vlnovodu, je zatlumena, kdežto část, jež přijde do signálového převodníku, je změněna na elektrický signál obrácením magnetostrikčního efektu [Beneš, 2014].

Naměřený čas pak dovoluje určit vzdálenost s extrémně vysokou přesností (rozlišení měření se pohybuje v rozsahu 10 až 250 mikronů, tedy 0,01 až 0,25 mm dle zvoleného typu.). Snímače běžně stíhají měřit pohyb až 1 m/s při pracovní teplotě okolí 0 až 85 °C. Jejich elektrický výstup je v podobě 5V *TTL*[4] signálu.



Obr. 8 Princip funkce lineárního magnetického snímače a struktura magnetické pásky [zdroj: http://automatizace.hw.cz/files/images/image/smallImage1201.jpg]

Magnetické snímače slouží obecně k bezdotykové a bez opotřebení probíhající detekci poloh v řídicí technice. Jsou používány všude tam, kde již indukčnostní senzory z hlediska vzdálenosti nestačí, protože proti nim nabízí podstatně delší spínací vzdálenosti při stejných nebo i menších rozměrech snímače. Další jejich výhodou je použití jen magnetismu. Magnetická pole totiž prochází všemi

[3] Vedení pro přenos elektromagnetických vln

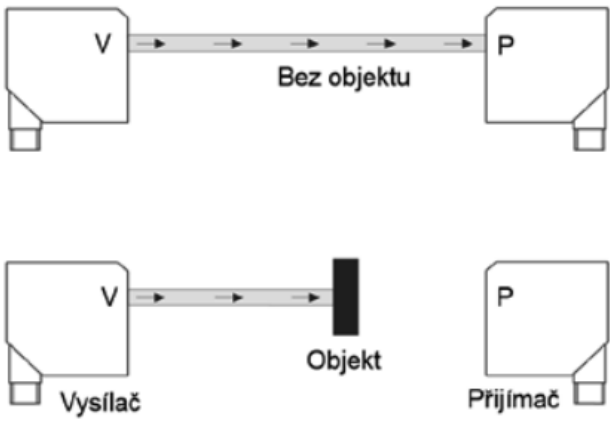
[4] Standard používaný pro implementaci digitálních integrovaných obvodů

nemagnetickými materiály, a tak mohou tyto senzory rozpoznávat magnety, které jsou umístěny např. za stěnami z barevných kovů, ušlechtilé ocele, hliníku, umělých hmot nebo dřeva. V případě kleštiny je tento senzor využit k absolutnímu polohování pojízdného mechanismu po ozubeném hřebenu. Díky tomuto snímači kleština ví, na jaké pozici od nulového bodu se aktuálně nachází.

2.1.6 Optoelektronické senzory

Optoelektronické senzory, v technické praxi častěji používanější název optické závory, jsou jedním z nejčastěji používaných prostředků ke snímání předmětů v automatizaci výrobních procesů. V zásadě jsou používány 2 typy závor: jednocestná optická závora a reflexní optická závora. Kleština využívá jednocestnou variantu.

Jednocestné optické závory, jak je ukázáno na Obr. 9, se skládají z vysílací a přijímací jednotky, které jsou umístěny na protilehlých stranách snímací cesty (v jedné rovině). Jestliže je nějakým předmětem přerušena přímá cesta světla mezi vysílačem a přijímačem, změní se elektrické vlastnosti fotodetektoru a tato změna se projeví tak, že je ve výstupním stupni signalizována změna stavu. Snímání není ovlivněno vlastnostmi povrchu snímaného objektu. Dosah snímání jednocestných optických závor je až 60 m. Jednocestné závory využívají červené, infračervené a laserové světlo. Laserové jednocestné závory jsou používány i ke snímání velmi malých předmětů, např. na vzdálenost 30 m rozliší předmět o průměru 2 mm, zatímco na vzdálenost 3 m dokážou ve snímaném prostoru identifikovat předmět o velikosti 0,5 mm [ŠIKUT, 2010].



Obr. 9 Jednocestná světelná závora [zdroj: Beneš, 2014, strana 73]

V našem případě detekuje jednocestná světelná závora začátek naváženého materiálu, a je fyzicky přítomna na konci vstupní válečkové tratě před horizontálním hydraulickým svěrákem u vrtacího centra.

2.1.7 Závěr podkapitoly

Rozvoj senzorů v následujícím období bude pokračovat rostoucím trendem. Můžeme předpovídat, že v nejbližším období se budou rozvíjet hlavně *MEMS*[5] senzory a biosenzory. Ve větším rozsahu se bude uplatňovat inteligence senzorů s parametrizací vyhodnocování, komunikace u obvodů se bude rozšiřovat do oblasti bezdrátového propojení ze senzorů. Podstatně větším tempem se rovněž bude rozšiřovat výzkum nových materiálů pro senzory.

2.2 Akční členy

Akčními členy jsou myšleny veškeré prvky určené k využití zpracované informace. Akční členy nastavují velikost akční veličiny, tj. realizují vstup do regulované soustavy. Jsou to tedy prvky nacházející se na konci řetězce zpracování informace. Mezi nejčastější představitele akčních členů patří hlavně pohony a na ně navazující regulační orgány.

Pohony jsou zařízení, která převádějí signály z členů pro zpracování informace na výchylku konající požadovanou práci s požadovaným výkonem. Regulační orgány jsou zařízení pro ovládání toku hmoty nebo energie systémem. Ne vždy je možno rozdělit akční člen na pohon a regulační orgán.

Podle energie, která je využitá k vykonání práce pohonů, rozlišujeme 3 druhy pohonů: elektrické, pneumatické a hydraulické.

2.2.1 Pneumatické pohony

Pneumatické pohony mění energii stlačeného vzduchu v mechanickou energii. Vyznačují se jednoduchým a robustním provedením, čistotou provozu, vysokou provozní spolehlivostí, velkými přestavnými silami (řádově až 10^4 N) a poměrně krátkými přestavnými dobami. Jsou vhodné do provozů s agresivním prostředím a nebezpečím požáru či exploze a to vše při nízkých vstupních nákladech. Pneumatické pohony dělíme podle prvku převádějícího tlak na sílu nebo výchylku na:

- pohony s membránou,
- pohony s pístem,
- pohony s vlnovcem,
- pohony speciální

Podle způsobu generování pohybu na:

- jednočinné,
- dvojčinné

[5] integrace mechanických elementů, senzorů a vyhodnocovací elektroniky na jeden křemíkový substrát prostřednictvím různých výrobních technologií

Podle dráhy výstupního prvku na:

- posuvné,
- kyvné,
- rotační

Pneumotor je zařízení, ve kterém se energie stlačeného vzduchu přeměňuje na jiný druh energie, zpravidla mechanickou [Odstrčilíková,2003].

Samotný pneumotor se skládá z válce, v němž se pohybuje píst. Tlakové médium, stlačený vzduch dodává pneumotoru kompresor, jenž je doplněn o jednotku úpravy vzduchu obsahující filtr zbavující vzduch nečistot a většinou i maznici, která dodává do vzduchu malé množství oleje sloužícího k mazání jednotlivých pohyblivých členů systému. Ovládající šoupě rozděluje tlak pracovního vzduchu na jednu nebo druhou stranu pístu, tím se píst pohybuje a čelisti kleštiny otevírají/upínají.

2.2.2 Elektrické pohony

Elektromotory mění elektrický proud v mechanickou energii. Každý elektromotor se skládá z nepohyblivé části zvané stator a z pohyblivé části zvané rotor neboli kotva, která se otáčí uvnitř statoru. Průchodem elektrického proudu vinutím statoru a vinutím rotoru se vytváří dvě magnetická pole, která na sebe vzájemně působí přitažlivými a odpudivými silami tak, že se rotor otáčí. Ztrátou elektrického proudu v odporech vinutí vzniká teplo, které se prakticky jeví jako zahřívání elektromotoru. Motory se proto musí chladit. U otevřených a polo-otevřených motorů chladí ventilátor, u uzavřených, povrchově chlazených, chladící žebra na statoru. Účinnost motorů se pohybuje mezi 75 – 90 %. Energie se ztrácí na Jouleovo teplo, Foucaultovými proudy, hysterezí, jiskřením apod.

Podle napájecího napětí dělíme elektrické motory na stejnosměrné a střídavé. Podle vzájemného působení magnetických polí na synchronní a asynchronní.

Z výhod plynoucích použitím elektrického pohonu oproti neelektrickému můžeme zmínit např. rychlou (téměř okamžitou) pohotovost nasazení a možnost krátkodobého přetížení. Dále pak malá hmotnost, snadná vyměnitelnost a téměř bezúdržbový provoz. Naopak hlavními nevýhodami elektrických pohonů jsou neustálá závislost na přívodu el. energie (výjimkou tvoří jen pohony s napájením z akumulátorů), vysoké jmenovité otáčky (tuto nevýhodu lze za použití mechanických převodů odstranit). [Vávra, 2004].

Otáčky točivého pole jsou určeny síťovým kmitočtem a počtem pólů trojfázového vinutí (počet pólových párů). Otáčky jsou také označovány jako frekvence otáčení. V elektrických strojích dochází ke ztrátám stejně jako v transformátorech, a to ke ztrátám v železe, ztrátám ve vinutí daným činným odporem drátu a dále také ke ztrátám třením – v ložiskách a na kartáčích.

Měřítkem celkových ztrát je účinnost motoru **η** (1.1.), která je dána poměrem odváděného výkonu $\boldsymbol{P_2}$ a odebíraného výkonu $\boldsymbol{P_1}$, formálně se jedná o zápis:

$$\eta = \frac{P_2}{P_1} \tag{1.1.}$$

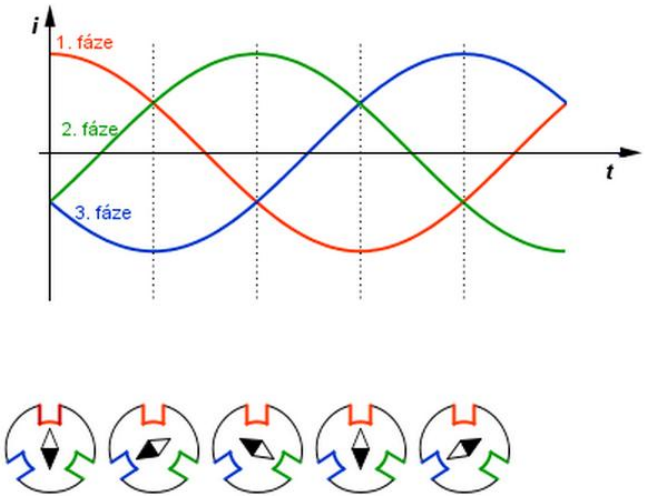
Výkon odevzdaný elektromotorem P_2 lze vypočítat z točivého momentu a otáček. Vstupní výkon P_1 lze změřit jako elektrický výkon odebíraný motorem ze sítě. Točivý moment je u elektromotoru výsledný účinek působení magnetického pole statoru a proudu procházejícího otáčejícím se rotorem. Při jmenovitém výkonu má motor jmenovitý moment při jmenovitých otáčkách.

2.2.2.1 Střídavé elektromotory

Podle počtu fází je dělíme na jednofázové a třífázové, podle konstrukce a způsobu provozu na synchronní, asynchronní, komutátorové, se stíněným polem a krokové.

U synchronního motoru se rotor otáčí současně (synchronně) s rotací magnetického pole, naopak u asynchronního motoru se rotor za rotací magnetického pole zpožďuje (má tzv. skluz).

Jednofázové motory se napájejí jednoduchým střídavým napětím (ze zásuvky), zatímco třífázové motory využívají k vyvolání pohybu rotoru základní vlastnost 3 fázové soustavy, a to že, stator tvoří tři elektromagnety, které jsou navzájem posunuty o 120°, jak je znázorněno na Obr. 10. Každým vinutím elektromagnetu protéká proud vždy jen jedné fáze třífázového proudu.



Obr. 10 Vzájemný posun 3 napětí na 3 fázích o 1/3 otáčky, tedy 120°
[zdroj: http://skola.hellebrand.cz/text0910/ele/motory_str.pdf]

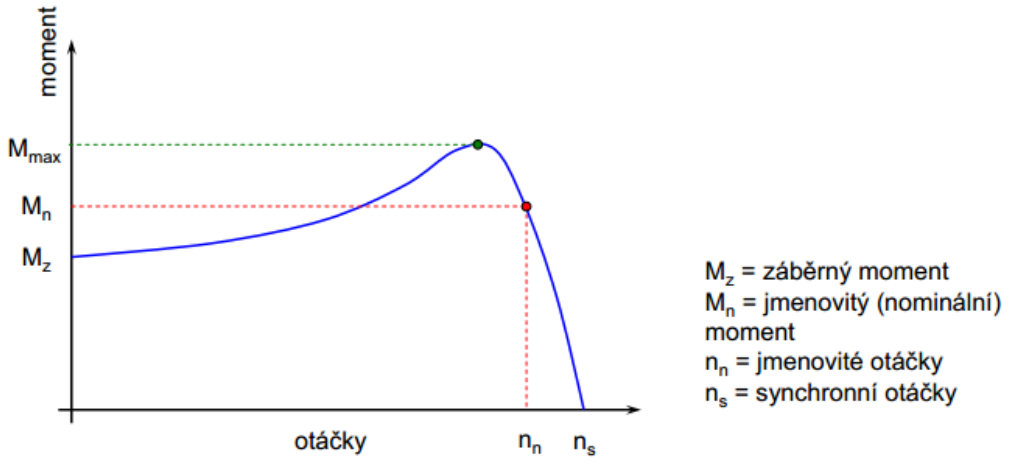
2.2.2.2 Asynchronní třífázový motor

Třífázový asynchronní motor je v průmyslu nejpoužívanější elektrický motor. Je levný, dobře regulovatelný a má jednoduchou konstrukci. Vyrábí se pro široké rozpětí výkonů (od desítek Wattů po MegaWatty). Přehození dvou fází způsobí

změnu smyslu otáčení točivého pole a tedy i hřídel se bude otáčet opačným směrem. Rotor motoru nemá žádné vinutí, ale pouze vodivě propojenou klec – motor nakrátko. Regulaci otáček můžeme provést stupňovitě, je-li motor vybaven přepínatelnými póly nebo změnou frekvence pomocí měniče frekvence.

V případě modulu kleštiny je motor využit pro samotný pohyb kleštiny po válečkové trati a je řízen měničem frekvence. Hřídel motoru je osazena ozubeným pastorkem a hnací síla se tak převádí otáčením pastorku po ozubeném hřebeni umístěném po celé délce válečkové tratě. Řízení změnou frekvence je moderní způsob řízení motorů a dovoluje nám velký rozsah regulace otáček.

Průběh momentu v závislosti na otáčkách (viz Obr. 11) ukazuje nárůst až do hodnoty $\boldsymbol{M_{max}}$,kdy začne pokles rychlosti změn indukčního toku ve smyčkách rotoru převažovat nad vlivy zvětšujícími moment. Při jmenovitých otáčkách působí jmenovitý moment $\boldsymbol{M_n}$ (jmenovité zatížení). V nezatíženém stavu dosahuje motor téměř synchronních otáček $\boldsymbol{n_s}$. V okolí jmenovitého momentu $\boldsymbol{M_n}$ jsou změny skluzu úměrné změnám zatížení $\Delta\mathbf{M}$, neboť charakteristika je zde téměř lineární. V určitém bodu již další navyšování momentu nemá významnějšího účinku a otáčky se zvětší jen nepatrně, za to nárůst vstupního odebíraného proud je značný.



Obr. 11 Momentová charakteristika 3 fázového asynchronního motoru [zdroj: http://skola.hellebrand.cz/text0910/ele/motory_str.pdf]

Pohony s třífázovými asynchronními motory musí být často brzděny, např. v případě rychlého zastavení. Při elektrickém brzdění asynchronních motorů se kinetická energie rotoru vrací zpět do napájecího zdroje tzv. rekuperačním brzděním nebo se přemění v odporech na tepelnou energii. Rychlé brzdění znamená, že je třeba se postarat velmi rychle o energii, která je v rotujícím zařízení uložena. Velká část energie se přemění zpět na elektrickou energii, která se opět ocitne ve frekvenčním měniči. Následkem je zvýšení napětí na kondenzátorech v mezistupni měniče. Pokud napětí překročí povolenou úroveň, měnič se odpojí a objeví se chybové hlášení, většinou "Přepětí na mezistupni".

Pro brzdění elektrického motoru lze použít několik metod:

- Generátorické brzdění – lze jej využít pro zastavení motoru pouze, pokud je možné měnit frekvenci otáčení magnetického pole frekvenčním měničem. Jedná se o nejhospodárnější způsob brždění, protože vyrobená energie je navrácena zpět do sítě.
- Brzdění protiproudem – změněním smyslu otáčení magnetického pole statoru se vytváří brzdný moment, působící proti směru otáčení rotoru. Vzniká obrovské přetížení motoru a nárůstu proudu, toto řešení není tedy moc vhodně. Navíc veškeré teplo vytvořené bržděním zůstává v motoru

2.3 Systém automatické regulace

Jedním ze současných trendů v oblasti moderních regulovaných pohonů je snižování jimi spotřebované elektrické energie a zvyšování jejich účinnosti. Téměř 60 % elektrické energie spotřebovávají elektrické motory pro pohon mechanických zařízení. Jsou-li pohony neregulované, značná část této energie je neefektivně zmařena. Výrobci strojních zařízení proto stále častěji instalují před elektrické motory pro pohon jejich zařízení měniče frekvence. Stejně tak provozovatelé starších zařízení se intenzivně snaží vhodně nahradit dosavadní způsoby regulace řešením s měniči frekvence. Hlavními přednostmi tohoto řešení jsou kromě regulace otáček také rozběhy motorů, zabudované ochranné funkce a snížení mechanického opotřebení navazujících zařízení. Měniče frekvence lze najít v pračkách, pohonech vrat, tepelných čerpadlech, v jeřábech, kompresorech, drtičích, papírenských strojích a v mnoha dalších úlohách v domácnostech i v průmyslu [Pavelková, 2010].

2.3.1 Měnič frekvence

Pro většinu lidí je pojem měnič frekvence zcela neznámým pojmem. Pro podniky vyrábějící elektrické motory a mnoho lidí pracujících v průmyslu, kteří s těmito motory pracují, to je opačně. Bez nějakého přehánění lze konstatovat, že měniče způsobily revoluci v regulaci otáček elektromotorů.

Již v historii proběhlo nepřeberné množství pokusů jak regulovat asynchronní motory. Před druhou světovou válkou byly zaznamenány první pokusy jak pomocí frekvence hospodárně regulovat otáčky asynchronního motoru, avšak dostupná technika, která by zajišťovala bezkontaktní spínání byla ještě v plenkách. Koncem šedesátých let minulého století se objevili první průmyslově vyráběné měniče frekvence. Od té doby se výrazně vyvinula výkonová elektronika, výrazně vzrostla i spolehlivost a nastavení.

Měnič frekvence v současném pojetí je elektronický přístroj, který umožňuje měnit frekvenci sítě na požadovanou frekvenci. Měniče frekvence jsou určeny pro nejrůznější použití, avšak v běžném technickém slovníku se jimi rozumějí měniče pro asynchronní motory. Tímto spojením získávají tyto střídavé točivé stroje schopnost hospodárně regulovat otáčky v širokém rozsahu.

Měniče frekvence se skládají z výkonové části zajišťující přeměnu parametrů napájecí sítě a z řídicí elektroniky, která ovládá výkonovou část a umožňuje okolní komunikaci. Řídicí elektronika moderních měničů postoupila již tak daleko, že nám dovoluje vykonávat mnoho úloh naráz, které by jinak musely být zahrnuty v nadřazeném řídicím systému. Ovládání frekvenčního měniče je možné z panelu, kde je obvykle k dispozici několikařádkový alfanumerický displej a několik málo tlačítek. V dnešní době se však ovládání z panelu používá velmi zřídka a spíše se přechází na ovládání z nadřazeného řídicího systému po průmyslové sběrnici.

Z hlediska vnitřního řízení měniče frekvence se vyskytuje několik systémů, o kterých je třeba se zmínit. Nejjednodušší je skalární řízení, které v podstatě vytváří síť proměnného napětí a kmitočtu nezávisle na motoru, je založeno na řízení *U/f* [6]. Je proto dynamicky nejpomalejší, avšak pro jednoduché aplikace plně vyhovuje. Naopak velmi přesné a dynamické řízení je vektorové, které však vyžaduje otáčkovou zpětnou vazbu (tachogenerátor). Jak pro skalární tak pro vektorové řízení je typický prvek modulátoru, který řídí spínání prvků střídače s pravidelnou spínací frekvencí. Tyto technologie nabízejí dnes všichni výrobci měničů [ABB, 2011].

V současné době je asi nejdokonalejší řízení tzv. přímé řízení momentu *DTC*. Jádrem systému jsou hysterézní regulátory momentu a magnetického toku, které využívají optimalizovanou spínací logiku, odpadá tak prvek modulátoru. Velmi důležitou částí řízení je přesný model motoru. V něm se vypočítává skutečný moment, statorový magnetický tok a otáčky hřídele z proudu měřeného ve dvou fázích motoru a ze stejnosměrného napětí v meziobvodu. Tyto výpočty jsou během jedné sekundy uskutečněny 40 000 krát, takže *DTC* přesně ví, jak se chová hřídel motoru. Hlavními parametry modelu motoru jsou indukčnosti a odpor statoru. Bere se v úvahu rovněž vliv magnetické indukce na velikost indukčností [Automa, 2007]. Tuto technologii dnes nenabízí každý výrobce. Exkluzivním vývojem a neustálým zdokonalováním této metody se již od roku 1988 zabývá firma ABB.

2.3.1.1 Činnost měniče frekvence

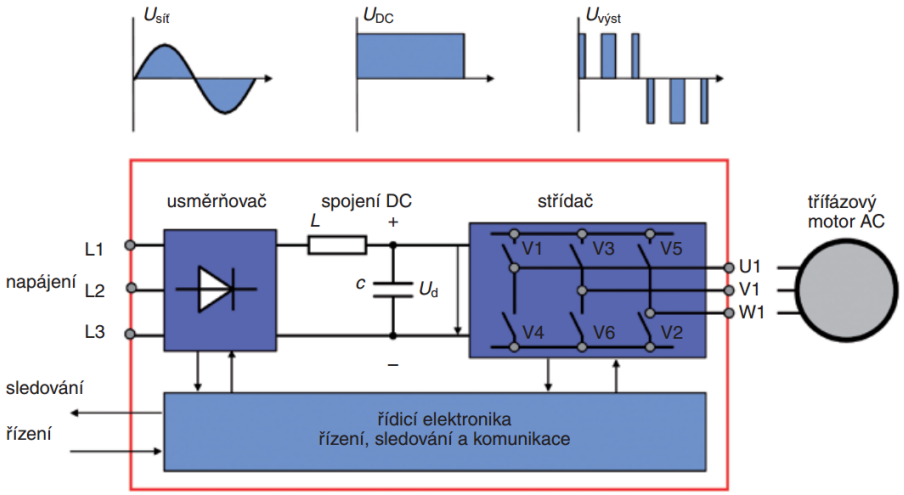
Fungování měniče frekvence lze nejlépe pochopit z blokového schématu na Obr. 12. Jak můžeme vidět, napětí ze sítě projde nejprve odrušovacím filtrem, kde se usměrní. Nejběžnější usměrňovač využívaný v technické praxi je diodový. Ve stejnosměrném meziobvodu se napětí filtruje pomocí tlumivky a kondenzátorů a toto stejnosměrné napětí se přivede na vstup střídače, který opět vytvoří střídavou třífázovou síť, nyní však s proměnným napětím a frekvencí. Na výstup měniče frekvence je připojen asynchronní motor, jehož otáčky jsou přímo úměrné frekvenci. Měnič sám o sobě je v dnešní době osazováním téměř výhradně spínacími tranzistory *IGBT*[7].

Regulační obvody zajišťují jednak vlastní činnost měniče, optimalizují práci motoru a mají rovněž dohlížecí funkci. V případě výrazné odchylky některých

[6] Poměr napětí a frekvence

[7] Bipolární tranzistor s izolovaným hradlem zkonstruován pro velký rozsah spínaných výkonů

parametrů (např. napětí, proudu, teploty, aj.) vydají varování a v případě dalšího nebezpečného vývoje oznámí poruchu a měnič odstaví. Celé řízení je v současné době digitální a tedy vysoce spolehlivé. Aplikační programy je možno obvykle zvolit (přepnout) z několika možností, což u vyspělých výrobků umožňuje zvolit např. ovládání z více míst, využít PID regulátor pro regulaci, použít speciální software podle požadavku zákazníka, apod [Pavelková, 2010].



Obr. 12 Princip měničů frekvence
[zdroj: http://propohony.cz/images/stories/menice_regulatory/abb/abb_3_2.jpg]

2.3.1.2 Elektromagnetická kompatibilita (EMC)

EMC zařízení zjednodušeně znamená, že zařízení smí generovat jen tak vysoké úrovně rušivých signálů, aby neovlivňovalo jiné zařízení, a současně musí mít takovou imunitu, aby nebylo těmito zařízeními samo rušeno. Toto je velmi přísně stanoveno Evropskými normami (EN), platnými jako ČSN EN v nezměněné formě i u nás. Normy tedy zaručují, že přístroj ani není rušen a ani neruší síť, ke které je připojen.

2.3.2 PID regulace

Již více než 60 let se v průmyslu využívají regulátory typu PID. Ačkoliv se regulátor nachází uvnitř měniče frekvence, je tedy jeho součástí, jeho význam, popis funkce a přínos pro řízení pohonů jsou tak obsáhle, že mu bude věnována samostatná kapitola.

PID v řídicí technice znamená proporcionální, integrační, derivační složku univerzálního regulátoru (tzv. PID regulátor). Od původně pneumatických se přes analogové přešlo na současné číslicové, ale jejich algoritmus řízení (proporcionálně–integračně-derivační) zůstává v podstatě stejný. Regulátor PID dnes představuje standardní a osvědčené řešení pro převážnou většinu průmyslových regulací. Navzdory této skutečnosti nelze říci, že existuje nějaká standardní a všeobecně přijatá metoda pro návrh PID regulátoru na základě

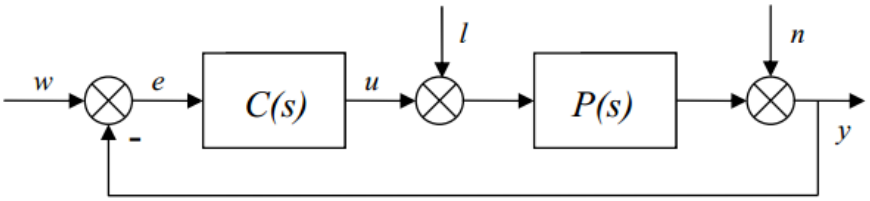
známého modelu řízené soustavy [Čech, Schlegel, 2004]. Regulátory se konstruují proto, aby řízení procesu nevyžadovalo nepřetržitou pozornost a ruční zásahy operátora. Běžnými příklady regulátoru jsou např. bytový termostat nebo tempomat v automobilu.

Uvažujme regulační smyčku s PID regulátorem $C_{(s)}$ a řízenou soustavou se stabilním přenosem $P_{(s)}$. Regulátor automaticky mění akční veličinu ***(u)*** tak, aby regulovaná veličina ***(y)*** měla žádanou hodnotu ***(w)***, zatímco ***(l)*** a ***(n)*** reprezentují poruchy působící na řízenou soustavu. Regulátor a jím regulovaná soustava tvoří regulační obvod (smyčku): na vstup regulátoru je přivedena spolu s požadovanou hodnotou ***(w)*** i skutečná hodnota regulované veličiny ***(y)*** a výstup regulátoru ***(u)*** působí, po případné transformaci, na vstup do soustavy. V porovnávacím členu je požadovaná hodnota ***(w)*** porovnána se skutečnou hodnotou ***(y)***, tím vzniká regulační odchylka ***(e)***.

Regulační odchylka v čase $e_{(t)}$ je tedy definována jako rozdíl mezi požadovanou a skutečnou hodnotou regulované veličiny (1.2.),

$$e_{(t)} = w_{(t)} - y_{(t)} \tag{1.2.}$$

která je přivedena na vstup ústředního členu regulátoru, zde je zesílena, požadovaným způsobem zpracována a výsledný regulovaný signál uvede v činnost akční člen. Ten provede prostřednictvím ***(u)*** zásah do regulované soustavy. Tento zásah musí být takový, aby se v následujících okamžicích vyrovnala regulovaná veličina ***(y)*** na požadovanou hodnotu, potom regulační odchylka ***(e)*** zanikne.

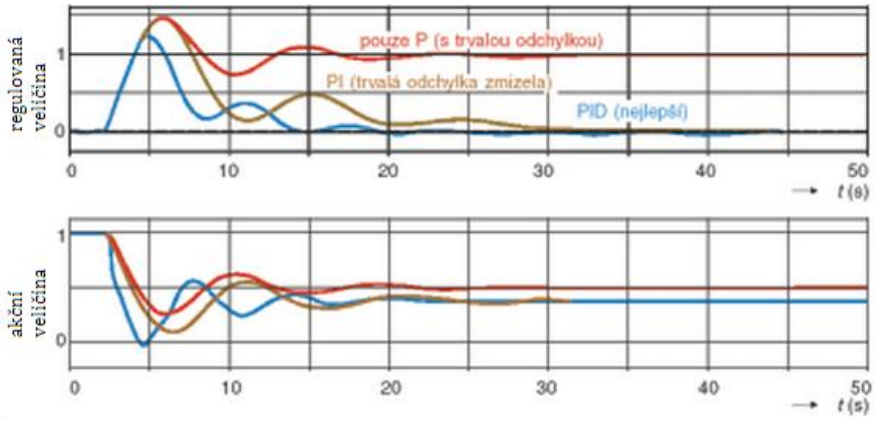


Obr. 13 Blokové schéma regulátoru

PID regulátoru jsou vlastní tři způsoby reakce na vznik regulační odchylky. Hovoří se o proporcionálním, integračním a derivačním chování regulátoru. Podíl každé z těchto tří složek na výsledném chování PID regulátoru lze nastavit prostřednictvím stavitelných konstant – parametrů regulátoru, ve volitelném poměru (správná volba je úkolem projektanta regulačního obvodu). Vžitá označení parametrů PID regulátoru jsou [Petrovas, Lisaukus, Rinkeviciene, 2011]:

- zesílení K – řídí proporcionální složku P,
- integrační časová konstanta T_I – řídí integrační složku I,
- derivační konstantu T_D – řídí derivační složku D

Složky P, I a D se skládají (sečítají) ve výslednou akční veličinu (akční zásah) a společně podmiňují průběh regulačního pochodu (Obr. 14). Ten, protože jde o uzavřený regulační obvod, závisí také na vlastnostech regulované soustavy.



Obr. 14 Průběh regulačního pochodu jednotlivých složek regulátoru
[zdroj: http://www.odbornecasopisy.cz/imagesold/a0303771.gif]

Většina regulovaných soustav, se kterými se lze setkat v technické praxi, má charakter dynamického systému prvního nebo druhého řádu, nekmitavého, s případným dopravním zpožděním. Pro ně ve spojení s PID regulátorem platí následující úvahy[Automa, 2003].

2.3.2.1 Proporcionální chování

Využívá se zde vztahu, kdy akční zásah regulátoru je úměrný regulační odchylce. Proto má vztah následující, jednoduchý tvar:

$$u_{(t)} = Ke_{(t)} \tag{1.3.}$$

Použití samotného proporcionálního regulátoru vede ke vzniku trvalé regulační odchylky. Zvětšováním zesílení K lze trvalou regulační odchylku zmenšit. Vzniká však nebezpečí, že dojde k tzv. nestabilitě regulačního obvodu, tj. stavu, kdy regulovaná veličina neomezeně narůstá až k dorazu nebo poškození zařízení. K odstranění trvalé regulační odchylky se do činnosti regulátorů obvykle přidává integrační složka chování.

2.3.2.2 Integrační chování

Při integračním chování je akční zásah úměrný době, po kterou existuje regulační odchylka, tedy:

$$u_{(t)} = u_{(0)} + \frac{1}{T_I}\int_0^t e_{(t)}\, d_{(t)} \tag{1.4.}$$

Jak je patrné při pohledu na regulační pochod s regulátorem se zapojenou proporcionální i integrační složkou (PI regulátor), trvalá regulační odchylka zmizela (viz Obr. 14). Je eliminována integračním chováním regulátoru, při kterém

regulátor neustále mění akční veličinu, dokud se mu nepodaří dosáhnout nulové regulační odchylky.

Zvětšováním podílu integrační složky (zmenšováním T_I) kmitavost regulačního pochodu obecně roste. Do jisté míry ji lze zmírnit přidáním derivační složky.

2.3.2.3 Derivační chování

Při derivačním chování se výstup z regulátoru vytváří jako úměrný rychlosti změny regulační odchylky:

$$u_{(t)} = T_D \frac{de_{(t)}}{d_{(t)}} \tag{1.5.}$$

Derivační chování může v předstihu kompenzovat změny regulované veličiny, a proto se ho využívá k tlumení zákmitů regulačního pochodu. Princip je v tom, že jakmile se po změně žádané nebo skutečné (v důsledku poruch) hodnoty regulované veličiny začne regulovaná veličina znovu blížit své (nové) žádané hodnotě, způsobí derivační složka chování regulátoru preventivně změnu jeho zesílení „špatným“ směrem (tj. „od“ žádané hodnoty). Derivační složka chování se často používá také k zamezení překmitu průběhu regulačního pochodu.

2.4 Programovatelné logické automaty

Programovatelný automat, v technické praxi označován jednoduše jako PLC je číslicově pracující elektronický systém konstruovaný pro použití v průmyslovém prostředí, využívající programovatelnou paměť pro interní ukládání uživatelsky orientovaných instrukcí pro provádění specifických funkcí (logických, sekvenčních, časovacích, čítacích, komunikačních a organizačních) za účelem řízení strojů či procesů, a to prostřednictvím digitálních nebo analogových vstupů a výstupů [Švarc, 2002]. Co se týče provedení, dělíme PLC do několika skupin:

- **Kompaktní programovatelné automaty**
  Jasně daná fixní konfigurace vstupů a výstupů. Napájecí zdroj, CPU, podsystém I/O, speciální moduly (komunikační rozhraní) jsou provedeny v jednom kompaktním celku bez možnosti rozšíření (jedině připojením dalšího rozšiřujícího kompaktního modulu). Kompaktní provedení PLC představuje asi 20 % trhu v průmyslové praxi – využívají se v nenáročných aplikacích, a jednoduchých technologických procesech, rovněž najdou využití u jedno-účelových strojů.

  V posledních letech ovšem zažili kompaktní PLC „boom“, kdy se začaly využívat k řízení inteligentních domů, či jednotlivých periferií v domácnostech (ovládání garáže, zavlažování, zabezpečení venkovních prostor, apod.).

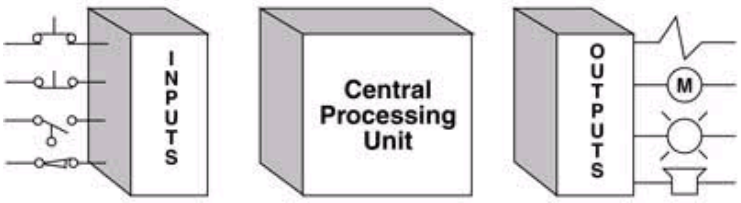
- **Modulární programovatelné automaty**
  Tyto automaty umožňují modulární výstavbu systému, a tím poskytují nesrovnatelně větší volnost ve volbě konfigurace. Do různých variant plochého zadního roštu se zasouvají jednotlivé moduly. Velký rozsah zdrojů napájení, CPU jednotek, I/O modulů, speciálních modulů a komunikačních modulů činí z těchto systémů jeden z nejpružnějších systémů automatizace na světě. Výkonový zdroj (buď na 230V AC nebo na 24V DC) má danou pozici, podobně jako CPU, rovněž přídavná paměť a komunikační procesory mají přesné místo v rámu. Ostatní karty mají polohu libovolnou. Rám je vybaven vnitřní paralelní systémovou sběrnicí pro signálové a elektrické propojení modulů s CPU a pamětí. Modulární provedení PLC je v dnešní době nejpoužívanější řešení napříč všemi odvětvími automatizace.

- **Soft PLC**
  Tvoří v současnosti novou kategorii řídicích systémů na bázi PC. Počítač realizuje všechny řídicí funkce PLC, avšak navíc obsahuje vývojové a odlaďovací prostředí, operátorské rozhraní. Bývá v něm zaitegrována rovněž vizualizace, archivace technologických dat, optimalizační, expertní a diagnostické funkce.

  Řízení jednotlivých strojů a nepříliš složitých technologií však není platformou pro aplikaci těchto systémů – zde ještě dlouho budou dominovat klasická PLC, soft PLC se uplatňují v případech náročného řízení rozsáhlých technologií, kde jsou velké nároky na výpočetní algoritmy a kde je požadováno rozsáhlé zpracování a archivace velkých objemů dat [Kozelský, 2010].

2.4.1 Historie a současnost

Rozvoj mikropočítačové techniky přinesl širší možnosti v podobě realizace složitějších automatizačních algoritmů pomocí programových prostředků. Poněvadž klasické mikropočítače neměly výstupní signály přizpůsobené potřebám technologie, začaly se vyvíjet specializované mikropočítače, u kterých byly technické prostředky a programové vybavení určeny především pro řízení technologických procesů. Tyto mikropočítače se začaly nazývat programovatelnými automaty, zkráceně PLC. V době svého vzniku (konec 60. let) si programovatelné automaty kladly za úkol nahradit efektivnějším způsobem reléovou a později i bezkontaktní logiku. Proto jejich architektura vycházela z toho, že budou zpracovávat binární informace. Jako HW jádro používali bitové procesory. Proto se na architekturu PLC kladly značné nároky v podobě bitově orientovaných CPU, sady jednoduchých instrukčních soborů pro zpracování logických rovnic, atd. Nicméně dnes je již toto řešení značně zastaralé a nevyhovující technologickým nárokům v průmyslové praxi.



Obr. 15 Zjednodušená struktura PLC [zdroj: http://www.amci.com/tutorials/images/inside-of-plc.gif]

Moderní PLC se svým provedením, souborem instrukcí a komunikačními schopnostmi výrazně odlišují od prvních přístrojů tohoto druhu vzniklých před více než půl stoletím. Pro současné programovatelné automaty je typické: velký výpočetní výkon, značný rozsah paměti, schopnost komunikace a dostupnost vyspělých algoritmů založených na nejnovějších teoretických poznatcích (neuronové sítě, fuzzy logika, PID regulace, apod.). Charakteristickým rysem moderních automatů jsou jejich komunikační schopnosti. Díky nim lze PLC zapojovat do sítí a vytvářet z nich distribuované systémy s různými topologiemi a způsoby komunikace.

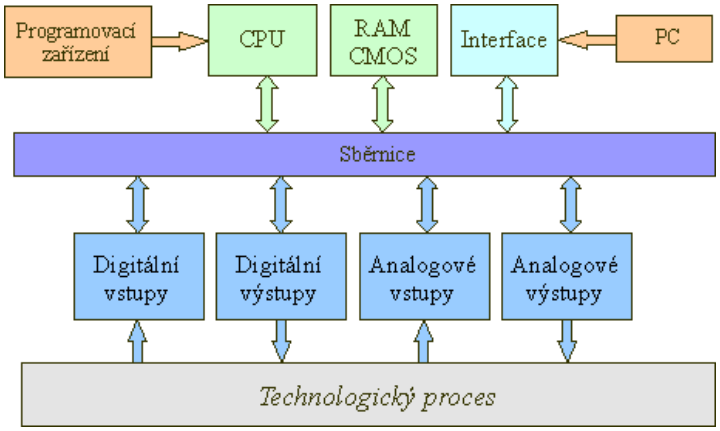
Programovatelné automaty mohou být podřízeny počítačovým systémům, současně ale mohou v síti počítačů komunikovat i na rovnocenné úrovni. Standardem na této úrovni se stávají PLC komunikující prostřednictvím průmyslového rozhraní Ethernet. PLC mohou být i součástí informačního a řídicího systému celé firmy. Takto pojatá automatizace je mnohdy označována jako úplná.

Možnost zapojovat PLC do sítí a vytvářet z nich distribuované systémy otevírá také cestu k realizaci paralelních inteligentních systémů. Takto lze efektivně řešit např. nejen rozsáhlé úlohy založené na aparátu konečných automatů a Petriho sítí, ale i neuronové sítě, fuzzy-neuronové algoritmy a mnoho dalších úloh náročných na numerické výpočty. Vyspělé PLC a jejich vývojové systémy mohou také komunikovat s výkonnými matematickými a simulačními programovými produkty (Matlab/Simulink), které lze s výhodou využít k řešení i těch nejnáročnějších grafických prezentačních úloh [Šmejkal a Urban, 2007].

Popsané současné řídicí systémy, stručně shrnují nejdůležitější vlastnosti současných programovatelných automatů nikoliv z pohledu jejich hardwaru nebo softwaru, ale především z pohledu použití k realizaci inteligentních algoritmů v průmyslové praxi, a to především v oborech řízení, regulace a technické diagnostiky.

2.4.2 Struktura a princip činnosti modulárních PLC

Je patrné, že schéma standardního modulárního PLC je velmi podobná architektuře mikropočítače (viz Obr. 16). Základem PLC jsou v principu tři funkční bloky: zpracování informace, vstupy/výstupy a paměť. Ostatní bloky jsou podpůrné a doplňkové. Hlavní stavební kámen tvoří vnitřní sběrnice, kolem které je modulárně vytvořen celý PLC.



Obr. 16 Blokové schéma modulárního programovatelného automatu

Hlavní prvky PLC mají následující funkci:

- **Vstupy/výstupy** – na binární vstupy se běžně připojují tlačítka, přepínače, koncové snímače a jiné snímače s dvouhodnotovým charakterem signálu. Binární výstupy jsou určeny k buzení cívek relé, stykačů, pneumatických a hydraulických převodníků, ale i ke stupňovitému řízení pohonů a měničů frekvence.

  Naopak analogové vstupy slouží k připojení snímačů tlaku, vlhkosti, teploty. Pomocí analogových výstupů lze ovládat spojité servopohony a jiné spojitě ovládané akční členy.

  Vstupy a výstupy bývají od čidel a ovládacích prvků v technologickém procesu galvanicky odděleny, aby se při průrazu síťového napětí při poškození čidla, pohonu nebo kabelu nezničil i automat. Vstupy jsou navíc osazeny tvarovacím filtrem, který odstraňuje nežádoucí efekty při nedokonalém spínání elektrických kontaktů snímačů.
- **CPU** – centrální jednotka zpracovává informace, tj. podle programu čte z operační paměti hodnoty vstupních proměnných, provádí s nimi početní operace a přestaví výstupní členy. Obsahuje mikroprocesor a řadič, zaměřený na rychlé provádění instrukcí.
- **Paměť** – zde jsou uloženy uživatelské registry, čítače a časovače, komunikační, časové a jiné systémové proměnné. Taktéž slouží pro uložení uživatelského programu. Na rozdíl od počítače si PLC při poruše řídicího systému musí zapamatovat poslední stav, od něhož po obnovení funkce pokračuje dál v činnosti, což klade nároky na velký objem paměti.

Důležitými funkčními prvky PLC jsou:

- **Časovače** – slouží k řízení doby trvání operací, jednak v řízeném procesu, jednak v řídicím programu (např. čekání na signál z procesu); časovače mají nastavitelný čas běhu, jsou spouštěny a zastavovány binárním signálem a po proběhnutí nastaveného času dávají na výstupu též binární signál, takže se velice dobře dají zařazovat do kombinační i sekvenční logiky,

- **Čítače** – slouží k počítání vstupních pulsů nebo k vysílání určeného počtu pulsů na výstup; počet pulsů je předvolitelný, čítače jsou ovládány binárními signály a jejich výstupy jsou rovněž binární signály, takže mohou být také snadno zařazovány do struktur logického řízení,
- **Sekvenční registry** (nebo se jim také říká posuvné registry) – jsou to posloupnosti bitů, ve kterých je každý bit samostatně adresovatelný, tj. můžeme se na něj v programu odkazovat. Vložení nové informace (logické 0 nebo 1) na vstup sekvenčního registru způsobí, že se tato informace umístí do 1. bitu a obsah všech ostatních bitů se posune o jednu pozici směrem ke konci registru (poslední údaj se ztrácí); tím je možno určitou informaci posunovat a po určeném počtu kroků ji zase vyjmout a zpracovat.

Činnost PLC je založena na cyklickém provádění řídicího programu. Jednotlivým úlohám (*task*) jsou přiřazeny priority. Doba cyklu PLC je zpravidla definována jako doba, kterou PLC potřebuje k načtení dat, vyslání dat na výstupy a zpracování 1k instrukcí (nikoli 1 instrukce, neboť program o 1 instrukci nedává smysl). Typická doba cyklu moderních PLC (k roku 2014) je 1000 – 10 000 μs, přičemž PLC s rychlými CPU a dobou cyklu v řádech desítek *μs* již dnes nejsou výjimkou. Např. firma *B&R*[8] přišla v tomto roce s novinkou v podobě snížení doby reakce v průmyslové automatizaci až na 1μs, což umožňuje vykonávání časově kritických subprocesů ve velice krátkém čase. Zde už ovšem pomalu narážíme na fyzikální zákony a možnosti např. přenosu informace ve vodiči od snímače až na vstupní kartu programovatelného automatu.

V CPU se všechny operace provádějí mezi registry. Je jich tam několik, ale z hlediska základní činnosti PLC jsou důležité programový čítač a bitakumulátor. Programový čítač slouží k uložení adresy té instrukce v paměti, která se bude provádět v příštím kroku, zajišťuje tedy čtení programu z paměti a jeho provádění instrukci za instrukcí. Pro provádění logických operací je důležitý registr zvaný bitakumulátor. Na začátku logické operace se do něj uloží první operand, druhý operand je v pracovním registru a výsledek operace je uložen zase do bitakumulátoru. Logická hodnota obsahu bitakumulátoru pak rozhoduje o tom, zda se provedou či neprovedou následující instrukce, které modifikují obsah datové paměti a tím i výstupů z automatu do procesu. Před zahájením cyklu se sejmou hodnoty všech vstupů a zapíší do příslušné části datové paměti. Obsah celé datové paměti je v průběhu cyklu modifikován a po ukončení cyklu se hodnoty výstupních proměnných přenesou z datové paměti na výstupy z automatu. Instrukce prováděné v programu se týkají logických proměnných, ale rovněž tak práce s číselnými proměnnými, pokud je automat vybaven možnostmi aritmetických operací [Kadlec, Kmínek, 2005].

[8] Rakouská společnost vyrábějící automatizační techniku, zejména pak PLC

2.4.3 Power Panel 500

Toto zařízení můžeme chápat jako řídicí PLC vybavené dotykovou obrazovkou. Jedná se o centrální provozní řídicí jednotku, která je vybavena integrovaným řízením (totožné s řízením u PLC), vizualizací a komunikačním systémem. To uživateli otevírá možnosti kompletního inteligentního řešení, které lze snadno zapojit do jakékoliv automatizační infrastruktury.

Ovládání je dotykové s možností přídavných funkčních kláves kolem panelu. Panely jsou k dispozici v několika výkonových řadách s různě velkými dotykovými displeji, u řezacího centra je použit panel s 10“ LCD displej, viz Obr. 17.

Spolupráce výkonného procesoru Intel Atom™ a 1GB RAM paměti poskytuje dostatečný výkon i pro náročnější úlohy. Prostřednictvím modulárního sběrnicového rozhraní lze k systému snadno připojit vzdálené vstupy/výstupy a pohony (jak je zobrazeno na Obr. 25). Panel je vybaven rozhraním *X2X*[9] a Ethernet Powerlink. Může však být rozšířen o linky RS-232/RS-485, CAN a Profibus-DP slave podle požadavků zákazníka. Dále na panelu najdeme dva porty USB pro rychlou výměnu dat. Aplikační program je uložen na paměťové kartě typu Compact Flash, kterou je možné také využívat pro úschovu uživatelských dat.

Obr. 17 Operátorský panel Power Panel 500
[zdroj: http://files.vogel.de/vogelonline/vogelonline/companyfiles/4645.pdf, str.15]

[9] Sběrnice využívaná pro vysokorychlostní komunikaci mezi I/O a řídicím systémem

3 Komunikace v průmyslové automatizaci

Jako každý počítač, tak i programovatelný automat musí být vybaven vnitřními a vnějšími komunikačními kanály pro přenos dat, adres a řídicích signálů. Čím větší je decentralizace celého automatizačního systému, tím větší musíme klást důraz právě na komunikační podsystém.

Průmyslové automatizované systémy mohou být velice komplexní a většinou jsou strukturované do několika hierarchických úrovní (viz Obr. 18), kde každá z těchto úrovní obsahuje patřičnou komunikační úroveň s různými požadavky na komunikační systém. Průmyslové komunikační systémy mohou být z hlediska funkcionality klasifikovány do následujících kategorií [Djiev, 2003]:

- Informační úroveň

  Představuje nejvyšší úroveň společnosti, nebo automatizovaného systému. Řídicí jednotka na úrovni společnosti shromažďuje veškeré potřebné informace a řídí celý automatizovaný systém. Přenášejí se rozlehlé bloky údajů a dat. Na informační úrovni se používají lokální sítě, které se propojují do rozsáhlejších průmyslových sítí.

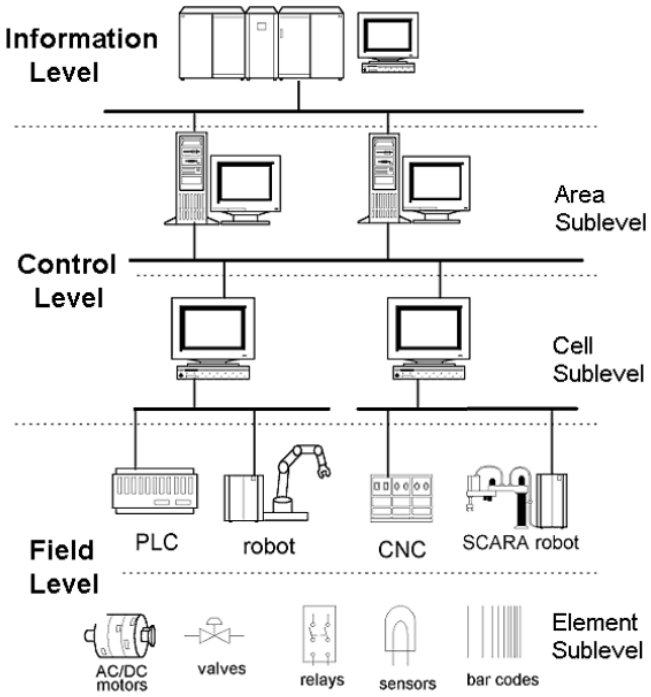
- Řídicí úroveň

  Na této úrovni se přenášejí především programy, parametry a údaje. V malých řídicích jednotkách může být nutné zavést programy během jednoho výrobního cyklu. Z toho vyplývají značné časové požadavky na přenos.

- Úroveň zařízení (sběrnice snímačů, akčních členů, provozních přístrojů)

  Nejnižší úrovní v automatizovaném průmyslovém systému, která obsahuje elementární zařízení jako snímače a akční členy. Úlohou zařízení na této úrovni je přenos informací mezi vyráběným produktem a technologickým procesem. Tyto informace můžou být binární či analogové. Měřené veličiny se musí přenášet v různých časových intervalech, v závislosti na technologickém procesu. Pro komunikaci na této úrovni se využívají paralelní a sériová rozhraní. Jako standard používaný v sériové komunikaci se používal protokol RS422 či RS485.

  Komunikační systémy na této úrovni se navzájem odlišují například ve velikosti přenášených bloků s údaji, čase odezvy a jiných parametrech. Systémy na této úrovni umožňují diagnostiku a konfiguraci zařízení z nadřízené úrovně. Jsou kladeny značné nároky na co nejúspornější formu přenášených údajů. Na úrovních zařízení se používají zejména sběrnice typu CAN, Profibus, Modbus, Powerlink, aj.



Obr. 18 Hierarchie průmyslového automatizačního systému [zdroj: DJIEV, 2003, strana 5]

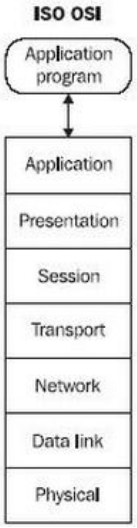
3.1 Referenční model ISO/OSI

Aby se předešlo problémům s používáním velkého množství nekompatibilních standardů, přistoupila mezinárodní organizace ISO k definování modelu pro komunikaci otevřených systému OSI. OSI sám o sobě není standard, ale spíše nám říká, jak identifikovat a oddělit odlišné části komunikačního procesu. Praktickým cílem OSI je optimální propojení sítě, v které bude možno přenášet údaje mezi různými místy.

3.1.1 Vrstvy modelu OSI

V modelu OSI je definováno 7 funkčních vrstev (jeho struktura je zobrazena na Obr. 19). Každá vrstva komunikuje přímo jen se sousední vrstvou, která se nachází nad/pod ní. Moduly lokalizované na stejné vrstvě, ale v odlišném uzlu sítě (tj. běží na jiném stroji) se nazývají *peer* (rovnocenné). Model definuje podmínky, při jejichž dodržení mohou různí účastníci přenosu spolehlivě komunikovat navzájem mezi sebou. Model je založen na vrstevnaté struktuře, kde daná vrstva poskytuje funkce (data) nadřízené vrstvě. V případě vysíláni zprávy volá vyšší vrstva službu vrstvy nejblíže nižší a naopak směrem nahoru poskytuje nižší vrstva svoje služby vrstvě vyšší. Každá vrstva má definovány dvě základní funkce. První jsou služby dané vrstvy a druhá funkce je protokol vrstvy [Smutný, Klečka, 2005].

Pro činnost průmyslových komunikačních sběrnic nejsou potřebné (ani žádoucí) všechny vrstvy modelu OSI. Používají se následující vrstvy modelu OSI [Bélai, 2007].



Obr. 19 Jednotlivé vrstvy referenčního modelu ISO/OSI

3.1.1.1 Fyzická vrstva

Úlohou této vrstvy je zajistit přenos jednotlivých bitů na úrovni signálů. Na této úrovni má každá komunikační sběrnice definované fyzické charakteristiky komunikačních obvodů: velikost proudu, přenosovou rychlost, topologii, max. počet připojitelných zařízení, apod.

3.1.1.2 Linková vrstva

Tato vrstva má za úkol zajistit přenos mezi zařízeními, mezi kterými je signálové spojení.

3.1.1.3 Aplikační vrstva

Tato vrstva zabezpečuje překlad požadavku uživatelské vrstvy do linkové vrstvy. Umožňuje přístup do množiny komunikačních služeb podporujících činnost distribuovaných systémů. Může v ní být implementovány příkazy na práci se zařízením (parametrizace, diagnostika).

3.2 Komunikační sběrnice

Za první průmyslovou komunikační sběrnici je možné považovat sběrnici *MIL-STD-1553*, která vznikla v roce 1970 pod záštitou ministerstva obrany USA. V této sběrnici byly nejprve implementovány spodní dvě vrstvy modelu OSI – fyzická a linková. Protokoly aplikační vrstvy byly přidávány postupně tak, aby splňovali požadavky aplikací. V 90. letech minulého století nastal boom v podobě zavádění nejrůznějších komunikačních sběrnic. Vzniklo nepřeberné množství speciálních sběrnic, což mělo negativní následek v podobě malé zaměnitelnosti zařízení od různých výrobců. V důsledku toho nastal proces přirozené selekce nejschopnějších typů průmyslových komunikačních sběrnic, ovšem snaha o vytvoření jediného

standardu průmyslové komunikační sběrnice nebyla úspěšná a proto se i dnes po celém světě můžeme setkat s několika typy komunikačních sběrnic [Berge, 2002].

V následující části je přehledně shrnut současný stav vývoje v dané oblasti, a to formou stručného popisu nejpoužívanějších stávajících standardů průmyslové komunikace kodifikovaných rozhodujícími mezinárodními standardizačními organizacemi, zejména IEC. Autor se zaměří zejména na Ethernet Powerlink, poněvadž právě tato sběrnice je využita při komunikaci jednotlivých modulů řezacího centra.

3.2.1 Průmyslový Ethernet

Již déle než dvacet let je komunikačním de facto standardem v místních sítích (LAN) komunikační síť Ethernet. Postavení standardu získala pro své nesporné kvality, jednak pro to, že byla ve správný čas na správném místě, tj. v období překotného vývoje a rozšíření osobních počítačů byla velmi vhodným prostředkem k jejich propojování do místních sítí.

V průběhu druhé poloviny 80. let minulého století však byly učiněny pokusy využít v té době ještě zcela novou metodu k účelům komunikace v průmyslových řídicích systémech. Příkladem může být síť Sinec H1, uvedená v katalogu firmy Siemens AG již v roce 1985. Šlo o síť plně kompatibilní se standardem IEEE 802.3, avšak již s robustními mechanicky provedenými konektory a s důkladným stíněním koaxiálního kabelu pro propojení PLC řady Simatic S5 [Automatisierungsgeraete, 1985]. Zásadní obrat ve vztahu oboru automatizace k Ethernetu přinesl až vývoj nových internetových technik. Výrobci komponent pro průmyslovou automatizaci viděli příležitost využít velmi efektivní způsoby komunikace zaváděné v oblasti IT pro komunikaci v průmyslu bez nutnosti vkládat do vývoje v této oblasti své vlastní prostředky. Následně se proto od konce 90. let dvacátého století stává Ethernet komunikačním prostředkem i v oblasti průmyslové automatizace.

Obrovskou výhodou průmyslového Ethernetu je, že dnes už je rozvod strukturované kabeláže ve výrobních objektech stejnou samozřejmostí jako rozvod elektrické energie, což přináší obrovskou výhodu právě v rozšíření Ethernetu.

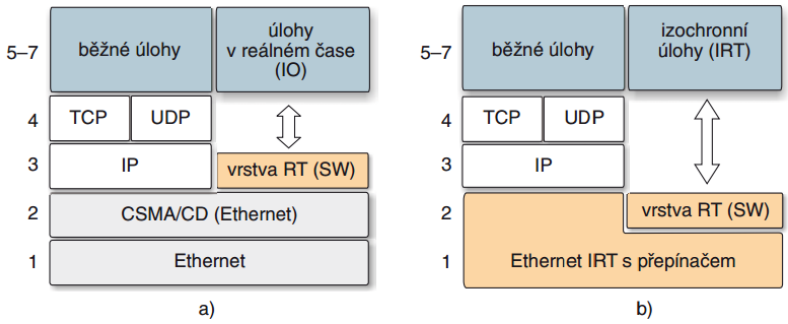
3.2.2 Profinet

Komunikační systém Profinet byl vyvinut organizaci PNO s významným přispěním firmy Siemens a je k dispozici od roku 2002. Na Obr. 20 vlevo je znázorněn komunikační model Profinet V2, který je označován jako *Profinet IO*[10]. Pracuje tak, že standardní zprávy bez požadavků na přenos v reálném čase (*non real-time*) jsou přenášeny standardní cestou *TCP/UDP/IP*[11], zatímco druhý, paralelní kanál obsahuje programové překlenutí *(SW by-pass*) vrstev 3 a 4 komunikačního zásobníku, takže lze dosáhnout dokonalejších vlastností reálného času. K jejich

[10] Profinet IO (*Input/Output*) slouží k realizaci propojení periferií v cyklickém režimu komunikace

[11] Sada protokolů pro komunikaci v počítačové síti

dalšímu vylepšení je u systému Profinet redukována délka přenášeného bloku dat a je zaveden mechanismus prioritních slotů podle standardu IEEE 802.1p (až do priority 7 u komunikace v reálném čase). V systému Profinet verze V3, známém jako *Profinet IRT*[12], je pro vrstvy Ethernetu použit speciální hardware realizující hardwarové překlenutí vrstev TCP/IP. Spolu s přepínanou sítí Ethernet dosahuje Profinet V3 izochronnosti a je vhodný k řízení např. pohonů. Přenos běžných zpráv bez požadavků na přenos v reálném čase, včetně přístupu k internetu, je zajištěn paralelní cestou TCP/UDP/IP [Zezulka, 2008].



Obr. 20 Komunikační modely Profinet: a) Profinet IO, b) Profinet IRT (RT – Real-time) [zdroj: Zezulka, 2008, strana 28]

Celkově probíhá komunikace v systému Profinet ve dvou módech. Prvním módem je tzv. Profinet IO, určený k obsluze distribuovaných jednotek I/O (přenos v reálném čase a izochronní přenos). Druhý mód je označen Profinet CBA, což je přenos zpráv prostřednictvím protokolů TCP/IP bez požadavku na doručení v reálném čase. Dalším rozšířením systému Profinet je zavedení bezpečných mechanismů komunikace ve variantě označené Profisafe.

3.2.3 Ethernet Powerlink

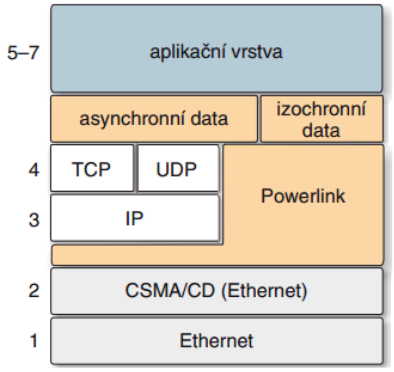
Powerlink nabízejí nekompromisní výkon a funkce reálného času na základě globálního standardu Ethernet. Přenosová rychlost 100 Mb/s a přesnost synchronizace +/- 100 ns umožňují zkombinovat do jediné sítě i ty nejnáročnější úkoly řízení, robotiky, CNC a řízení pohybu. Tento systém vznikl v rakouské automatizační firmě *Bernecker and Rainer* – B&R a je podporován mnoha významnými evropskými výrobci automatizační techniky, organizovanými ve sdružení EPSG. Jde o jeden z nejrozšířenějších standardů průmyslového Ethernetu s velmi dobrými vlastnostmi reálného času nad standardním provedením fyzické a spojové vrstvy Ethernetu.

Z komunikačního modelu tohoto systému (viz Obr. 21) je patrné, že zatímco přenosy časově nekritických dat, jako jsou např. internetové zprávy, se uskutečňují protokoly TCP/UDP/IP, přenosy časově kritických dat (izochronní přenos) probíhají mezi standardní vrstvou 2 a aplikační vrstvou speciálním protokolem při

[12] Profinet IRT (*Isochronous Real Time*) je určen pro úlohy probíhající v reálném čase s tvrdými požadavky na dodržení doby odezvy a synchronizace.

využití principů standardu IEEE 1588[13]. Komunikační zásobník řídí kompletně celý přenos dat v síti. Řídicí metoda má název SCNM a zabezpečuje komunikaci v reálném čase. Každá stanice má přesně stanovená komunikační práva, na jejichž základě může posílat data libovolné stanici na síti. V daném čase má přístup k přenosovému médiu jen jedna stanice. Nemůže tedy docházet ke kolizím a je zajištěn striktně deterministický přenos. Vedle těchto individuálních časových slotů pro operace v reálném čase zabezpečuje SCNM také časové sloty pro standardní časově nekritické zprávy v asynchronním datovém provozu. Vlastnosti bezpečného přenosu jsou realizovány v rozšíření Powerlink safety, a to podle IEC 61508[14]. Varianta Powerlink V2 obsahuje profily automatizačních přístrojů a profily komunikačních podsystémů [Zezulka, 2008].

Spojení pomocí Powerlinku je stěžejním komunikačním prostředkem v této práci, a to konkrétně v komunikaci mezi jednotlivými ostrůvky modulů a hlavním řídicím PLC. Tím je zajištěná vzájemná viditelnost všech modulu z jednoho ovládacího pracoviště a rovněž i pro obsluhu bude jednoduší vše ovládat z jednoho operátorského panelu, než každý modul – kleštinu, vrtačku a pilu ovládat zvlášť!



Obr. 21 Komunikační model Ethernet Powerlink [zdroj: Zezulka, 2008, strana 27]

3.2.3.1 Kabely pro Ethernet Powerlink

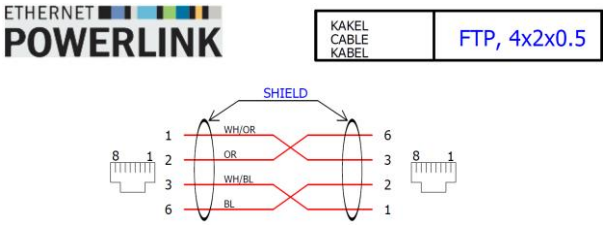
Je zřejmé, že průmyslové prostředí klade na kabely větší požadavky než kancelářské prostředí na standardní síť Ethernet. Potřebné charakteristiky kabelů pro komunikaci v průmyslu byly definovány koncem 80. let minulého století v souvislosti s rozšířením průmyslových komunikačních sítí. Přenos dat v automatizaci a měření má poněkud odlišný charakter než v jiných odvětvích. Požadavkem je spolehlivost, odolnost proti rušení, snadná instalace [Poucha,2006].

V automatizaci se již velmi dlouho používá konektor RJ-45 v průmyslovém provedení až do stupně krytí IP67[15], zpravidla s označením RJ45-IP67. Struktura zapojení jednotlivých žil kabelu i konektoru RJ-45 je zobrazena na Obr. 22.

[13] Standard pro synchronizaci komponent distribuovaných řídicích systémů

[14] Norma pro oblast funkční bezpečnosti elektrických, elektronických systémů

[15] Stupeň krytí udává odolnost elektrozařízení proti vniknutí cizího tělesa či vniknutí kapalin



Obr. 22 Topologie zapojení konektorů RJ-45 u sběrnice Ethernet Powerlink

Z hlediska dlouhodobého použití kabelu pro průmyslový Ethernet je rozhodující provedení jeho pláště. Plášť musí ochránit jádro kabelu před prachem, vlhkostí, agresivními kapalinami a předčasným zestárnutím, tj. obecně zabránit ztrátě požadovaných vlastností jádra. Při volbě nejvhodnějšího provedení a materiálu pláště kabelu musí projektant sítě znát prostředí, v němž se bude kabel nacházet [Zezulka, 2008].

3.2.4 Další systémy průmyslového Ethernetu

Vedle již uvedených komunikačních standardů na bázi systému Ethernet se prosazují i další firemní a jiné systémy, které zatím na trhu nemají takový význam. Jde např. o protokoly HSE, JetSync, Renet, Safeethernet, SynqNet, SynUTC atd., které vesměs posilují jen některou ze slabých stránek Etherentu TCP/IP a nečiní si nárok na roli průmyslové sběrnice jako takové.

Celkově lze konstatovat, že množství úspěšných řešení průmyslových komunikačních sběrnic vycházejících z Ethernetu je důkazem, že průmyslový Ethernet se stane nejvýznamnějším komunikačním prostředkem.

3.2.5 CAN

Jedná se o sériovou sběrnici pro úroveň senzorů a akčních členů. Její přesný popis a zároveň i standard je definován mezinárodní normou ISO-11898, kde je uvedena nejen specifikace elektrického rozhraní (fyzická vrstva), ale i specifikace datového protokolu (linková vrstva).

Architektura protokolu je *multi-master* s asynchronním přenosem, kde jako přístupová metoda je využita modifikovaná CSMA/CD, jenž zaručuje velmi rychlý průchod zpráv s vysokou prioritou (doba odezvy se běžně pohybuje od 100 μs pro přenosovou rychlost 1 Mb/s).

Z obecného modelu komunikace se v případě sběrnice CAN využívá pro systémy distribuovaného řízení v reálném čase aplikační vrstva, která zprostředkovává služby nutné pro chod celého systému. Tyto služby nezahrnují jen distribuci aplikačních a řídících dat, ale také i možnost konfigurace, správy a testování funkčnosti jednotlivých uzlů i celého systému. Pro tuto vrstvu vzniklo v historii sběrnice CAN několik specifikací, z nichž nejpoužívanější je zřejmě *CANopen*.

3.2.5.1 CANopen

CANopen je protokol založený na standardu aplikační vrstvy *CAL*, který je podporován mezinárodním sdružením výrobců průmyslové techniky pro komunikace na sběrnici CAN. Jeho cílové využití je v aplikacích průmyslové automatizace, kde jsou zařízení jako např. distribuované I/O moduly, pohony, serva apod., které komunikují přes sběrnici CAN. Otevřenost a zároveň důsledné zachování kompatibility jsou jedny z hlavních vlastností protokolu CANopen.

Pro přenos informací u protokolu CANopen se využívají dva druhy zpráv: SDO a PDO. SDO (*Service Data Object*) zprávy jsou využívány zejména pro dlouhé přenosy dat s nízkou prioritou přístupu na sběrnici a jsou vysílány asynchronně. Jejich normální využití je pro konfiguraci uzlů. CANopen specifikuje, že každý uzel na síti musí mít nejméně jeden SDO objekt. Pomocí tohoto objektu lze provádět editaci parametrů v tzv. object dictionary, který obsahuje všechny nezbytné parametry určitého modulu.

PDO (*Process Data Object*) zprávy jsou především používány pro přenos dat v reálném čase a typicky mají vyšší prioritu pro přístup na sběrnici CAN než objekty SDO [Bélai, 2007].

3.2.6 Komunikace s elektrickými pohony

Elektrické pohony od nejjednodušších měničů frekvence, až vysoko dynamické servopohony se v čím dál větší míře připojují k nadřízeným řídicím jednotkám, což vede k tomu, že žádaná hodnota (poloha, rychlost, či moment zařízení) je generována z nadřízené úrovně, přičemž aktuální hodnota řízené veličiny je odesílána zpět do automatizovaného systému. Na zabezpečení správné činnosti pohybového systému je potřeba, aby systém na přenos informací umožňoval acyklický a cyklický přenos údajů, synchronizaci hodin komunikujících zařízení a vzájemnou komunikaci mezi pohony.

Cyklicky se přenáší akční zásah a příkazy z řídicí jednotky do pohonu. Na straně druhé se z pohonu do řídicí jednotky přenáší hodnoty monitorovaných veličin a stavy pohonů. Při cyklické komunikaci se přenášejí časově kritické veličiny.

Současně s cyklickou výměnou dat se používá acyklický přenos monitorovaných veličin a parametrů nastavovaných operátorem. Při této, acyklické komunikaci se přenáší údaje, které nejsou časově kritické např. parametry.

V některých aplikačních režimech je možné realizovat cyklický přenos údajů v synchronním režimu, což znamená, že jsou synchronizované periody vzorkování řídicí jednotky, pohonu společně s cyklem sběrnice. Zásluhou synchronizace jsou ve všech pohonech současně vzorkované aktuální veličiny a aktivované vypočítané akční zásahy.

Uvažujme případ, že jsou části algoritmu řízení pohybu distribuované v samotných pohonech. Potom je potřeba se zamyslet nad realizací přenosu údajů i přímo mezi pohony. V takovém případě je nutno zvolit jeden – „master“ pohon, který řídí činnost ostatních pohonů – „slave“. Při takové komunikaci dochází k přenosu údajů mezi pohony bez účasti řídicí jednotky [Bélai, 2007].

4 Vývojové prostředí pro implementaci

Rozhodujícími faktory při zvyšování produktivity a konkurenceschopnosti podniků jsou čas, náklady a kvalita. Cílem každého podniku (nejen z odvětví automatizace) musí být snaha co nejvíce zkrátit dobu od nápadu k hotovému výrobku, při zachování maximální flexibility. Současně je třeba minimalizovat náklady na vývoj a výrobu a zajistit potřebnou kvalitu výsledného produktu. Prostředky pro implementaci automatizovaných procesů v průmyslu vytváří charakteristické propojení jednotlivých dílčích oblastí – zejména projektování řídicích systémů, komunikační techniky, diagnostiky, funkční a informační bezpečnosti a robustnosti. Význam úplného a jednotného prostředí pro přípravu, realizaci a údržbu automatizačního projektu v dnešní době nebývale vzrostl. Uživatelé od takovýchto nástrojů očekávají všestrannou použitelnost, přehlednost, intuitivní a současně praxí prověřený inženýrský softwarový nástroj.

Vývojové systémy pro PLC lze provozovat na téměř libovolném typu PC. Někteří dodavatelé je označují za programovací prostředí pro PLC, termín vývojový systém je ale výstižnější, zejména proto, že současné produkty zdaleka neřeší jen programování, tj. zapsání a opravy programu, jeho odladění, dokumentování a archivaci. Vyspělé vývojové systémy dovolují vytvářet komplexní řídicí systémy.

Obr.23a Vývojové prostředí dnes již umožňuje víc než jen programování [zdroj:BERNECKER & RAINER, 2013]

Od současných vývojových systémů se očekává, že poskytnou programátorům komfort, umožní vysokou produktivitu jejich práce s minimálním rizikem programátorských chyb. K pohodlí přispívá kvalita a logická uspořádanost dialogu s programátorem, systém nápovědy a jazyk, kterým s programátorem komunikuje. Důležité jsou prostředky, které programátora oprostí od zbytných úkonů, např. automatické deklarování proměnných. K dispozici bývají prostředky, které usnadňují ladění programu a řeší diagnostiku programu i řízené soustavy (např. funkce osciloskopu, záznamníku dat – dataloggeru).

Významná je možnost simulovaného běhu programu PLC (virtuálního PLC), která dovoluje ladit program v předstihu, před připojením reálného PLC. Komunikační možnosti současných PLC a vývojových systémů dovolují i práci na dálku, dálkovou diagnostiku, vzdálenou správu programů PLC, komunikaci programu PLC s operátorským rozhraním a s vizualizačními systémy *SCADA*[16]. Jestliže PLC má i funkce webového serveru, je přirozeným požadavkem, aby vývojový systém dovoloval vytvořit jeho webové stránky.

4.1 Norma IEC EN 61131-3

Norma IEC EN 61131-3, určuje formu programových organizačních jednotek (programů, funkcí a funkčních bloků), datové typy, syntaxi jejich deklarací a deklarací proměnných a syntaxi čtyř programovacích jazyků – dvou textových – IL a ST, a dvou grafických – LD a FBD, a dále nástroje SFC pro popis algoritmů sekvenčního chování (viz Tab. 1). Norma je jednotícím prostředkem pro programy různých typů PLC od různých výrobců. Program zapsaný podle zásad normy je „teoreticky“ možné použít pro jakýkoliv typ PLC, jehož výrobce a vývojový systém normu respektují. Přestože přímá přenositelnost programů bývá problematická, důležité je už to, že programu budou rozumět všichni programátoři PLC znalí normy.

Tab. 1 Programovací jazyky pro PLC podle normy IEC EN 61131-3

Zkratka	**Označení v angličtině**	**Označení v češtině**	**Charakteristika, použití**
LD	Ladder Diagram	Příčkový diagram	Názorný, programování jednoduchých logických úloh
FBD	Function Block Diagram	Jazyk funkčních bloků	Názorný, programování logických a regulačních úloh
ST	Structured Text	Jazyk strukturovaného textu	Názorný, univerzální, pro všechny typy úloh
IL	Instruction List	Jazyk seznamů instrukcí	Nenázorný, zdlouhavý zápis, neefektivní
SFC	Sequential Function Chart	Prostředek sekvenčního programování	Velmi názorný a produktivní, programování složitých algoritmů

Norma ale nic neříká o implementaci programu v různých PLC, o způsobu zadávání, ladění a archivace programů PLC. Jejich vlastnosti, možnosti a komfort jsou ponechány zcela na vůli a invenci jejich tvůrců, kteří jsou obvykle i výrobci odpovídajících PLC. Programátor vyzbrojený znalostí normy sice vytvoří

[16] SW, který z hlavního pracoviště sleduje průmyslová zařízení a procesy a umožňují jejich ovládání.

univerzálně použitelný program pro jakýkoliv systém PLC, ale při jeho konkrétní implementaci se musí seznámit se specifickými vlastnostmi vývojového systému.

4.1.1 Základní stavební bloky programu

Základním pojmem při programování podle normy IEC 61131-3 je termín *Program Organisation Unit*[17], zkráceně POU. Jak vyplývá z názvu, POU je nejmenší nezávislá část uživatelského programu. POU mohou být dodávány od výrobce řídicího systému nebo je může popsat uživatel. Každá POU může volat další POU a zároveň může volané POU předávat jeden nebo více parametrů, Existují tři základní typy POU:

- Funkce (FUN) – nejjednodušší POU, funkce může vracet pouze jeden výsledek
- Funkční blok (FB) – může vracet více než jeden výsledek
- Program – představuje vrcholovou programovou jednotku v uživatelském programu. Centrální jednotka PLC může zpracovávat více programů (v různé periodě a s odlišnou prioritou)

Každá POU se skládá ze dvou základních částí: deklarační a výkonové. V deklarační části se definují proměnné potřebné pro činnost POU. Výkonná část pak obsahuje vlastní příkazy pro realizaci požadovaného algoritmu.

4.2 Vývojové systémy PLC

V další části budou popsány vybrané vývojové systémy pro PLC, kde budou zodpovězeny nejdůležitější otázky, které jsou kladeny uživatelem při výběru vývojového prostředí, a to:

- pro které PLC je systém určen,
- které programovací jazyky podporuje (jazyky podle normy a jiné),
- zda podporuje nástroje pro ladění programu a diagnostiku,
- zda dovoluje změny v běžícím programu PLC,
- zda obsahuje prostředky pro vytváření a dokumentování projektu řízení,
- zda dovoluje simulovat program PLC a obsahuje prostředky pro vizualizaci,
- v jakých jazycích komunikuje s uživatelem (jazyk menu a nápovědy) a v jakém jazyce je dostupná dokumentace,
- jaké je rozpětí jeho ceny

4.2.1 ABB: PS501 (CoDeSys)

PS501 je software pro vývoj aplikací PLC od společnosti ABB, založený na platformě CoDeSys. Umožňuje programování podle normy IEC 61131-3 v pěti různých normovaných jazycích, navíc i v *CFC*[18]. Vývojové prostředí je doplněno podpůrnými nástroji pro diagnostiku, servis, vizualizaci apod. Režim off-line umožňuje vývoj aplikace bez připojeného PLC včetně vyzkoušení poruchových

[17] Programová organizační jednotka

[18] Obdoba FBD s volnou grafickou strukturou

stavů a následného nahrání výsledného programu do PLC. Ladění dovoluje krokovat program po jednom kroku nebo cyklu a nastavit body přerušení, tzv. *breakpoints*. Účinným nástrojem je i osmikanálový osciloskop.

Vizualizace obsahuje animované prvky, bitmapy, zobrazování textu, umožňuje zadávat vstupní hodnoty, zobrazovat hodnoty z PLC, správu alarmů a událostí, použít prvky ActiveX. Přístup k vizualizaci je možný z PC s webovým prohlížečem prostřednictvím webového serveru integrovaného přímo v PLC. Konfigurační nástroje pro komunikaci jsou k dispozici pro CANopen, DeviceNet, Ethernet, Modbus a CS31, Profinet, EtherCat,UDP,… Cena základní verze softwaru PS501 je 899 Eur, aktualizace softwaru je zdarma.

4.2.2 B&R: Automation Studio

Vývojové prostředí Automation Studio, dále jen AS, provází uživatele všemi fázemi tvorby projektu a obsahuje nástroje pro správu, údržbu, diagnostiku a servis veškeré automatizační techniky B&R.

Umožňuje vývoj aplikačních programů pro řízení, vizualizaci, komunikaci, řízení pohonů (jednoosých systémů i mnohaosých sestav), CNC, ovládání robotů a úloh funkční bezpečnosti. Software je nezávislý na cílové platformě a dovoluje používat více hardwarových konfigurací v jednom projektu – díky tomu lze pružně vytvářet varianty projektů.

AS pracuje se všemi programovacími jazyky podle normy IEC 61131-3. K dispozici je on-line nápověda a bohatá knihovna funkcí a vzorových kódů (včetně kódů definovaných v normách IEC a PLCopen), navíc je podporována tvorba a správa vlastních knihoven. Propojení se systémy Matlab a Simulink dovoluje automaticky vygenerovat zdrojový kód na základě návrhu a simulace algoritmů.

AS poskytuje nástroje pro ladění a diagnostiku, jako je debugger, zobrazení aktuálních stavů i grafických průběhů proměnných a parametrů nebo seznam chyb. Řídicí program lze otestovat pomocí simulačního PLC na počítači včetně vizualizace a funkcí I/O, pohonů, CNC i robotiky. Výsledný software je možné nahrát za běhu PLC přímo do řídicího systému stroje, uložit na paměťovou kartu nebo vytvořit data pro aktualizaci prostřednictvím USB.

Plná verze tohoto vývojového prostředí stojí řádově stovky Eur, nicméně je zde nutný každoroční tzv, „Update package“, který je rovněž zpoplatněný.

4.2.3 Schneider Electric: Unity Pro

Unity Pro je jednotný inženýrský software pro celý životní cyklus řídicích systémů řady *PAC*[19] od Schneider Electric. Podporuje všech pět normovaných programovacích jazyků podle normy IEC 61131-3 a dovoluje zaznamenávat změny programu v on-line režimu.

Unity Pro obsahuje simulátor a početné funkce pro ladění a diagnostiku, např. nastavení stavů I/O, krokování, verifikaci programu, animační tabulky, vyhledá-

[19] PAC je moderní, výstižnější pojmenování pro programovatelné automaty

vání proměnných nebo nastavení priorit úloh. Grafické rozhraní umožňuje také zobrazovat hardwarové komponenty systému s výpočtem jejich napájení.

Jeho knihovna obsahuje základní objekty (časovače, čítače, matematické funkce), funkční bloky (PID regulátory, motory) a hardwarové, komunikační, diagnostické i procesní objekty. Uživatel si může vytvořit vlastní specifické funkční bloky nebo využít nadstavbu pro programování v jazyce Visual Basic, popř. C++.
K dispozici jsou funkce porovnání dvou projektů, import a export ve formátu XML, nahrání nebo stažení z PAC, konverze ze starších nástrojů pro PLC. Cena tohoto SW se pohybuje v řádech stovek až tisíců Eur, dle zvolené konfigurace.

4.2.4 Siemens: Step 7 v platformě TIA Portal

Nástroj TIA Portal přináší společné prostředí pro vývoj aplikačních programů pro řídicí systémy tvořené programovatelnými automaty a decentralizovanými periferiemi (stanicemi I/O). TIA Portal je určen i pro vývoj operátorských rozhraní pro stroje a zařízení s použitím operátorských panelů (*HMI*[20]) i pro dispečerské systémy (kategorie SCADA). Pro programování PLC slouží nástroj Step7 V11, vytváření aplikací pro HMI systémy umožní nástroj WinCC V11.

Nová verze Step 7 V11 je distribuována ve dvou verzích: Basic a Profesional. Zatímco varianta Basic je orientována na menší projekty s využitím nových řídicích systémů na bázi Simatic S7-1200, varianta Profesional plně podporuje většinu hojně používaných programovacích jazyků podle standardů IEC 61131-3.

Pomocí PLCSim lze bez hardwaru v kanceláři pohodlně připravit např. servisní zásah, provádět různé testy nebo odladit podstatnou část uživatelského softwaru pro PLC. Cena základní verze tohoto vývojového SW se pohybuje v řádech tisíců Eur, což je oproti konkurentům značně dražší řešení, nicméně společnost Siemens je jedním, ne-li největším producentem na poli průmyslové automatizace. V mnoha případech vytváří a nastavuje zcela nové přístupy, z nichž se často stanou obecné standardy daného odvětví.

4.2.5 Teco: Mosaic

Jediný ryze český zástupce průmyslových řídicích systémů. Vývojový systém Mosaic respektuje zásady normy IEC EN 61131-3. Programy lze vytvářet ve všech normovaných jazycích. Významný je simulátor PLC. Na něm lze program vyzkoušet i bez připojeného PLC. Program je možné měnit bez nutnosti zastavit řízení a restartovat systém (změny on-line).

V manažeru projektu lze zadávat konfiguraci buď výběrem modulů anebo rozpoznáním modulů na sběrnici. Pomocí nástroje WebMaker lze vytvořit webovou stránku PLC. WebMaker lze využít i pro vizualizaci v samotném prostředí Mosaic. PanelMaker usnadňuje obsluhu operátorských panelů – textových i grafických. Nástroj PIDMaker je určen k nastavení a simulování PID regulátorů

[20] Označují se takto systémy pro vizualizaci technologických procesů pracujících jako rozhraní mezi technologickým zařízením a jeho obsluhou

a vygenerování potřebného kódu pro PLC. Pro záznam měřených hodnot je určen Datalogger. Průběhy libovolných veličin lze zobrazit v GraphMakeru. Produktivitu programování zlepšují knihovny funkcí a funkčních bloků.

Software Mosaic spolu se základní dokumentací je dodáván ve čtyřech jazykových verzích (CZ, EN, DE, RU). Podrobná dokumentace je v češtině a angličtině. Mosaic je bezplatně dostupný ve verzi Lite, která je omezená jen na velikost sestavy PLC. Verze pro větší sestavy stojí 600 Eur.

4.3 Závěr kapitoly

Vzhledem ke skutečnosti, že dodavatel veškerého HW vybavení řezacího centra jako je PLC, vstupní/výstupní karty, měnič frekvence, dotykový panel, aj. je předem určen, není v tomto konkrétním případě možnost volby.

Dodané HW vybavení je od společnosti B&R, proto i implementace aplikace systému řízení bude realizována ve vývojovém prostředí Automation Studio od společnosti B&R.

5 Metodika

Implementace aplikace bude realizována pomocí funkčních bloků a modulů. Jednotlivé části celého procesu budou postupně programovány do jednoho řídicího celku, který pomocí nakonfigurovaných interfaců bude ovládat řízené soustavy (subprocesy) a informovat obsluhu o aktuálním dění na operátorském panelu. Jak již bylo popsáno dříve, implementace proběhne ve vývojovém prostředí Automation Studia od společnosti B&R.

Jelikož hlavním cílem této práce je vytvoření funkční aplikace, musí metodika práce toto reflektovat. Každá aplikace si musí projít několika fázemi životního cyklu. Těmito fázemi jsou:

- Specifikace problému,
- Analýza a návrh aplikace,
- Implementace aplikace,
- Testování,
- Provoz a údržba.

Fáze specifikace problému představuje většinou slovní popis funkce aplikace. Další fází je analýza a návrh aplikace. Tyto fáze jsou často propojeny, protože běžně během analýzy dochází k hrubému návrhu aplikace. Cílem těchto fází je dekompozice hlavního problému na jednotlivé podproblémy a návrh řešení jednotlivých podproblémů.

Poté následuje fáze implementace aplikace, kdy je za použití zvoleného jazyku (může jich být i víc) a nástrojů SW vybavení, aplikace programována. Součástí této fáze by mělo být i pravidelné testování, které přináší tu výhodu, že během programování se můžou objevit jisté nepříjemnosti, a právě díky testování je můžeme minimalizovat či úplně odstranit.

Nakonec po provedení celkového testování aplikace dochází k uvedení do provozu, což je považováno za klíčový moment celého vývoje. Teprve zde se zjistí, zda navržený postup byl správný. Poté už programátor provádí pouze údržbu aplikace, případné dílčí změny podle požadavku zadavatele.

5.1 Tvorba řídicího programu

Tvorba řídicího programu se skládá z několika kroků:

- Studium HW vybavení centra s důrazem na ovládání měniče frekvence,
- Seznámení se s vývojovým prostředím Automation Studio,
- Programování funkčních bloků pro vykonávání jednotlivých dílčích subprocesů (řízení pneumatických válců, spouštění *FI*[21], polohování kleštiny s FI po trati aj.) s využitím knihovny „FB_*Motorky*“,

[21] Z anglického *Frequency inverter* – měnič frekvence

- Vytvoření řídicího programu, provázanost s funkčními bloky pomocí interfaců,
- Testování funkčnosti programu v simulačním prostředí *PLCSim*, které je součástí Automation Studia,
- Uvedení řezacího centra do provozu a nastavení všech parametrů nutných pro bezchybný chod centra (seřízení koncových snímačů, nastavení referenčních a limitních pozic, apod.),
- Testování funkčnosti modulu kleštiny v reálných podmínkách,
- Optimalizace PID regulátoru FI kleštiny pro plynulý a stabilní chod ve všech pásmech rychlostí při různých hmotnostech naváženého materiálu

Problematika je studována z literárních a internetových zdrojů a rovněž také z oficiální technické dokumentace [BERNECKER & REINER, 2014].

5.2 Tvorba vizualizace

Tvorba vizualizace byla rozdělena do následujících kroků:

- Studium vytváření vizualizačních obrazovek pro průmyslové aplikace,
- Návrh a tvorba graficky jednoduchých a intuitivních komponent pro obrazovky modulu kleštiny (tlačítka, ikony stavů akčních členů, datová pole s ukazateli, signálky aj.), které lze snadno vytvořit za použití grafického editoru typu *WYSIWYG*[22],
- Vytvoření vizualizačních obrazovek a propojení s odpovídajícími proměnnými z řídicího programu
- Testování funkčnosti a reakcí v simulačním prostředí *PLCSim*,
- Nahrátí vizualizace do operátorského panelu Power Panel 500 a testování funkčnosti a odezvy na jednotlivé pokyny z panelu.

Pro vytvoření výsledných vizualizačních obrazovek je využita oficiální technická dokumentace [BERNECKER & REINER, 2014].

5.3 Závěr kapitoly

Cílem této kapitoly bylo seznámení se s hlavními přístupy práce nutných k vytvoření výsledného programu a vizualizace.

Poslední část této práce se bude zabývat shrnutím a zhodnocením dosažených výsledků a potvrzením či vyvrácením splnění cílů práce. Navíc budou rozebrány možnosti budoucích rozšíření.

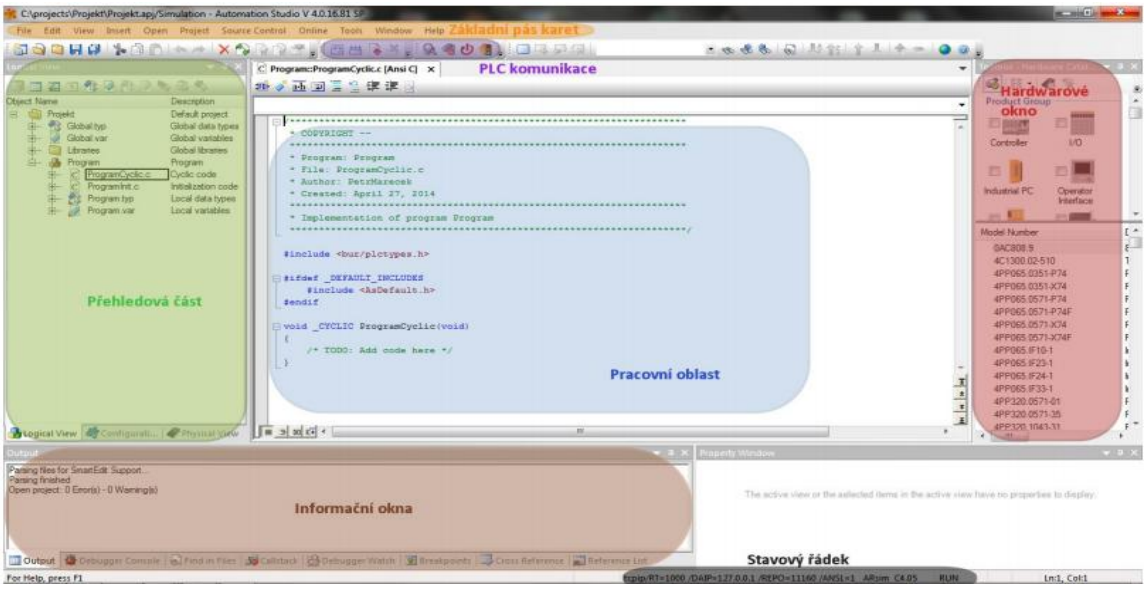
[22] Způsob editace nejen grafických prvků (v doslovném překladu „co vidíš, to dostaneš“)

6 Vlastní práce

V této kapitole bude popsána implementace řídicí aplikace (program a vizualizace), která tvoří stěžejní část této práce, které spolu budou spolupracovat při řízení modulu řezacího centra.

6.1 Základy práce s Automation Studiem

Na Obr. 23 lze vidět, standardní rozložení AS s oddělenými jednotlivými okny pro práci s projektem. V horní liště je zobrazen **Základní pás karet**, kde je možno jednotlivé sekce různě škálovat, přidávat nové nebo připnout na některou z lišt dle libosti programátora.

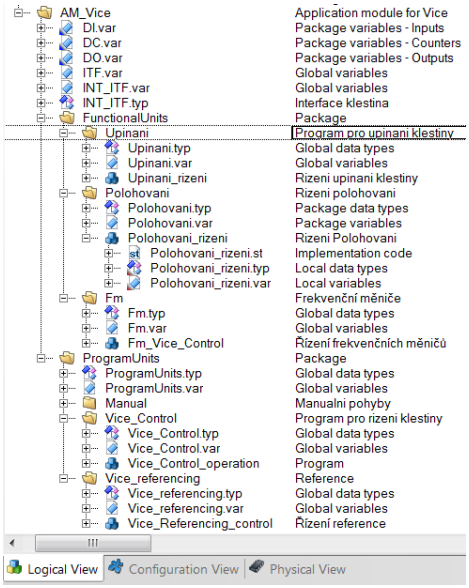


Obr. 23 Prostředí AS po vytvoření nového projektu

Vlevo, zeleně ohraničená **Přehledová část**, je rozdělena na 3 přepínatelné záložky. První z nich – *Logical view* vyplňuje z velké části stromová struktura projektu se všemi potřebnými komponentami (detailní zobrazení struktury vytvářené aplikace je naznačeno na Obr. 24). Soubor „*Global.var*“ obsahuje všechny globální proměnné deklarované v projektu. Dále pak složka „*Libraries*“, ve které jsou uloženy veškeré knihovny využívané v projektu, jak od firmy B&R (na řízení servopohonů, převody jednotek, ovládání pohonů, …) tak i uživatelem definované knihovny. Veškerá logika aplikace je skryta pod položkou „*Program*“, kde se dále větví. Každý program se skládá ze dvou celků – INIT část a CYCLIC část.

Inicializační část programu proběhne pouze jednou po nahrání programu do PLC – využívá se k předdefinování výchozích hodnot, nadefinování referenčních poloh stroje, akceleračních parametrů pohonů, apod. Cyklická část programu, jak již název napovídá, se vykonává stále dokola v pevně daných časových cyklech –

Task Class, přičemž se dbá na to, aby se kritické subprocesy (řízení měničů frekvence, polohování serv, aj) vykonávali v nejrychlejším *tasku*, a naopak ty méně důležitě subprocesy nezatěžovali tolik procesor a vykonávali se v méně prioritních časových cyklech. Minimální délka cyklu je různá podle zvoleného typu CPU, ale běžná je hodnota 10-100 ms.



Obr. 24 Stromová struktura vytvářené aplikace v prostředí Automation Studia

V druhé záložce *Configuration view* se nachází HW/SW konfigurace a mapování vstupů/výstupů. Konfigurací můžeme vytvářet libovolné množství, avšak aktivní může být vždy jen jedna. Toto řešení nám dovoluje vytvořit zvlášť konfiguraci pro simulaci, zvlášť několik konfigurací pro reálné podmínky, kdy až na místě u stroje, poté co se všechny konfigurace otestují, se vybere ta nejideálnější pro dané provozní podmínky.

Na poslední záložce *Physical view* se skládá hardwarová sestava podle skutečného PLC, umístěného v rozvaděči. Jednotlivé moduly (vstupní/výstupní karty, čítače, napájecí zdroje, komunikační karty,…) se přiřazují liniově, tak jak jsou zapojeny za sebou v reálném PLC. V konfiguraci I/O karet potom nastavíme proměnné (binární výstup na *DO*[23] kartu, binární vstup na *DI*[24] kartu).

Pravé, **Hardwarové okno** s volbou jednotlivých modulů, slouží pro rychlejší vyhledávání karet s jejich následným vložením do kompletní sestavy. Vložení probíhá jednoduše stylem *Drag & Drop*. Poté už je jen potřeba nakonfigurovat jednotlivé kanály na kartách odpovídajícími proměnnými.

Vlastní program je psán v **Pracovní oblasti**, která podporuje zvýrazňování syntaxe a pomoc našeptávače podle právě používaného programovacího jazyka.

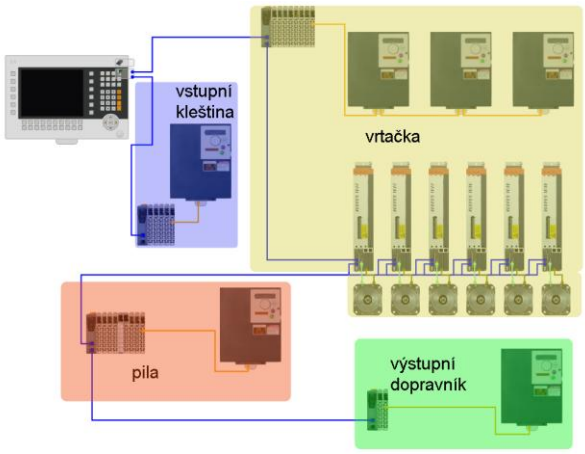
[23] Výstupní karta, běžně osazená 8 binárními výstupy

[24] Vstupní karta, běžně osazená 12 binárními vstupy

Oblast **Informačních oken** informuje programátora o aktuálním dění v programu. Pomocí stavových řádků stručně podává informace, upozorňuje na chyby v kódu, či nepoužití nadeklarovaných proměnných apod. Dále pak lze v této části *debuggovat* jednotlivé kusy kódu řádek po řádku a přiřazovat jim *breakpointy* pro snazší nalezení chyb v programu.

PLC komunikace, jež je označena fialovou barvou, slouží k nahrání programu do PLC, pokud je zvolená konfigurace s reálným strojem, popřípadě do simulátoru, který supluje takovýto reálný stroj. Kliknutím na tlačítko *Build* se zkompilují všechny softwarové objekty (programy, knihovny, atd.) v aktivní konfiguraci. Tlačítko *Monitor* spustí monitorovací mód pro sledování aktuálních hodnot v proměnných.

Stavový řádek v dolní části okna říká s jakým PLC komunikujeme a v jakém stavu se aktuálně připojené PLC nachází (*Run/Offline/Service/Simulation*).



Obr. 25 HW konfigurace celého centra s jednotlivými ostrůvky řízených modulů

6.1.1 ST – jazyk strukturovaného textu

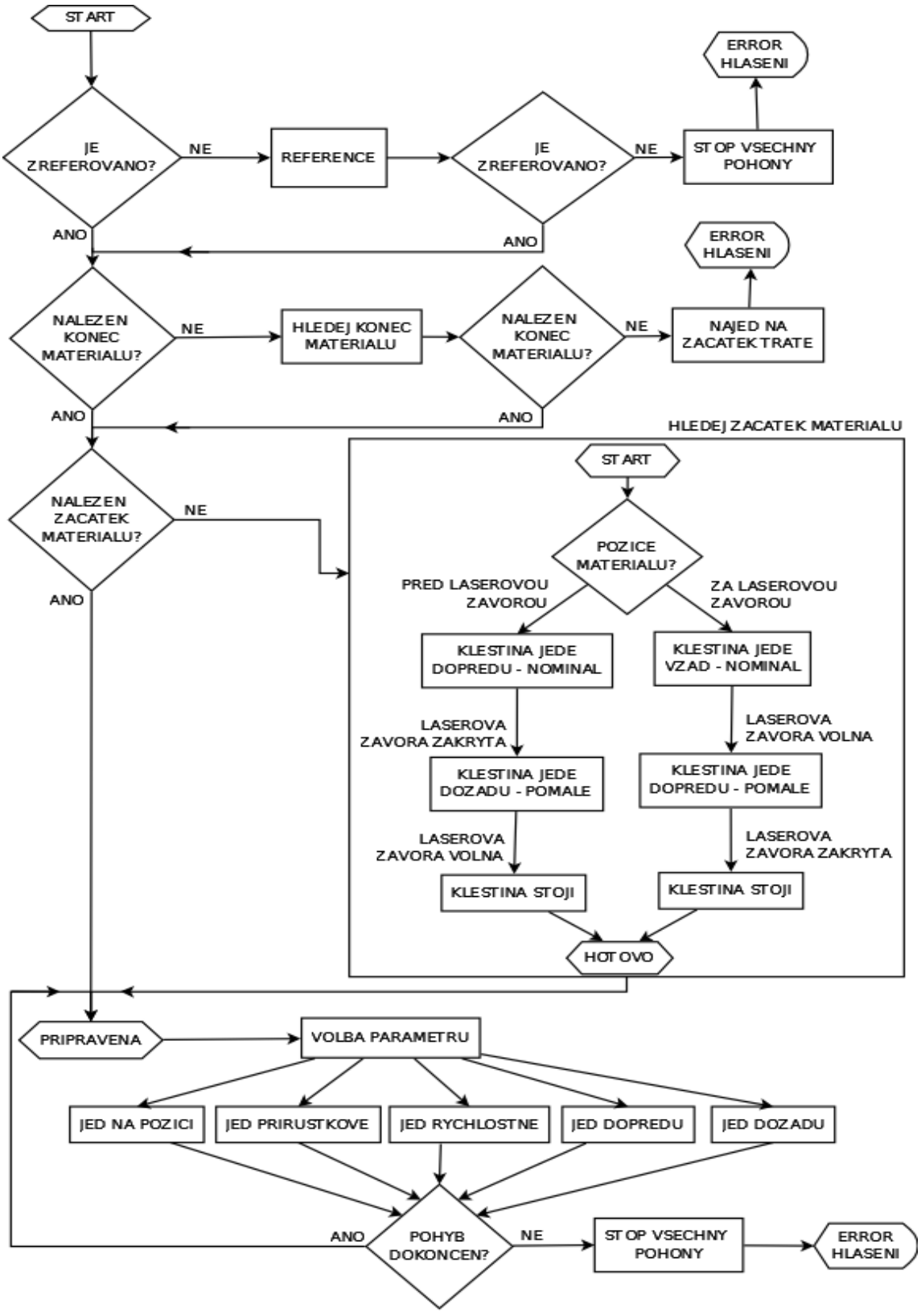
Jazyk strukturovaného textu (z anglického *Structued Text*) je velmi výkonný vyšší programovací jazyk, který se svou syntaxí podobá Pascalu. I přes tuto podobnost s Pascalem je ST samostatný programovací jazyk vyvinutý pro průmyslové řídicí aplikace. Může být použit k vyjádření chování funkcí, funkčních bloků a programů.

Obsahuje všechny podstatné prvky moderního programovacího jazyka, včetně větvení (*IF-THEN-ELSE* a *CASE OF*) a iterační smyčky (*FOR*, *WHILE* a *REPEAT*). Tyto prvky mohou být vnořovány.

Jednotlivé řádky programu mohou být psány v libovolném stylu. Mezi klíčová slova a identifikátory mohou být vkládány tabelátory, znaky konce řádků a komentáře. Jazyk je dobře čitelný a přehledný [Koziorek, Chromčák, 2007]. Modul kleštiny, včetně všech funkčních bloků bude psán jazykem ST.

6.2 Navrhované řešení

V této kapitole je nastíněn algoritmus řízení modulu kleštiny (viz Obr. 26), který bude sloužit jako základ pro programování při tvorbě výsledné aplikace.



Obr. 26 Vývojový diagram modulu kleštiny řezacího centra

Po zapnutí stroje a sepnutí relé bezpečnostního okruhu chvíli trvá, než se všechny moduly řezacího centra zinicializují. Hotová inicializace a úplná připravenost modulů je signalizována na každé kartě zvlášť a to rozsvícením zelené LED do stavu *Ready*. Kdyby LED signalizace problikávala, či svítila červeně, nastala někde chyba a musí se odstranit – nelze pracovat se strojem, který nemá všechny prvky zinicializované a tedy připraveny k použití.

Dalším důležitým bodem je nastavení referenčních a koncových poloh, pro pohyb kleštiny po válečkové trati. Tento úkon obnáší pojezd kleštiny do koncových pozic na obou stranách tratě a fyzické odměření vzdálenosti čelistí kleštiny od nulového bodu (nulový bod byl zvolen pás pily při natočení 90°). Těmto proměnným se nastaví příznak *Retain*, což označuje blok v paměti, kde jsou proměnné uchovány i při vypnutém stroji a nedochází k jejich ztrátě při opětovném uvedení do provozu. Dalším krokem je nastavení rychlostí pohonů (nominální, minimální, maximální), akcelerace/decelerace, módu zvoleného při referování (více v kapitole 6.3.1.2 Polohování (FB_Moving_Vice)), časových konstant při řízení válců – timeout, čas pro upnutí, čas pro otevření, aj.)

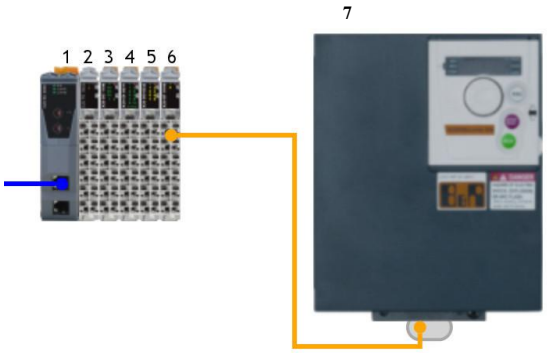
Mějme na paměti, že při prvotním spuštění stroje a nastavení všech nutných parametrů, se kleština nachází někde na válečkové trati, ale nevíme přesně kde, tudíž jakýkoliv pokyn pro pohyb kleštiny by měl vyvolat chybové hlášení, že kleština není zreferovaná. Jakmile je provedena reference, již je možno s kleštinou v manuálním či servisním režimu libovolně manipulovat.

Dále je žádoucí, v případě automatického režimu nalézt konec a začátek materiálu, který na válečkovou trať navezla obsluha řezacího centra. V případě, že bychom se pokoušeli nalézt konec materiálu, ale ve skutečnosti by na válečkové trati žádný materiál nebyl, tak dojetí kleštiny na laserovou závoru (ta se nachází na začátku vrtacího centra) by bylo impulsem pro návrat kleštiny do výchozí polohy a vyhlášení chyby ve smyslu – není žádný materiál k navezení. Situace kdy není žádný materiál na trati a kleština započne cyklus hledání konce, by normálně neměl nastat, nicméně musíme tuto situaci brát v potaz a provést patřičné kroky k ošetření tohoto stavu.

Po nalezení konce následuje nalezení začátku materiálu, které spočívá v dojetí s upnutým materiálem na laserovou závoru. Poté je již vše připraveno k obrábění (vrtání, řezání) podle předem nadefinované receptury z operátorského panelu.

6.2.1 Sendvičový systém distribuovaných I/O modulu kleštiny

V této kapitole se podíváme na uspořádání karet v rozvaděči a jejich konkrétní využití u modulu kleštiny. Na jednotlivé svorky vstupních karet jsou přivedeny signály ze vstupních snímačů. Na svorky výstupních karet jsou přivedeny ovládací signály akčních členů. Jednotlivé karty spolu komunikují po sběrnici X2X a vzdálenost dvou karet může být až 100 m. Využívá se zde sendvičový systém propojení, kdy jsou karty zapojeny modulárně vedle sebe a kompletně tvoří tzv. ostrůvek (viz Obr. 27). Stavy vstupů/výstupů jsou signalizovány na čelních panelech karet diodami LED. Rozsvícená LED indikuje stav logické 1.



Obr. 27 Sendvičový systém zapojení I/O karet s měničem frekvence pro modul kleštiny

Popis jednotlivých karet ostrůvku kleštiny:

1. **BC0083**

   Karta, použitá jako řadič sběrnice zajišťující komunikaci mezi ostrůvkem (X2X sběrnice) a ovládacím panelem (Powerlink). Karta je vybavená dvěma RJ45 porty s max. přenosovou rychlostí 100 Mbit/s a nejkratší dobou cyklu 200 µs.

2. **PS9400**

   Napájecí karta pro ostrůvek, napájecí napětí 24V.

3. **DC1196**

   Čítačová karta pro potřeby inkrementálního enkodéru s 5 V výstupem. Vybavena je navíc 2 digitálními vstupy (možno využit pro signály koncových nebo referenčních pozic). Napájení je 24 V.

4. **DO8332**

   Karta pro 8 digitálních výstupů se jmenovitým výstupním proudem 2 A, a napájením 24 V. Na této kartě je zapojen pneumatický válec pro otevírání/upínání čelistí (DO_Cylinder viz Obr. 28).

   Zpozdí-li se vlivem nějaké poruchy pravidelný programový cyklus, vynulují se po uplynutí času pro programový cyklus výstupní paměti a nastavovací výstupní členy v řízeném procesu se uvedou do výchozího stavu. To zabraňuje vzniku nebezpečných provozních stavů.

5. **DI9371**

   Karta pro 12 digitálních vstupů, napájena 24 V. Na této kartě jsou napojeny všechny senzory (viz Obr. 28), tj. koncové limitní snímače vepředu/vzadu, referenční snímač, indukčnostní snímač pro detekci materiálu v čelistech, snímače otevřených/upnutých čelistí, snímač zaclonění laserové závory.

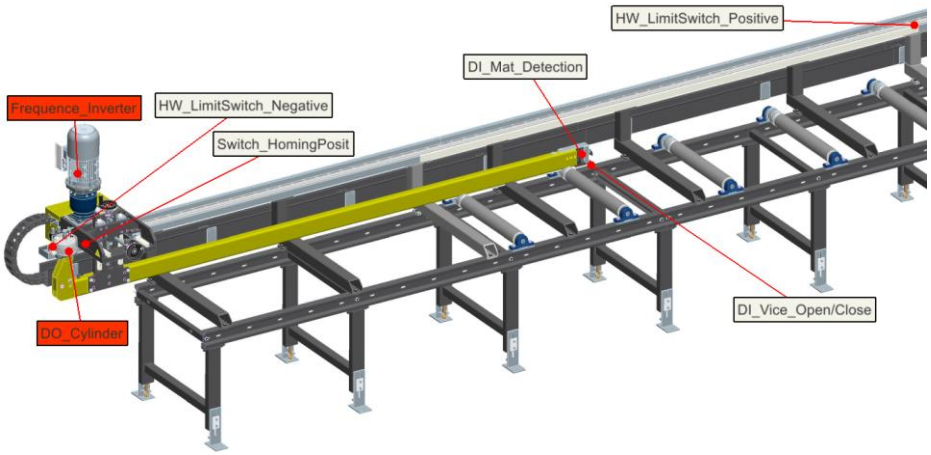
6. **BT9100**

   Karta zajišťující komunikaci s měničem frekvence pomocí X2X sběrnice.

7. **Acopos Invertor X64**

   Měnič frekvence se jmenovitým výkonem 1.5 kW, zapojením 3x400V a max. dodávaným proudem 8.4 A. Měnič X64 je vybaven různými digitálními

a analogovými kanály I/O k řízení rozličných provozních funkcí stroje. Měnič je vybaven rovněž ovládanou brzdou. Při začleňování měniče do automatizačního projektu pomáhá uživateli asistenční nástroj *Motion Wizard*, který také provede základní nastavení nejdůležitějších parametrů přístroje. Tím se vytvoří předpoklady ke snadnému následnému uvedení měniče do provozu.



Obr. 28 Model kleštiny s rozložením snímačů a akčních členů na válečkové trati

6.3 Implementace – Řídicí blok

Aplikace je rozdělena na funkční část a programovou část a je implementována jako modul, tzn., bude vytvořen hlavní řídicí program, který bude přistupovat pomocí programového rozhraní k jednotlivým ovládaným funkčním subprocesům. Toto řešení si klade za cíl vysoký stupeň modularity, jenž má velký potenciál v tom, když by si zákazník přál vytvořit řezací centrum z jiných komponent (místo kleštiny např. podavač s přeuchopováním, či namísto natáčecí pily by mu stačila jen běžná pila bez natáčení), tak v případě napevno naprogramované aplikace by musel programátor napojit všechny vstupy/výstupy na nové bloky, všechny vizualizační objekty by se museli ručně přemapovat na nové proměnné a celkově práce by byla těžkopádná a zdlouhavá.

Naopak při implementaci s využitím modulárních systémů stačí nevyužitý modul programově skrýt, namapovat modul nový, a jen napojit jednotlivé programové rozhraní a tím značně ulehčit práci programátora a v konečném měřítku zrychlit celý proces dodání hotového stroje.

6.3.1 Funkční část

Funkční část se stará o funkční podstatu ovládaných subprocesů. Jedná se o ovládání pneumatických válců, inicializaci a spouštění/vypínání měniče frekvence a polohování s kleštinou pomocí měniče frekvence.

Například je zadán příkaz `RS_Clamping_Vice.Cmd.Open` (otevřít čelisti upínače). Funkční blok provede daný příkaz, tedy otevře čelisti a přestaví svůj výstupní stav (`RS_Clamping_Vice.State.Open`) do logické 1. To, jak na vzniklou situaci má v automa-

tickém cyklu kleština v danou chvíli reagovat, je otázkou programové části. Funkční blok tohle už neřeší, ten svoji část splnil – dostal příkaz, provedl jej a přestavil odpovídající výstup. V této kapitole budou stručně vysvětleny veškeré naprogramované funkční bloky s ukázkami kódu.

6.3.1.1 Upínání (FB_Clamping_Vice)

Funkční blok zajišťující pohyb pneumatického válce upínacích čelistí kleštiny. Upnuté nebo naopak otevřené čelisti mohou být signalizovány dvěma způsoby. Může se jednat o koncové snímače či tlakové snímače. Toto je potřeba funkčnímu bloku (dále jen FB) říct. K tomu slouží struktura `Mode`. Následující kód ukazuje, jednu z možností nastavení parametrů ve struktuře `Mode` (snímač pro otevření čelistí je koncový, snímač upnutí materiálu je tlakový):

```
RS_Clamping_Vice.Mode.EndSwitchClose                := FALSE;
RS_Clamping_Vice.Mode.EndSwitchOpen                 := TRUE;
RS_Clamping_Vice.Mode.PressureSwitchClose           := TRUE;
RS_Clamping_Vice.Mode.PressureSwitchOpen            := FALSE;
```

Pro nastavení časových konstant pro pohyb čelistí je připravena datová struktura `Setting`. Parametry se nastavují v milisekundách. Cílem těchto konstant je ošetřit stavy, kdy se zadá příkaz například pro otevření čelistí (`RS_Clamping_Vice.Cmd.Open`), ale příkaz se nestihne v daném časovém limitu vykonat (důvodů může být několik: chyba v kabelu, vadný snímač, nesprávně zapojené svorky, atd.) Proto je potřeba s touto problematikou počítat a vhodně reagovat. Řešením jsou právě časové konstanty, jež využívají FB `TON`[25] pro vytvoření časového rámce nutného k vykonání příkazu. Pokud by výstupní člen do definovaného časového limitu nebyl přestaven, vyhlásí blok chybu a obsluha provede potřebné kroky k jeho odstranění.

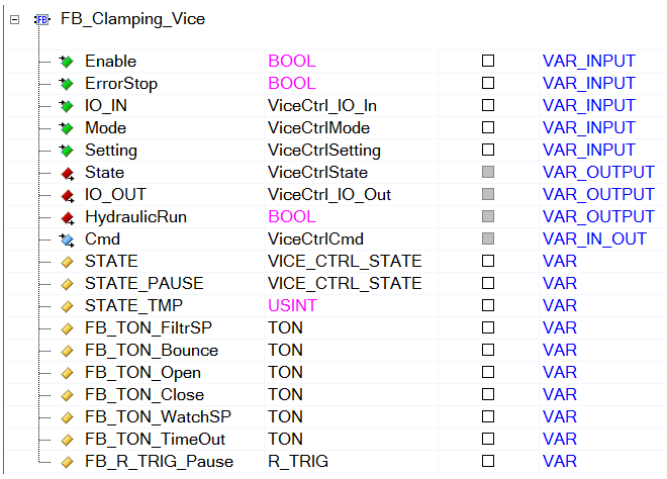
```
Retain_ClampVice.Setting.TimeClosing            := 2000;
Retain_ClampVice.Setting.TimeOpening            := 2000;
Retain_ClampVice.Setting.TimeFiltrSP            := 1000;
```

Parametry `Mode` a `Setting` se inicializují v *INIT* části programu, protože můžou být během chodu stroje pozměněny. Naopak co libovolně upravovat nelze jsou signály z fyzických snímačů, které jsou přivedeny na konkrétní svorky vstupních karet. Tyto signály ze snímačů, jsou v přímém styku s reálným měřeným prostředím a je třeba je rovněž namapovat, avšak již v části *CYCLIC*. K tomuto je určena struktura `IO_Mapping`:

```
RS_Clamping_Vice.IO_Mapping.In.SensorClose      := DI_SP_Vice_Close;
RS_Clamping_Vice.IO_Mapping.In.SensorOpen       := DI_SQ_Vice_Open;
```

[25] Výstup z bloku se aktivuje po náběžné hraně vstupního signálu a uplynutí časové prodlevy

Struktura předávaných parametrů u `FB_Clamping_Vice` musí být totožná se strukturou řídicího systému `RS_Clamping_Vice`. Navíc jsou zde implementovány interní lokální proměnné a časové funkční bloky `TON` sloužící pro stanovení časových rámců jednotlivým příkazům, jak je ukázáno na Obr. 29.



Obr. 29 Deklarace proměnných funkčního bloku pro ovládání pneumatických válců

V *CYLIC* části programu se všechny parametry předají přes interface danému FB, který je zpracuje, přestaví své logické výstupy a tyto výstupy jsou přivedeny na výstupní svorky akčního členu, tedy pneumatického válce (`DO_PV_Vice_Close`/ `DO_PV_Vice_Open`).

```
FB_Clamping_Vice.Mode                := R_Clamping_Vice.Mode;
FB_Clamping_Vice.IO_IN               := RS_Clamping_Vice.IO_Mapping.In;
FB_Clamping_Vice.Setting             := Retain_ClampVice.Setting;
FB_Clamping_Vice(Cmd                 := RS_Clamping_Vice.Cmd);

RS_Clamping_Vice.IO_Mapping.Out      := FB_Clamping_Vice.IO_OUT;
RS_Clamping_Vice.State               := FB_Clamping_Vice.State;
RS_Clamping_Vice.NeedHydraulic       := FB_Clamping_Vice.HydraulicRun;
//IO_OUT
DO_PV_Vice_Close                     := RS_Clamping_Vice.IO_Mapping.Out.ClosingValve;
DO_PV_Vice_Open                      := RS_Clamping_Vice.IO_Mapping.Out.OpeningValve;
```

6.3.1.2 Polohování (FB_Moving_Vice)

Funkční blok zajišťující veškerý pohyb kleštiny po válečkové trati. Pro samotné ovládání asynchronního motoru přes měnič frekvence je využita knihovna *FB_Motorky*.

Knihovna slouží pro polohování s krokovým motorem, měničem frekvence nebo stejnosměrným motorem. Je vyvinuta brněnskou pobočkou firmy B&R a vhodná zejména pro karty MMxxx zajišťující řízení stejnosměrných motorů, karty SMxxx, DSxxx pro řízení krokových motorů a pro řízení měničů frekvence.

Knihovna poskytuje absolutní a relativní pozicování, rychlostní pohyb, tří různé způsoby reference, podporu HW a SW limitů, konverzi jednotek (z m/min na *unit/sec*[26]), aj.

V *INIT* části programu je potřeba nastavit veškeré atributy jako: rychlosti motoru (nominální, minimální, maximální), akcelerace/decelerace, způsob referování, časové limity a zejména parametry pro ovládání PID regulátoru (K_p a T_n konstanty). Ukázka kódu z inicializační části programu s nastavováním potřebných atributů vypadá následovně:

```
RS_Moving_Vice.Param.PositionTolerance                      := C_POSITION_TOLERANCE;
RS_Moving_Vice.Param.SwLimitTolerance                       := C_SW_LIMIT_TOLERANCE;

Retain_Moving_Vice.Setting.Acceleration.Nominal             := 1200;
Retain_Moving_Vice.Setting.Decceleration.Nominal            := 1200;
Retain_Moving_Vice.Setting.MotrokyInitEnc.maxLagError       := 50000;
Retain_Moving_Vice.Setting.MotrokyInitEnc.reverseDirection  := FALSE;
Retain_Moving_Vice.Setting.MotrokyInitEnc.taskTime          := 0.004;
Retain_Moving_Vice.Setting.MotrokyInitEnc.encoderIncrements := 4000;
Retain_Moving_Vice.Setting.MotrokyInitEnc.maxAcceleration   := 500000;
Retain_Moving_Vice.Setting.MotrokyInitEnc.units             := 10000;
Retain_Moving_Vice.Setting.PID_Sets[0].Kp                   := 0.22;
Retain_Moving_Vice.Setting.PID_Sets[0].Tn                   := 0;
Retain_Moving_Vice.Setting.PID_Sets[1].Kp                   := 0.5;
Retain_Moving_Vice.Setting.PID_Sets[1].Tn                   := 0;
Retain_Moving_Vice.Setting.Speed.Homing1                    := 8;//7
Retain_Moving_Vice.Setting.Speed.Homing2                    := 4;//6
Retain_Moving_Vice.Setting.Speed.Max                        := 32;
Retain_Moving_Vice.Setting.Speed.Nominal                    := 10;
Retain_Moving_Vice.Setting.TimeTimeout                      := 1000;
Retain_Moving_Vice.Setting.TimeBrake.TimeOff                := 500;
Retain_Moving_Vice.Setting.TimeBrake.TimeOn                 := 500;
```

Všechny parametry jsou voleny na základě testování s ohledem na technickou specifikaci motoru. Parametry PID regulátoru jsou ve dvou strukturách a rozdílné jsou proto, poněvadž skupina parametrů v první struktuře `PID_Sets[0]` slouží pro polohování na dlouhé vzdálenosti, a skupina parametrů obsažená v `PID_Sets[1]` slouží pro polohování na krátké vzdálenosti. Lze říci, že na druhou skupinu parametrů kleština jen dopolohovává, aby dosáhla požadované polohy s danou přesností (parametr `RS_Moving_Vice.Param.PositionTolerance`).

Pro potřeby reference je nutno nadefinovat v jakém pracovním pásmu vzdáleností od nulového bodu se bude kleština pohybovat. Po úspěšné referenci se zinicializují obě koncové softwarové pozice kleštiny (parametr `NegativeSwEnd`

[26] Interní jednotka rychlosti měniče frekvence

a `PositiveSwEnd`). Jak již bylo řečeno, nulový bod pro celé řezací centrum je pás pily ve výchozí poloze, tj. při natočení 90° vůči ose otáčení.

```
Retain_Moving_Vice.Setting.Position.Homing                  := 9580;
Retain_Moving_Vice.Setting.Position.NegativeSwEnd           := 650;
Retain_Moving_Vice.Setting.Position.PositiveSwEnd           := Retain_Moving_Vice.Setting-
.Position.Homing - 10;
```

FB_Motorky nám dovolují několik metod reference, v následující části jsou jednotlivé možnosti vysvětleny a poté zvolena nejvhodnější metoda.

- Homing.Mode:

  0 (Direct) – hodnota aktuální pozice kleštiny se stane referenční, neprobíhá žádný pohyb

  1 (End_Switch) – Referování na koncové snímače

  2 (Ref_Switch) – Referování na referenční snímač, změna pohybu referování při dosažení koncového snímače

- Homing.StartDirection:

  Homing.Mode = 1:

  V tomto případě parametr `Homing.StartDirection` nemá význam

  Homing.Mode = 2:

  0 Referování začíná v kladném směru

  1 Referování začíná v záporném směru

- Homing.EdgeSwitch:

  Homing.Mode = 1:

  0 Referování směrem na koncový snímač v kladném směru

  1 Referování směrem na koncový snímač v záporném směru

  Homing.Mode = 2:

  0 Referování na nástupnou hranu referenčního snímače

  1 Referování na sestupnou hranu referenčního snímače

- Homing.TriggerDirection

  Homing.Mode = 1:

  0 Sekundární pohyb reference je v kladném směru

  1 Sekundární pohyb reference je v záporném směru

  Homing.Mode = 2:

  0 Sekundární pohyb reference je v kladném směru

  1 Sekundární pohyb reference je v záporném směru

Kleština je vybavena referenčním snímačem, který jsou umístěn před koncovým snímačem na samotném konci kleštiny. Nastavení parametrů určující způsob referování vypadá následovně:

```
Retain_Moving_Vice.Setting.Homing.Mode             := 2; // referovani na ref. koncak
Retain_Moving_Vice.Setting.Homing.EdgeSwitch       := 0; // ref. na nastup hranu
Retain_Moving_Vice.Setting.Homing.StartDirection   := 0; // referovani dozadu
Retain_Moving_Vice.Setting.Homing.TriggerDirection := 1; // po ref. sjede dopredu
```

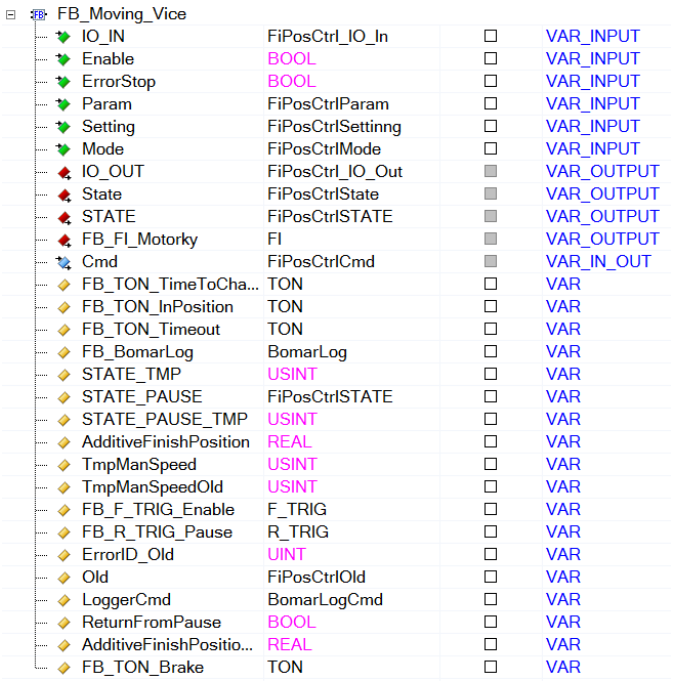
V *CYLIC* části programu se všechny parametry předají přes interface danému FB, který je zpracuje, přestaví své logické a analogové výstupy a tyto výstupy jsou přivedeny jako parametry a vstupní informace pro ovládání motoru přes funkční blok řízení měniče frekvence (`FB_FM_Vice`).

```
RS_Moving_Vice.IO_Mapping.In.ErrorFromFI          := RS_Fm_Vice.State.Error;
RS_Moving_Vice.IO_Mapping.In.FiReady     :        := RS_Fm_Vice.State.Ready;

FB_Moving_Vice.Enable                             := RS_Moving_Vice.Enable;
FB_Moving_Vice.ErrorStop                          := RS_Moving_Vice.ErrorStop;
FB_Moving_Vice.IO_IN                              := RS_Moving_Vice.IO_Mapping.In;
FB_Moving_Vice.Param                              := RS_Moving_Vice.Param;
FB_Moving_Vice.Setting                            := Retain_Moving_Vice.Setting;
FB_Moving_Vice(Cmd                                := RS_Moving_Vice.Cmd);

RS_Moving_Vice.State                              := FB_Moving_Vice.State;
RS_Moving_Vice.ActualControlState                 := FB_Moving_Vice.STATE;
RS_Moving_Vice.IO_Mapping.Out                     := FB_Moving_Vice.IO_OUT;
```

Následující kód přijímá instrukce od `RS_Moving_Vice.IO_Mapping.Out` a je použit k ovládání samotného měniče frekvence. Zároveň skrz parametr `RS_Moving_Vice.IO_-Mapping.Out.AnalogOutput` nastaví požadovanou rychlost pohonu a měnič frekvence se postará o odpovídající převod dané rychlosti na otáčky pro asynchronní motor. Z funkčního bloku `FB_Moving_Vice` je ovládaná rovněž i brzda na motoru, kdy před započetím jakéhokoliv pohybu je brzda odbrzděna, a po jeho skončení je zase zabrzděna (`ExternalBrake`).

```
RS_Fm_Vice.Cmd.Start                    := RS_Moving_Vice.IO_Mapping.Out.RunFi;
RS_Fm_Vice.Cmd.Stop                     := NOT (RS_Moving_Vice.IO_Mapping.Out.RunFi);
RS_Fm_Vice.Param.SetSpeed               := RS_Moving_Vice.IO_Mapping.Out.AnalogOutput;

RS_Fm_Vice.Mode.UseExternalBrakeSignal  := FALSE;
ExternalBrake                           := FB_Moving_Vice.IO_OUT.ReleaseBrake;
```

Struktura předávaných parametrů u `FB_Moving_Vice` musí být totožná se strukturou řídicího systému `RS_Moving_Vice`. Navíc jsou zde implementovány interní lokální proměnné a časové funkční bloky `TON` sloužící pro stanovení časových rámců jednotlivým příkazům, jak je ukázáno na Obr. 30.



Obr. 30 Deklarace proměnných funkčního bloku pro polohování

6.3.1.3 Ovládání měniče frekvence (FB_FM_Vice)

Funkční blok zajišťuje ovládání – spouštění měniče frekvence, jenž frekvenčně řídí motor pohonu. Motor pohonu kleštiny používá k zajištění pozice brzdu. Toto řešení je efektivnější než kdyby měl motor k zajištění pozice neustále polohovat tzv. "na místě", neboť neustále dopolohování zapříčiňuje nepřetržitý chod motoru a tedy oteplování jeho pláště a snížení efektivnosti a kultivovanosti motoru.

V *INIT* části programu nastavujeme způsob řízení měniče frekvence (jak bylo řečeno výše, mezi frekvenčním a otáčkovým řízením bylo zvoleno frekvenční) a použití dílčích prvků na motoru – brzdy a větráku. Ukázka kódu z inicializační části programu s nastavováním potřebných atributů vypadá následovně:

```
RS_Fm_Vice.Mode.Frequency           := TRUE;
RS_Fm_Vice.Mode.Rpm                 := FALSE;
RS_Fm_Vice.Mode.UseBrake            := TRUE;
RS_Fm_Vice.Mode.UseFan              := TRUE;

Retain_Fm_Vice.Setting.TimeFanTOF   := 180000;
```

V předchozí kapitole byla ukázána těsná provázanost a kooperace funkčního bloku polohování (FB_Moving_Vice) s řídicím systémem měniče frekvence (RS_FM_Vice). Úkolem tohoto bloku je tedy měnič frekvence zinicializovat a uvést jej do provozu. V *CYLIC* části programu se všechny parametry předají přes interface danému FB. Ten je zpracuje, přestaví své logické výstupy a tyto výstupy jsou přivedeny do ovládací struktury měniče frekvence, který řídí frekvenci motoru, a tím rychlost pohybu kleštiny po válečkové trati.

```
FB_Fm_Vice.Enable                    := RS_Fm_Vice.Enable;
FB_Fm_Vice.ErrorStop                 := RS_Fm_Vice.ErrorStop;
FB_Fm_Vice.Mode                      := RS_Fm_Vice.Mode;
FB_Fm_Vice.Param                     := RS_Fm_Vice.Param;
FB_Fm_Vice.Setting                   := Retain_Fm_Vice.Setting;

FB_Fm_Vice(Cmd                       := RS_Fm_Vice.Cmd);
RS_Fm_Vice.State                     := FB_Fm_Vice.State;
```

Struktura předávaných parametrů u FB_FM_Vice musí být totožná se strukturou řídicího systému RS_FM_Vice. Navíc jsou zde implementovány interní lokální proměnné a časový funkční blok TOF[27] sloužící pro sepnutí větráku na hřídeli motoru, který jej ochlazuje během chodu. Struktura FB je zobrazena na Obr. 31.

FI_Ctrl			
IO_IN	FiCtrl_IO_IN	☐	VAR_INPUT
Enable	BOOL	☐	VAR_INPUT
ErrorStop	BOOL	☐	VAR_INPUT
Mode	FiCtrlMode	☐	VAR_INPUT
Setting	FiCtrlSetting	☐	VAR_INPUT
Param	FiCtrlParam	☐	VAR_INPUT
State	FiCtrlState	☐	VAR_OUTPUT
IO_OUT	FiCtrl_IO_OUT	☐	VAR_OUTPUT
Cmd	FiCtrlCmd	☐	VAR_IN_OUT
FB_TOF_Fan	TOF	☐	VAR
FB_R_TRIG_Pause	R_TRIG	☐	VAR
IsPause	BOOL	☐	VAR

Obr. 31 Deklarace proměnných funkčního bloku pro řízení měniče frekvence

6.3.2 Programová část

Programová část zajišťuje řízení vytvořených funkčních bloků v širších souvislostech. Nemá na starost jeden konkrétní subproces, ale řídí určitý celek, který obsahuje sled příkazů pro podřízené funkční bloky, jejich reakci v rámci zpětné vazby a na sebe navazujících dalších událostí na tyto stavy.

6.3.2.1 Manuální pohyby

Program pro obsluhu manuálních pohybu z ovládacího panelu *Power Panel 500*. Manuálně lze ovládat otevírání/upínání čelistí upínače a pohyb kleštiny po trati

[27] Výstup z bloku se resetuje po náběžné hraně vstupního signálu a uplynutí časové prodlevy

(odlišná tlačítka pro pohyb rychle a pomale). Využívají se zde náběžné a sestupné hrany vstupního signálu, kdy se na náběžnou hranu daný pohyb aktivuje, a sestupná hrana funguje jako STOP a pohyb zastaví, jak je ukázáno na příkladu níže:

```
FB_Button_ViceForward    (In := INT_ITF_Feeder.Manual.Button.Forward);
IF(FB_Button_ViceForward.Out_R)THEN          //rising edge
     RS_Moving_Vice.Param.Position  := Retain_Moving_Vice.Setting.Position.NegativeSwEnd;
     RS_Moving_Vice.Param.Speed          := Retain_Moving_Vice.Setting.Speed.Nominal;
     RS_Moving_Vice.Param.Service        := TRUE;
     RS_Moving_Vice.Cmd.MoveToPosition   := TRUE;
ELSIF(FB_Button_ViceForward.Out_F) THEN       //falling edge
     RS_Moving_Vice.Cmd.Stop             := TRUE;
     RS_Moving_Vice.Param.Service        := FALSE;
END_IF
```

Jakmile obsluhující pracovník klikne na tlačítko na obrazovce pro pohyb kleštiny dopředu (`INT_ITF_Feeder.Manual.Button.Forward`), spustí se tomu odpovídající příkaz `Cmd` pro funkční blok a kleština se začne pohybovat směrem dopředu do té doby, než pracovník tlačítko neuvolní, případně než je dosažena koncová pozice vpředu (`NegativeSwEnd`).

6.3.2.2 Referování

Zreferovaná kleština tvoří základní podmínku pro jakýkoliv další pohyb. Bez zreferované kleštiny je libovolný příkaz pro pohyb kleštiny (ať už v manuálním režimu nebo v automatickém cyklu) vyhodnocen chybou.

Reference probíhá kliknutím na tlačítko z obrazovky panelu a úspěšné provedení je signalizována stavovou ikonou rovněž na panelu. Kód reference a posloupnost událostí je naznačena níže:

```
CASE STATE_TMP OF
     0:
          IF RS_Referencing.Cmd.Start THEN
               RS_Referencing.Cmd.Start            := FALSE;
               RS_Moving_Vice.Param.Service        := FALSE;
               RS_Clamping_Vice.Cmd.Open           := TRUE;
               STATE_TMP                           := 1;
          END_IF
     1:   // Kontrola otevreni klestiny
          IF Stepping((NOT RS_Clamping_Vice.Cmd.Open AND RS_Clamping_Vice.State.Open),
               Krokovani_Enabled,Next_state ) THEN
               STATE_TMP                           := 2;
          END_IF
     2:   //Pojezd s klestinou do homovaci polohy
          RS_Moving_Vice.Cmd.Homing                := TRUE;
          STATE_TMP                                := 3;
```

```
        3:
            IF Stepping((NOT RS_Moving_Vice.Cmd.Homing AND RS_Moving_Vice.State.HomingOK),
                Krokovani_Enabled,Next_state ) THEN
                STATE_TMP                       := 4;
            END_IF
        4:
            ITF_Feeder.State.Homing_OK          := TRUE;
            ITF_Feeder.State.Wait               := TRUE;
            ITF_Feeder.State.InMove             := FALSE;
            STATE                               := REFERENCE_WAIT;
            STATE_TMP                           := 0;
END_CASE;
```

6.3.2.3 Řídicí program

Nejpoužívanější a nejdůležitější blok v celém modulu ovládání kleštiny. Na základě požadavku z nadřazeného řízení vyhodnocuje jednotlivé příkazy a dává úkoly podřízeným funkčním blokům. Co se týče samotné logiky aplikace, tak nad tímto modulem se nachází už pouze řídicí framework celého řezacího centra, který koordinuje jednotlivé operace ze všech modulů, rozděluje úkoly a stará se o automatický chod stroje. Řídicí program je tedy jediný komunikační prostředek mezi nadřazeným framework řízením a ovládaným modulem kleštiny.

Netechnicky řečeno funguje řídicí program jako takový prostředník. Dostane příkaz z nadřazeného řízení, a ten dále rozvětví na sled operací pro dílčí funkční bloky. Jakmile jeden funkční blok dokončí svoji operaci, uvědomí o tom řídicí program, který zavolá v pořadí další FB. Jakmile jsou vykonány všechny operace patřící do daného příkazu, informuje řídicí program o vykonání žádaného příkazu nadřízený framework. Podle navrhovaného řešení byly vytvořeny následující příkazy pro framework:

- ITF_Feeder.Cmd.Homing

  Příkaz zajišťující referenci kleštiny. V případě změny parametrů souvisejících s referencí – SW limitní pozice či referenční pozice, se musí reference vykonat znovu. Jakýkoliv pohyb s nezreferovanou kleštinou vyvolá chybové hlášení.
- ITF_Feeder.Cmd.FindMat_E

  O hledání konce materiálu se stará tento příkaz. Při jeho aktivaci se otevřou upínací čelisti, poté se kleština rozjede nominální rychlostí po trati směrem k vrtacímu centru. Jakmile je pomocí indukčnostního spínače umístěného v čelistech upínače detekován materiál, kleština se zastaví. Nicméně materiál vlivem setrvačné síly ujede ještě nějakou vzdálenost. Poté co se kleština po detekování materiálu zastaví, rozjede se znovu, ovšem minimální rychlostí (defaultně nastaveno na 10 mm/s). Po pomalém dojetí k materiálu a jeho opětovné detekci přítomností v čelistech upínače se kleština zastaví a pneumaticky ovládanými čelistmi materiál upne.

Může ovšem nastat i situace, že žádný materiál nebude na trati fyzicky přítomen. I na tuto situaci by měla být kleština připravena a schopna reagovat.

V případě, že kleština při hledání konce dojede až na laserovou závoru a během tohoto pohybu nezaznamená jakoukoliv přítomnost materiálu v čelistech, tak se zde zastaví a jede zpět na koncovou pozici vzadu. Po dokončení pohybu informuje nadřazený framework chybovým hlášením o nepřítomnosti materiálu na válečkové trati.

- ITF_Feeder.Cmd.FindMat_S

  Poté, co je nalezen konec materiálu, je potřeba nalézt jeho začátek, aby kleština věděla, jak dlouhý materiál má. Je to pro případ, že by obsluha stroje chtěla např. řezat 15 kusů po 1000 mm, ale reálně by na trati byl materiál pouze o délce 8000 mm. Nalezení začátku materiálu spočívá v pohybu kleštiny s upnutým materiálem směrem k laserové závoře. Po jejím protnutí se kleština zastaví, a pomalou rychlostí směrem vzad se vrátí na nejzazší možnou polohu tak, aby nestínila laserovou závoru. Poté je operace dokončena a kleština zinicializovaná.

- ITF_Feeder.Cmd.Load

  Příkaz navezení materiálu ve své podstatě jen spojí dva předchozí příkazy do jednoho. Jedná se tedy o kompletní přípravu materiálu k navezení. Kleština nejprve otevře čelisti, nalezne konec materiálu, upne jej a poté nalezne i začátek materiálu. Tímto je kleština zinicializována.

- ITF_Feeder.Cmd.MoveToPosition

  Nejpoužívanější příkaz po zinicializování kleštiny. Framework podle zvolené délky kusu z receptury sdělí kleštině na jakou pozici má najet, v případě kolize na jakou pozici se vrátit. Příkaz je implementován následovně:

```
MOVE_TO_POS: //MOVE TO REQUIRED POSITION
     IF (ITF_Feeder.State.MatFound_S)  THEN
               CASE RS_Loading.STATE_TMP OF
               0:
               RS_Moving_Vice.Param.Position   := ITF_Feeder.Param.Req_position;
               RS_Moving_Vice.Cmd.MoveToPosition:= TRUE;
               RS_Loading.STATE_TMP            := 1;
               1:
               IF Stepping((NOT RS_Moving_Vice.Cmd.MoveToPosition AND
                RS_Moving_Vice.State.InPosition),Krokovani_Enabled,Next_state ) THEN
                        RS_Loading.STATE_TMP := 2;
               END_IF;
               2:
               RS_Loading.STATE_TMP            := 0;
               RS_Loading.STATE                := WAITING;
               ITF_Feeder.State.InPosition     := TRUE;
               END_CASE;
     ELSE      //nebyl nalezen material
               RS_Loading.STATE                := ERROR;
               RS_Loading.STATE_TMP            := 0;
               RS_Loading.Error                := TRUE;
               RS_Loading.ErrorID              := 200;
     END_IF;
```

- ITF_Feeder.Cmd.ManualEnable

  Manuální pohyby nelze ovládat kdykoliv, např. při započetí automatického cyklu, je naprosto nepřípustné jakkoliv zasahovat frameworku do řízení kleštiny. Je zakázané, aby obsluha během vrtání do materiálu otevřela svěrák, či dokonce nějak polohovala s kleštinou. Takovýmto zásahem do automatického cyklu hrozí zničení nástroje, či nebezpečí úrazu. Proto je potřeba manuální pohyby povolovat jen pokud se obsluha nachází na obrazovce *Manualni pohyby* nebo *Servisni parametry*.

  V případě, že je kleština zinicializovaná a obsluha přepne na jednu z výše zmíněných obrazovek, a polohuje s kleštinou, tak se inicializace vynuluje a před započetím automatického cyklu je třeba znovu nalézt konec a začátek materiálu.

- ITF_Feeder.Cmd.ReleaseMat

  Příkaz pro otevření čelistí a uvolnění materiálu.

6.3.3 Chybová hlášení

Součástí řídicího programu je i hlídání chybových hlášení. V případě, že by během jakéhokoliv příkazu nastala chyba, ať už vlivem podřízeného FB nebo v řídicím programu, tak se vždy aktuálně vykonávaný příkaz okamžitě zastaví (všechny podřízené součásti řízení a FB dostanou příkaz `Cmd.Stop`) a na panelu se objeví hlášení s danou chybou. Po potvrzení – „vykvitování“ chyby obsluhou se všechny součásti resetují (`Cmd.Reset`) a chyby vynulují.

6.3.4 Kolizní stavy

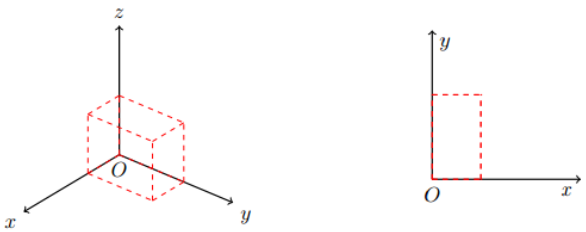
Vzhledem ke skutečnosti, že pila je natáčecí – v rozsahu ±60°, a dovoluje tedy řezat i úhlové tvary, je potřeba brát v potaz možné kolizní stavy, které mohou nastat. Zejména jde o kolizní stavy mezi kleštinou a ramenem pily natočeným v určitém úhlu. Tuto problematiku nelze zobecnit a vytvořit jednotný rámec (například v podobě úhlového limitu max. natočení ramene), poněvadž vznik kolize závisí na několika na sobě nezávislých faktorech: šířce a výšce materiálu, navážené délce a aktuálním natočení ramene pily.

Důležitá je rovněž predikce takovéto kolize, kleština musí vědět, že požadovanou délku nemůže navést ještě před započetím pohybu. V této kapitole bude stručně vysvětleno řešení kolizí.

6.3.4.1 Souřadný systém a orientace v rovině

K popisu tělesa v prostoru je zapotřebí využít souřadný systém. Polohu jakéhokoliv bodu můžeme potom určit odečtením jeho souřadnic z jednotlivých os.

Druhů souřadných systémů se využívá několik. Z nejznámějších můžeme například uvést kartézský, polární či válcový systém souřadnic. Na Obr. 32 je znázorněn kartézský systém souřadnic v prostoru a v rovině. Pro účely detekce kolizí u kleštiny bude využit kartézský souřadnicový systém v rovině.

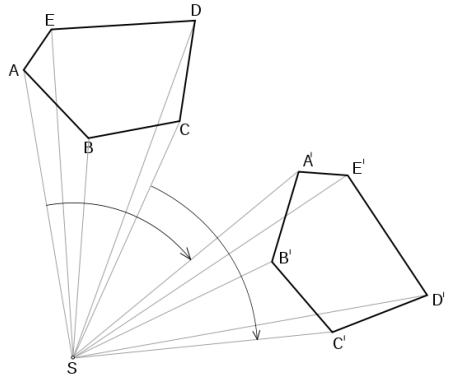


Obr. 32 Kartézské souřadnice v prostoru a rovině [zdroj: Rouš, 2012]

V rovině má těleso dva stupně volnosti – souřadnice *x*, *y*. Vzájemnou polohu souřadných systému dvou těles můžeme vyjádřit pomocí několika způsobu, které jsou mezi sebou převeditelné [Rouš, 2012]:

- rotační matice R,
- Eulerových úhlů – tři úhly definující postupné otáčení kolem jednotlivých os s důrazem na pořadí (konvence),
- vektoru a úhlu – každou rotaci z původního do nového stavu vyjádříme jako pootočení okolo vektoru o úhel

Nejlepší varianta pro naše podmínky se jeví rotační matice, která je nástrojem prostorové kinematiky. V geometrii představuje rotace v eukleidovské rovině geometrické zobrazení, které je charakterizováno tím, že tvar a velikost jednotlivých geometrických útvarů se při rotaci nemění, jak je znázorněno na Obr. 33. Při otočení se rovněž nemění dimenze rotovaného geometrického tělesa.



Obr. 33 Geometrické otočení [zdroj: http://goo.gl/psqueM]

Rotace v dvourozměrné Eukleidově rovině kolem počátku souřadnic o úhel $\boldsymbol{\alpha}$ je dán vztahem (1.6.) a (1.7.), využívajícím trigonometrické funkce, kde parametry $\boldsymbol{x}$ a $\boldsymbol{y}$ jsou souřadnice původního bodu, a parametry $\boldsymbol{x}'$ a $\boldsymbol{y}'$ souřadnice otočeného bodu.

$$x' = x\cos\alpha - y\sin\alpha \tag{1.6.}$$

$$y' = x\sin\alpha - y\cos\alpha \tag{1.7.}$$

6.3.4.2 Implementace (FB_CalcColission)

Nejprve je nutné identifikovat obě tělesa v rovině vůči nulovému bodu. K tomu nám poslouží výkresová dokumentace obou strojů (naplnění parametrů `L1x, L2x, L1y, L2y, L3y` a `A0, A1, A2, A3, M0, M1, M2, M3`). Vytvořením dvou FB (`FB_ViceColissionPosition` a `FB_SawColissionPosition`) pro výpočet souřadnicových systémů, je prvním krokem k detekci kolizních stavů. Předání parametrů na straně kleštiny vypadá následovně:

```
FB_ViceColissionPosition(
    Aktualni_Poloha     := (RS_FeederMoving.State.Position - INT_ITF_Feeder.Coli
                            ssionArea.offset),
    L1x                 := INT_ITF_Feeder.ColissionArea.L1x, //100
    L2x                 := INT_ITF_Feeder.ColissionArea.L2x, //150
    L1y                 := INT_ITF_Feeder.ColissionArea.L1y, //50
    L2y                 := INT_ITF_Feeder.ColissionArea.L2y, //80
    L3y                 := INT_ITF_Feeder.ColissionArea.L3y, //60
    offset              := INT_ITF_Feeder.ColissionArea.offset //offset
                        );
```

Rameno je statické a nikam se neposouvá, pouze se natáčí, odpadá zde tedy parametr `Aktualni_Poloha` a naopak je zde vcelku zásadní parametr `angle`, který určuje natočení ramene pily. Předání parametrů na straně pily vypadá následovně:

```
FB_SawColissionPosition(
    A0                  := INT_ITF_Feeder.ColissionArea.A0, //10
    A1                  := INT_ITF_Feeder.ColissionArea.A1, //10
    A2                  := INT_ITF_Feeder.ColissionArea.A2, //10
    A3                  := INT_ITF_Feeder.ColissionArea.A3, //50
    M0                  := INT_ITF_Feeder.ColissionArea.M0, //10
    M1                  := INT_ITF_Feeder.ColissionArea.M1, //30
    M2                  := INT_ITF_Feeder.ColissionArea.M2, //30
    M3                  := INT_ITF_Feeder.ColissionArea.M3, //10
    angle               := INT_ITF_Feeder.ColissionArea.uhel
                        );
```

Pokud je úhel natočení nenulový, výstupní struktura v nulovém natočení se přetransformuje dle výše popsaného vztahu rotace, aby nově vzniklý souřadnicový systém dané natočení ramene reflektoval. Toto zajišťuje následující kód:

```
IF (angle <> 0) THEN
    i := 1;
    angle_rad := (angle * 3.141592654) / 180.0;
    cos_alfa := COS(angle_rad);
    sin_alfa := SIN(angle_rad);
```

```
    WHILE i<10 DO
        tmp_x                       := ColissionArea.Pos[i].x;
        tmp_y                       := ColissionArea.Pos[i].y;
        ActualCollisionArea.Pos[i].x  := tmp_x * cos_alfa - tmp_y * sin_alfa;
        ActualCollisionArea.Pos[i].y  := tmp_x * sin_alfa + tmp_y * cos_alfa;
        i := i+1;
    END_WHILE
ELSE
    ActualCollisionArea             := ColissionArea;
END_IF
```

Dalším krokem je vytvoření FB, který bude souřadnicové systémy pomocí matematického modelu vyhodnocovat, zda nedochází k protnutí, či překrytí dvou úseček z různých souřadnicových systémů. Matematické modely kleštiny a pily máme již definovány – jedná se vždy o devíti prvkové pole, které jednoznačně popisují kleštinu/pilu v rovinně vůči nulovému bodu, kde každý prvek je definován svými souřadnicemi `x` a `y`. Předchozí dva FB, pro počítání souřadnic kleštiny a pily, slouží jako vstup pro tento hlavní FB, jenž má na starost právě vyhodnocování kolizí kleštiny a pily vůči sobě. Předání parametrů vypadá následovně:

```
FB_CalcColission (
    ViceColission   := FB_ViceColissionPosition.ColissionArea,
    SawColission    := FB_SawColissionPosition.ActualCollisionArea,
    Enable          := INT_ITF_Feeder.ColissionArea.EnableCyclic,
    SafetyHight     := INT_ITF_Feeder.ColissionArea.SafetyHight
                    );
```

První úsečka kleštiny je definovaná bodem `p1` a bodem q_1, tyto body dostanou ve funkčním bloku patřičné souřadnicové ohodnocení

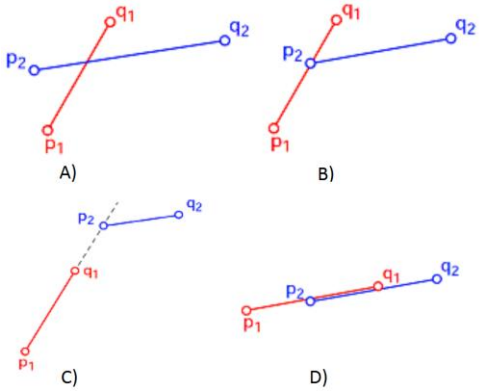
```
p1.x            := FB_ViceColissionPosition.ColissionArea.Pos[1].x
p1.y            := FB_ViceColissionPosition.ColissionArea.Pos[1].y
q1.x            := FB_ViceColissionPosition.ColissionArea.Pos[2].x
q1.y            := FB_ViceColissionPosition.ColissionArea.Pos[2].y
```

Stejné ohodnocení proběhne i pro úsečku pily:

```
p2.x            := FB_SawColissionPosition.ColissionArea.Pos[1].x
p2.y            := FB_SawColissionPosition.ColissionArea.Pos[1].y
q2.x            := FB_SawColissionPosition.ColissionArea.Pos[2].x
q2.y            := FB_SawColissionPosition.ColissionArea.Pos[2].y
```

A poté FB vyhodnotí, zda se tyto 2 úsečky protnou či nikoliv. Stejný postup je proveden pro všechny úsečky kleštiny vůči všem úsečkám pily. Z výsledku testování tohoto FB v prostředí *PLCSim,* vyšlo najevo, že existuje pár situací

s kterými si FB nedokáže poradit správně a musel být mírně pozměněn vyhodnocovací algoritmus (na základě případů B) a D) dle Obr. 34).



Obr. 34 Příklady možných stavů při vyhodnocování průsečíku úseček dvou objektů – v případě A) kolize nastane, v ostatních případech nikoliv

6.4 Implementace – Vizualizační blok

Řezací centrum bude ovládané z operátorského panelu Power Panel 500 (dále jen panel). Tento panel je vybaven dotykovou 10.4" TFT obrazovkou s *VGA*[28] rozlišením a rezistivní technologií dotyku.

Je vcelku zajímavé sledovat, že dnešní honba za nejvyšším rozlišením, jemností displeje, zvyšováním počtu jader CPU, atd. se naprosto vyhnula segmentu průmyslové automatizace. Jinak si nelze vysvětlit skutečnost, jak v zařízeních za stovky EUR, mohou být komponenty z dob typické pro konec minulého století. Vysvětlení je vcelku prosté. Např. v případě použití rezistivního displeje namísto dnes používanějšího kapacitního je to zejména vyšší životnost rezistivní technologie oproti kapacitní (5 let oproti 2 rokům), rovněž rezistivní displej reaguje na dotyk čímkoliv (obsluha v rukavicích), naproti tomu kapacitní displej reaguje pouze na vodivý dotyk.

Taktéž komponenty v průmyslové automatizaci musí být uzpůsobeny a certifikovány pro použití v takovýchto odvětvích (stupeň krytí, odolnost vůči elektromagnetickému rušení, aj.) a standardizace těchto technologií trvá roky.

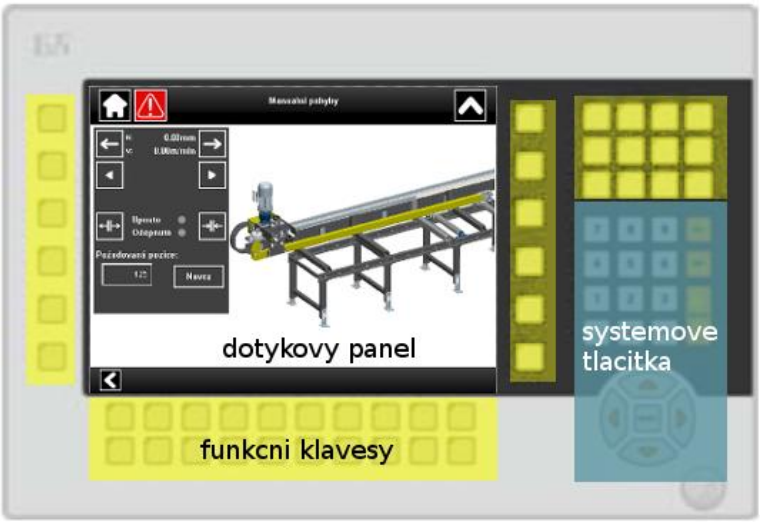
6.4.1 Tvorba grafických komponent

Pro ovládání kleštiny z panelu je nutné vytvořit jednotlivé ovládací a informační prvky jako např. tlačítka, textová pole, ikony stavů akčních členů, ukazatele, signálky, dvoustavové ikony typu ON/OFF, aj. (viz Obr. 35). Některé prvky je možné převzít z knihovny B&R k tomu určené, jiné musejí být vytvořeny ručně v grafickém editoru.

Vkládání hodnot do číselných parametrů je umožněno dvěma způsoby. Pomocí systémových tlačítek nacházejících se vedle dotykového panelu (Obr. 36).

[28] 640 × 480 obrazových bodů

Obr. 35 Ukázka tlačítek a stavových ikon využitých na stránce *Str_Manual_klestina*



Obr. 36 Dotykový panel s rozložením funkčních a systémových kláves

Případně přímo z dotykového panelu, kdy po kliknutí na požadovanou veličinu se otevře numerická klávesnice pro zadání hodnoty (viz Obr. 37). Potvrzení proběhne kliknutím na *Enter* (↵). Poté je číselná hodnota uložena do proměnné v programu, na kterou je z vizualizace namapována.

Obr. 37 Numerická klávesnice pro zadávání hodnot

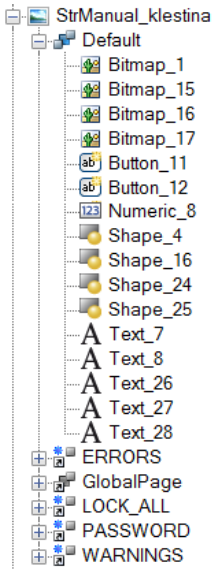
Po obvodu a pod dotykovým panelem se nachází 44 funkčních kláves, které je možno naprogramovat dle libosti, a použít je pro nejpoužívanější operace. Obsluha stroje bude např. na stránce *Str_Servis_klestina* konfigurovat parametry pro rychlost kleštiny a bude si chtít nastavené parametry ověřit na reálném pohybu kleštiny. Nebude tedy nutné vracet se zpět a otevřít stránku *Str_Manual_klestina* pro daný pohyb, ale využije se právě funkční klávesa, což zefektivní a zrychlí práci.

6.4.2 Stránky vizualizace

Po vytvoření nutných grafických komponent pro ovládání a monitoring, je zhotoveno několik stránek, které obsluha potřebuje k řízení kleštiny z operátorského panelu. Důraz je kladen zejména na jednoduché a intuitivní ovládání. Rovněž je žádoucí, aby na každé stránce byly zobrazeny jen důležité informace a zamezilo se zahlcení obsluhy nepotřebnými daty.

Na Obr. 38 je ukázána struktura vrstev stránek, kdy v každé vrstvě budou využity jiné prvky podle konkrétního použití stránky. Globální prky, jako tlačítko *Domů*, tlačítko *Error* a tlačítko *Zpět* jsou pro všechny stránky totožné, a je vhodné zachovat jejich identické rozložení napříč všemi stránkami. Tyto prvky budou obsaženy ve vrstvě *GlobalPage*, a při vytváření nové stránky budou tyto prvky již její součástí.

Ostatní prvky pro ovládání a informování jsou obsaženy ve vrstvě *Default*. Vrstva *ERRORS* obsahuje výpis chybových hlášení a v případě výskytu chyby se tento výpis aktivuje a vrstva dostane nejvyšší prioritu a zobrazí nad všemi ostatními vrstvami na obrazovce. Ostatní vrstvy (*PASSWORD,LOCK_ALL* a *WARNINGS*) jsou pouze vytvořeny, ale nejsou využity, počítá se s nimi jako možné rozšíření do budoucna – zamykání jednotlivých obrazovek podle přístupové příslušnosti obsluhy. Jiné pravomoci bude mít servisní technik při servisní prohlídce než obsluha stroje při každodenní práci se strojem.



Obr. 38 Stromová struktura rozložení stránky ve vrstvách s vnořenými grafickými prvky

Následuje seznam všech vytvořených stránek vizualizace a popis jejich využití:

- Str_Geometrie_klestina

  Stránka *Geometrie klestiny* slouží pro definování vzdáleností pozic kleštiny vepředu a vzadu od nulového bodu, obsah daných proměnných se použije při referenci. Jedná se o limitní koncové softwarové pozice, které vytyčují tzv. pracovní pásmo při polohování kleštiny.

- Str_Homing_klestina

  Stránka *Reference klestiny,* jak již sám název napovídá, má za úkol na pokyn obsluhy začít s referencí a informovat o jejím aktuálním průběhu. Úspěšně provedená reference je zobrazena zelenou signalizační kontrolkou.

  Na stránce reference se také nacházejí tlačítka pro hledání materiálu a případně pro pozastavení hledání materiálu. Pod pojmem hledání materiálu se rozumí hledání konce a začátku materiálu. Bylo by zbytečné dělit tento proces na dvě části, i když programově jsou to dva rozdílné bloky, a pokud by z nějakého důvodu chtěla obsluha hledání konce a začátku materiálu řídit zvlášť, není problém tento proces i ve vizualizaci rozdělit na dva procesy.

- Str_Manual_klestina

  Stránka *Manualni pohyby* zajišťuje veškerý pohyb kleštiny v manuálním režimu, tedy pohyb dopředu/dozadu a to jak nominální rychlostí, tak i pomale (hodnota rychlosti „pomale“ se musí předem nastavit na stránce *Servisni parametry*). Dále pak ovládá čelisti upínače (otevřít/upnout) a v neposlední řadě pohyb kleštiny na požadovanou pozici. Upnuté nebo otevřené čelisti jsou signalizovány zelenou kontrolkou.

- Str_Servis_klestina

  Na stránce *Servisni parametry* se nastavují všechny nutné parametry pro pohyb kleštiny. Jedná se o rychlost, akceleraci/deceleraci, parametry pro PID regulátor, timeouty, veličiny pro odměřování polohy, aj. Většina parametrů je také ošetřena pro případ pokusu o vložení nesprávné hodnoty – např. do buňky rychlosti nelze vložit rychlost menší než 0 mm/s nebo do buňky akcelerace nelze vložit hodnota větší než 10 000 mm/s^2, apod.

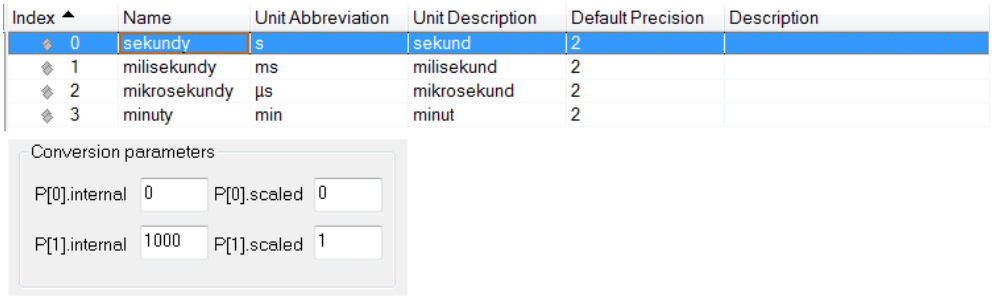
- Str_FM_klestina

  Stránka *FM klestiny* obsluhuje měnič frekvence kleštiny. Jedná se o nezávislé řízení měniče, a obsluhující pracovník stroje přistupuje přímo do řídicí struktury funkčního bloku měniče. Je tedy potřeba dbát zvýšené obezřetnosti, protože měnič frekvence nemá ponětí o nějaké koncové pozici. Jakmile dostane povel START, uvede motor kleštiny do chodu nadefinovanou rychlost. Pohybl kleštiny trvá do té doby, než ho obsluha tlačítkem STOP nezastaví. Tato stránka slouží spíše jako servisní, v případě, že by kleština v automatickém cyklu najela do nějakého zakázaného pásma, tak pouze na této stránce je možno měnič frekvence ovládat nezávisle na jakémkoliv programu. Obsluha stroje je na této stránce rovněž informována o aktuálním proudu a krouticím momentu dodávaném měničem na řízený asynchronní motor.

6.4.3 Jmenovité/Interní jednotky

Programovatelný automat pracuje uvnitř své řídicí struktury s interními jednotkami, kde u času jsou to milisekundy, u rychlosti otáčení motoru jsou to *unity/sec*[29]. Naopak obsluha stroje, při nastavování číselných hodnot parametrů pracuje se jmenovitými jednotkami tzn. časové parametry definuje v sekundách, rychlosti v mm/min (m/min.), atd. Je tedy nutné, aby proběhla konverze jednotek mezi těmi interními a jmenovitými.

Proto byl v bloku *Unit text* ve vizualizaci vytvořen seznam jednotek s jednotlivými převodními poměry mezi jmenovitými a interními veličinami, který poskytne relevantní údaje o totožných jednotkách jak obsluze stroje, tak řídicímu programu v aplikaci.



Obr. 39 Převodní tabulka časových jednotek

6.4.4 Závěr kapitoly

Cílem této kapitoly byla implementace stránek vizualizace a tvorba dílčích grafických komponent nutných pro ovládání a monitoring stavů kleštiny na dotykovém panelu. Jednotlivá tlačítka ve vizualizaci jsou namapována na příkazy v řídicím programu, takže obsluha jejich stiskem spustí odpovídající příkaz. Stavové ikony jsou namapovány na statusy funkčních bloků a informují tak obsluhu o současném stavu akčních členů. Aktuální rychlost, zrychlení, pozice kleštiny je pomocí převodníku jednotek zobrazena obsluze ve jmenovitých jednotkách.

Rovněž bylo v této kapitole vysvětleno využití jednotlivých stránek vizualizace pro obsluhu stroje.

[29] Interní jednotka měniče frekvence pro řízení motoru, kde 1000 unitů je jedna otáčka motoru

7 Závěr

7.1 Shrnutí práce

Na základě návrhu řešení byla implementována aplikace systému řízení pro modul řezacího centra.

Analýza technologií řezacího centra včetně dílčích podsystémů se nachází v druhé kapitole. Tato část obnášela intenzivní teoretické studium k porozumění principu funkčnosti a použití všech využitých komponent. Značné úsilí musel autor rovněž věnovat seznámení se s ovládáním měniče frekvence a jeho řízením pomocí interních funkčních bloků a systémových knihoven.

Komunikaci mezi měničem frekvence, ale i ostatními prvky ovládaného ostrůvku a řídicím dotykovým panelem zajišťuje real-timový protokol Powerlink Ethernet. Ten je typickým představitelem průmyslového Ethernetu, který nevyžaduje žádný speciální hardware, využívá všechny existující standardy a lze pro něj použít všechny čipy, zařízení a testovací systémy vytvořené pro standardní Ethernet. Možnostem komunikace mezi jednotlivými moduly se věnuje třetí kapitola.

Vzhledem ke skutečnosti, že dodavatel hardwaru pro řízení kleštiny byl předem dán, tak i veškeré programové řízení v podobě softwaru muselo být od stejného dodavatele. Aplikace byla tedy vytvořena ve vývojovém prostředí Automation Studio od společnosti B&R. Jedná se o vývojový nástroj, který provází uživatele ve všech fázích tvorby projektu, tj. konfigurace, programování, diagnostika, vizualizace, aj. Ucelený přehled o v dnešní době nejpoužívanějších vývojových prostředcích, je obsažen v čtvrté kapitole.

V kapitole Metodika autor popisuje jednotlivé kroky k naplnění cíle práce, tj. jak bylo postupováno při implementaci řídicí aplikace.

Hlavní zaměření práce – implementace aplikace systému řízení – pokrývá většinu praktické části. Aplikace je tvořena 2 celky a to řídicím programem a vizualizací. Obě dvě částí jsou silně provázány a vizualizace byla tvořena současně s přibývajícím počtem naimplementovaných subprocesů, a nikoliv až po dokončení kompletního řídicího programu. Implementace řídicího programu spočívala nejprve v přípravě funkčních bloků, ovládající jednotlivé subprocesy (řízení pneumatických válců, inicializaci a spouštění měniče frekvence, polohování s asynchronním motorem s využitím PID regulace). Poté jsou funkční bloky pomocí rozhraní předány do řídicí struktury programového řízení. Programové řízení je posloupnost příkazů a reakcí na tyto příkazy, které přistupují do struktur funkčních bloků, aktivují je a tím ovládají konkrétní subproces. Po vykonání daného subprocesu, funkční blok pomocí stejné struktury informuje nadřazené programové řízení o splnění příkazu, a nadřazené řízení na to reaguje patřičným způsobem.

Úkolem druhého celku – vizualizace – je poskytovat obsluhujícímu operátorovi přehled o stavu řízeného modulu a pomocí ovládacích prvků mu umožnit řídit chod a zasahovat do činnosti řezacího centra. Pro tyto požadavky bylo vytvořeno několik obrazovek pro ovládání z dotykového panelu, jejichž posláním je na pokyn obsluhy vykonat příslušný příkaz a rovněž poskytovat relevantní, spolehlivé a aktuální informace o řízeném modulu.

7.2 Zhodnocení výsledného řešení

Cílem práce bylo navrhnout a implementovat řídicí systém modulu pro ovládání řezacího centra. Autor se domnívá, že práce úspěšně splnila všechny požadavky zadání a závěry mohou sloužit i dalším autorům, studentům nebo zájemcům o oblast průmyslové automatizace. Cíl práce se tedy podařilo naplnit.

Aplikaci systému řízení lze po zkompilování nahrát do reálného stroje a bez problému spustit a používat k ovládání daného modulu, tj. kleštiny v testovacím režimu, to znamená s připojeným počítačem.

Zatím se však nejedná o aplikaci schopnou produkčního nasazení, a to zejména z toho důvodu, že nejsou dopracována všechna textová pole chyb ve vizualizaci, které by obsluhu v případě chyby informovali o původci chybového hlášení. To má za následek, že pokud by nyní takováto chyba nastala, tak by se zobrazilo chybové hlášení, ale nebylo by co potvrzovat (obrazovka s popisem chyby by byla prázdná) a obsluha by byla nucena stroj vypnout a znovu zapnout.

7.3 Možné rozšíření

Pokud se zaměříme na možné rozšíření aplikace do budoucna, jednalo by se zejména o implementaci procedury řešící uživatelskou autorizaci a s tím související možnost měnit určité parametry, jen po příslušné autorizaci uživatele. Procedura by řešila přístup k jednotlivým parametrům na základě priorit.

V produkčním nasazení jde o to, že daný stroj bude ovládán proškoleným pracovníkem. Tento pracovník může nastavovat a upravovat běžné parametry jako rychlost kleštiny, akcelerace/decelerace kleštiny při rozjezdu, časové konstanty pro vypršení pohybu, atp. avšak některé parametry jako volba *Task class* pro řízení měniče frekvence, či hodnoty konstant ovlivňující PID regulátor by měly být pro běžnou obsluhu stroje skryty, či „zamčeny“. Stejně tak pozice pro referování a koncové softwarové koncové pozice vzadu/vepředu se nastaví jen jednou, na začátku při uvedení stroje do provozu a poté již není nutno je jakkoliv měnit.

K těmto parametrům by tedy běžná obsluha mít přístup neměla, na druhou stranu je ale z vizualizace také úplně odstranit nelze, protože za jistých okolností – zejména servisního charakteru, je potřeba tyto údaje upravit.

Jako nejvhodnější možnost byla navržena a částečně implementována varianta s autorizací uživatele pomocí hesla (viz Obr. 40), kde v základním zobrazení (pro běžnou obsluhu), by byla přiřazena výchozí priorita 1, ve které je možné měnit běžné parametry (rychlost, akcelerace, timeout, …) a parametry

s vyšší prioritou pro servisního technika, zajišťující pravidelný servis stroje, by byly „odemčeny“ jen po zadání hesla pro prioritu 2. Po zadání příslušeného hesla by bylo možné měnit jednak všechny parametry z výchozí priority a navíc i parametry spadající pod prioritu 2.

Autorizace pro nejvyšší priority, by umožnila změnu jakéhokoliv parametru (i např. změny *Task class*, změnu citlivosti odměřování rychlosti kleštiny z hřídele motoru – parametr `Encoder Increments`, aj.). Priorita 3 by tedy umožňovala libovolný zásah do nastavování parametrů.

Dílčím rozšířením spojeným s nejvyšší prioritou by bylo odemčení přístupu do počítadla motohodin, kde by šli nejen vyčíst informace o počtu ujetých metrů kleštiny, či doba běhu stroje v hodinách ale i tyto informace vynulovat, např. po výměně motoru. V ostatních prioritních módech by bylo možné tyto hodnoty jen sledovat, bez možnosti jejich změny.

Obr. 40 Horní lišta obrazovky s ukázkou autorizace uživatele

Pro zajištění vyšší bezpečnosti, by byl naprogramován i určitý časový interval pro změnu parametrů v daném prioritním módu, kde po jeho vypršení by se vše vrátilo do základního zobrazení s výchozí prioritou 1 pro normální obsluhu stroje.

8 Literatura

ABB. Regulované pohony s měniči frekvence. Elektrické stroje a pohony [online]. 2011 [cit. 2014-10-25]. Dostupné z: http://www.propohony.cz/menice-a-regulatory/180-regulovane-pohony-s-menici-frekvence-2-dil

AUTOMA: časopis pro automatizační techniku. Praha: FCC Public, 2007, roč. 2007, 8-9. ISSN 1210-9592.

AUTOMA: časopis pro automatizační techniku. Praha: FCC Public, 2003, roč. 2003, 3. ISSN 1210-9592.

Automatisierungsgeraete S5-135 und S5-150U. Katalog 54, Siemens AG, 1985.

BÉLAI, Igor. *Komunikácia v priemyselnej automatizácii* (1-7)[online].Seriál článků. AT&P Journal, 2007, čísla 1 –10 [cit. 2014-11-16]. Dostupné z: http://www.atpjournal.sk/buxus/docs/atp-2007-04-53.pdf, http://www.atpjournal.sk/buxus/docs/atp-2007-06-64.pdf, http://www.atpjournal.sk/buxus/docs/atp-2007-10-81.pdf

BENEŠ, Pavel et al. *Automatizace a automatizační technika: prostředky automatizační techniky*. 5., rozš. a aktualiz. vyd. Brno: Computer Press, 2014, 304 s. ISBN 978-80-251-3747-5.

BERGE, Jonas. *Fieldbuses for process control: engineering, operation, and maintenance*. Research Triangle Park, NC: ISA, 2002, viii, 460 p. ISBN 15-561-7760-7.

BERNECKER & REINER. *Automation Studio 4*. Eggelsberg (Rakousko), 2013. Dostupné z: http://goo.gl/y9JLnD

BERNECKER & REINER. *Perfection in Automation* [online]. 2014 [cit. 2014-11-19]. Dostupné z: http://www.br-automation.com/cs/perfection-in-automation/.

ČECH, Martin a Miloš SCHLEGEL. Návrh PID regulátoru [online]. Plzeň, 2004[cit. 2014-10-25]. Dostupné z: http://automatizace.hw.cz/files/images/files/PIDRegions_cz.pdf

DJIEV, Stancho. *Industrial Networks for Communication and Control*. Sofie, 2003. Dostupné z: http://anp.tu-sofia.bg/djiev/PDF%20files/Industrial%20Networks.pdf. Studijní skripta. Technická univerzita v Sofii.

HRUŠKA, František. *Senzory. Fyzikální principy, úpravy signálů, praktické použití*. Zlín: Univerzita Tomáše Bati ve Zlíně, 2010. ISBN 978-80-7454-096-7.

Industrial Ethernet. *Planning and Instalation Guide for Industrial Ethernet*, roč. 3.0. IAONA e.V., Březen 2003.

KADLEC, Karel a Miloš KMÍNEK. Programovatelné logické automaty. *Měřicí a řídicí technika* [online]. Praha, 2005 [cit. 2014-11-02]. Dostupné z: http://uprt.vscht.cz/kminekm/mrt/F5/F5k53-PLC.htm

KOZELSKÝ, Michal. *Rozdělení PLC* [online]. 2010 [cit. 2012-05-02]. Dostupné z: http://lms.wmmania.cz/3-rocnik/praxe/rozdeleni-plc/

KOZIOREK, Jiří a Libor CHROMČÁK. *Logické systémy řízení a programovatelné automaty* [online]. Ostrava: Vysoká škola báňská - Technická univerzita, 2007, 1 DVD-R [cit. 2014-12-13]. ISBN 978-80-248-1490-2. Dostupné z: http://-www.elearn.vsb.cz/archivcd/FEI/LSA/Logicke%20systemy%20rizeni.pdf

ODSTRČILÍKOVÁ, Miroslava. *Automatizace*. Brno, 2003. Dostupné z: http://sps-se.kvalitne.cz/AT/AUT_3r_1cast.pdf. Naučná publikace. Střední průmyslová škola elektrotechnická, Brno.

PAVELKOVÁ, Naděžda. Moderní pohony s asynchronními motory a měniči frekvence. *Řízené elektrické pohony* [online]. 2010, č. 5, s. 30-33 [cit. 2014-10-20]. Dostupné z: http://automa.cz/res/pdf/41060.pdf

PETROVAS, Andrius, Saulius LISAUSKAS a Roma RINKEVICIENE. *Digital Automatic Control System with PID Controller*. Vilnius: Vilnius Gediminas Technical University, 2011, roč. 2011, č. 4. ISSN 1392 – 1215.

POUCHA, Pavel. Ethernet nahrazuje klasické průmyslové komunikační linky. In: Automa [online]. 2006 [cit. 2014-11-23]. Dostupné z: http://automa.cz/-index.php?id_document=31237

ROUŠ, Robert. *Řízení robotu KATANA v prostředí Control* Web. Brno, 2012. Diplomová práce. Mendelova univerzita.

SCHMIDT, T. – HOERCHER, G.: Netzaufbau, 4.2 Kabel. SPS Magazin, 11/2002, s. 36 – 39

SKIDELSKY, Robert. Project Syndicate. *The Rise of the Robots* [online]. Londýn, 2013 [cit. 2014-12-21]. Dostupné z: http://www.project-syndicate.org/commentary/the-future-of-work-in-a-world-of-automation-by-robert-skidelsky

SMUTNÝ, Lubomír a Radim KLEČKA. E-Automatizace. *Prostředky automatického řízení* [online]. 2005 [cit. 2014-10-12]. Dostupné z: http://www.e-automatizace.cz/ebooks/ridici_systemy_akcni_cleny/

ŠIKUT, Karel. Optické závory pro detekci objektů. *Automa* [online]. 2010, č. 1 [cit. 2014-10-12]. Dostupné z: http://odbornecasopisy.cz/res/pdf/40383.pdf

ŠMEJKAL, Ladislav a Luboš URBAN. Programovatelné automaty – PLC, nebo PAC?. In: *Odborné časopisy* [online]. 2007 [cit. 2014-11-02]. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=28832

ŠVARC, Ivan. *Automatizace: automatické řízení*. Brno: CERM, 2002, 201 s. ISBN 80-214-2087-1.

VÁVRA, Václav. *Základy elektrických pohonů*. Liberec, 2004. Dostupné z: https://www.pslib.cz/pe/skola/studijni_materialy/motory/obecne/002-Motory_TYPY_33str.pdf. Skripta. Vysoká škola báňská - Technická univerzita Ostrava.

VOJÁČEK, Antonín. Odměřování polohy s přesností na 0,001 mm. *Automatizace.hw.cz* [online]. 2010 [cit. 2014-10-12]. Dostupné z: http://automatizace.hw.cz/odmerovani-polohy-s-presnosti-na-0001-mm

ZEHNULA, Karel. *Snímače neelektrických veličin*. 2.vydání. Praha: Nakladatelství technické literatury, 1996, 371 s.

ZEZULKA, František. *Prostředky průmyslové automatizace*. 1. vyd. Brno: VUTIUM, 2004, 176 s. ISBN 80-214-2610-1.

ZEZULKA, František. *Průmyslový Ethernet VII*: Přehled současných standardů. *Automa* [online]. 2008 [cit. 2014-11-16]. Dostupné z: http://automa.cz-/res/pdf/36694.pdf

Přílohy

A Zdrojový kód funkčního bloku pro výpočet kolizních stavů

```
FUNCTION_BLOCK CalcColission
IF(Enable)THEN
     FOR k:=1 TO 9 DO
     FOR j:=1 TO 9 DO
          a1 := ViceColission.Pos[k].y - ViceColission.Pos[k+1].y;
          b1 := ViceColission.Pos[k+1].x - ViceColission.Pos[k].x;
          c1 := (ViceColission.Pos[k].x*ViceColission.Pos[k+1].y)-(ViceColission.Pos[k+1].x*Vice-
               Colission.Pos[k].y);

          a2 := SawColission.Pos[j].y - SawColission.Pos[j+1].y;
          b2 := SawColission.Pos[j+1].x - SawColission.Pos[j].x;
          c2  :=  (SawColission.Pos[j].x*SawColission.Pos[j+1].y)-(SawColission.  Pos[j+1].x*Saw-
               Colission.Pos[j].y);
// kazda usecka klestiny se pocita zda se neprotne(prusecik) snejakou   useckou pily, jakmile
// se protnou mezi useckami je tam kolize, kdyz se protnou za hranici usecky - kolize tam neni
          denom := (a1 * b2) - (a2 * b1);
          IF (denom <>0) THEN
               tmp_col := TRUE;
               ipx := (b1*c2 - b2*c1)/denom;
               ipy := (a2*c1 - a1*c2)/denom;
               IF ((EXPT((ipx-ViceColission.Pos[k].x),2) + EXPT((ipy-ViceColission.Pos[k].y),2))
                                                  >
                    (EXPT((ViceColission.Pos[k+1].x - ViceColission.Pos[k].x),2) +
                    EXPT((ViceColission.Pos[k+1].y - ViceColission.Pos[k].y),2)))
               THEN
                    tmp_col := FALSE;
               END_IF
               IF (EXPT((ipx-ViceColission.Pos[k+1].x),2)+EXPT((ipy-ViceColission.Pos[k+1].y),2)
                                                  >
                    (EXPT((ViceColission.Pos[k+1].x - ViceColission.Pos[k].x),2) +
                    EXPT((ViceColission.Pos[k+1].y - ViceColission.Pos[k].y),2)) )
               THEN
                    tmp_col := FALSE;
               END_IF
               IF ((EXPT((ipx-SawColission.Pos[j].x),2) + EXPT((ipy-SawColission.Pos[j].y),2))
                                                  >
                    (EXPT((SawColission.Pos[j+1].x - SawColission.Pos[j].x),2) +
                    EXPT((SawColission.Pos[j+1].y - SawColission.Pos[j].y),2)) )
               THEN
                    tmp_col := FALSE;
```

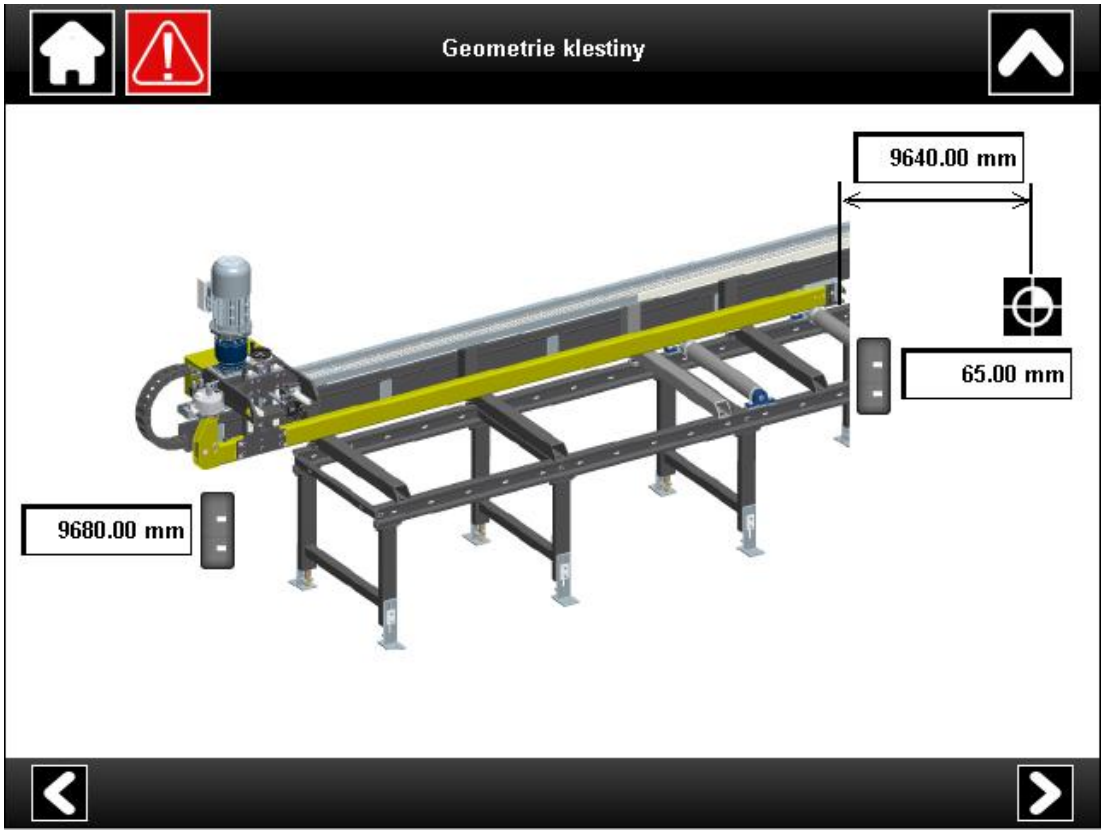
```
                END_IF
                IF (( EXPT((ipx-SawColission.Pos[j+1].x),2) +
                     EXPT((ipy-SawColission.Pos[j+1].y),2))
                                          >
                     (EXPT((SawColission.Pos[j+1].x - SawColission.Pos[j].x),2) +
                     EXPT((SawColission.Pos[j+1].y - SawColission.Pos[j].y),2)) )
                THEN
                     tmp_col := FALSE;
                END_IF
           ELSE
                tmp_col := FALSE;
           END_IF
           IF (tmp_col <> FALSE) THEN
                     State_Colission := State_Colission OR tmp_col;
           END_IF;
     END_FOR;
     END_FOR;
END_IF;
```

B Ukázka zdrojového kódu – hledání konce materiálu

```
FIND_E_MAT :              // SEARCHING END OF MATERIAL
     IF ITF_Feeder.State.Homing_OK THEN
          CASE RS_Loading.STATE_TMP OF
               0:         // Otevrit celisti klestiny  a pojezd s klestinou do vychozi polohy
                    RS_Moving_Vice.Param.Service        := FALSE;
                    RS_Clamping_Vice.Cmd.Open           := TRUE;
                    RS_Loading.STATE_TMP                := 1;
               1:
                    IF Stepping((NOT RS_Clamping_Vice.Cmd.Open AND RS_Clamping_Vice.State.Open),
                         Krokovani_Enabled,Next_state ) THEN
                         RS_Loading.STATE_TMP           := 2;
                    END_IF;
               2:         // Najdi konec  tyce na vstupni trati
                    RS_Moving_Vice.Cmd.MoveBack         := TRUE;
                    RS_Loading.STATE_TMP                := 5;
               5:         // detekce materialu v klestine
                    IF( DI_SQ_Vice_MatDetection )THEN
                         RS_Moving_Vice.Cmd.MoveBack    := FALSE;
                         RS_Loading.STATE_TMP           := 6;
                    ELSIF (RS_Moving_Vice.State.Position <= C_MAX_LENGHT_SAW_LASER + 100) THEN
                         RS_Moving_Vice.Cmd.MoveBack    := FALSE;
                         RS_Loading.STATE_TMP           := 7; // material nenalezen
                    END_IF
               6:         //priprava materialu k pojezdu na laser
                    IF Stepping((NOT RS_Moving_Vice.Cmd.MoveBack AND NOT RS_Moving_Vice.State.-
                    InMove),Krokovani_Enabled,Next_state ) THEN
                         RS_Loading.STATE_TMP           := 10;
                    END_IF;
               7:         //Vrati se do vychozi pozice a zahlasi chybu - material nenalezen
                    IF Stepping((NOT RS_Moving_Vice.Cmd.MoveBack AND NOT RS_Moving_Vice.State.-
                         InMove),Krokovani_Enabled,Next_state) THEN
                         RS_Loading.STATE_TMP           := 8;
                    END_IF;
               8:
                    RS_Moving_Vice.Param.Position := Retain_Moving_Vice.Setting.Position.Homing;
                    RS_Moving_Vice.Cmd.MoveToPosition   := TRUE;
                    RS_Loading.STATE_TMP                := 9;
               9:
                    IF Stepping((NOT RS_Moving_Vice.Cmd.MoveToPosition AND RS_Moving_Vice.State-
                         .InPosition),Krokovani_Enabled,Next_state )THEN
```

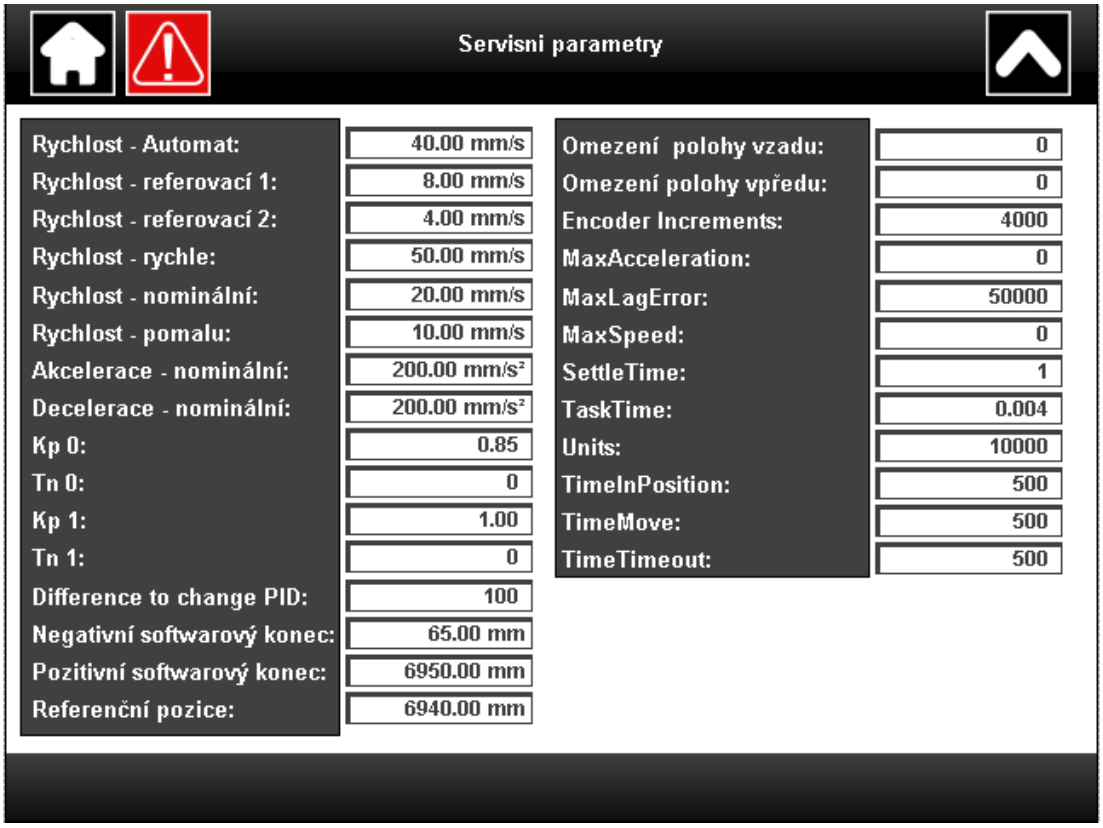
```
                    RS_Loading.STATE              := ERROR;
                    RS_Loading.STATE_TMP          := 0;
                    RS_Loading.Error              := TRUE;
                    RS_Loading.ErrorID            := 200;   //Material not found
               END_IF
          10:
               RS_Moving_Vice.Param.MoveSlow      := TRUE;
               RS_Moving_Vice.Cmd.MoveBack        := TRUE;
               RS_Loading.STATE_TMP               := 11;
          11:
               IF(DI_SQ_Vice_MatDetection) THEN
                    RS_Moving_Vice.Param.MoveSlow := FALSE;
                    RS_Moving_Vice.Cmd.MoveBack   := FALSE;
                    RS_Loading.STATE_TMP          := 12;
               END_IF;
          12:
               IF Stepping((NOT RS_Moving_Vice.Cmd.MoveBack AND NOT RS_Moving_Vice.State.-
                    InMove),Krokovani_Enabled,Next_state )THEN
                    RS_Loading.STATE_TMP          := 13;
               END_IF;
          13:  // Najede s upnutym materialem  na laserovou zavoru
               RS_Clamping_Vice.Cmd.Close         := TRUE;
               RS_Loading.STATE_TMP               := 14;
          14:
               IF Stepping((NOT RS_Clamping_Vice.Cmd.Close AND RS_Clamping_Vice.State.
                    Close),Krokovani_Enabled,Next_state ) THEN
                    ITF_Feeder.State.MatHolded    := TRUE;
                    ITF_Feeder.State.MatFound_E   := TRUE;
                    RS_Loading.STATE_TMP          := 0;
                    RS_Loading.STATE              := WAITING;
               END_IF;
     END_CASE
ELSE                 //neni zreferovano pred hledanim konce materialu
     RS_Loading.STATE                             := ERROR;
     RS_Loading.STATE_TMP                         := 0;
     RS_Loading.Error                             := TRUE;
     RS_Loading.ErrorID                           := 100;
END_IF;
```

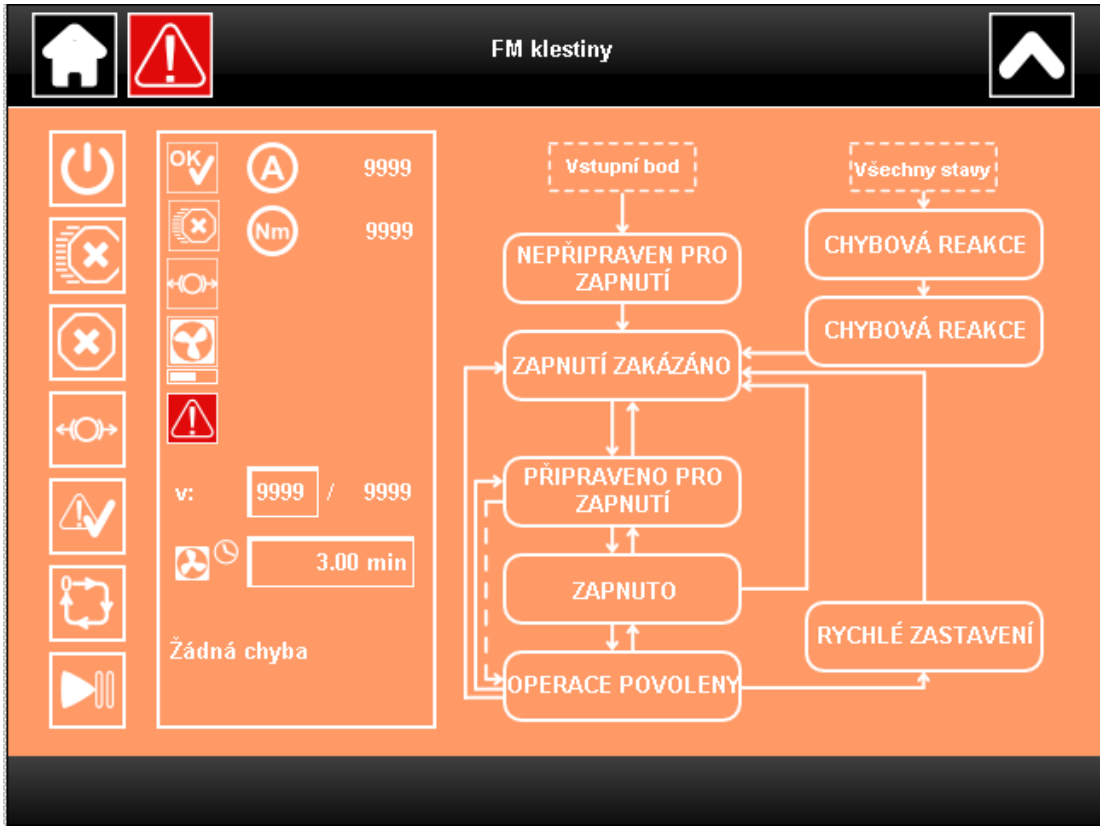
C Vytvořené stránky vizualizace pro dotykový operátorský panel




Reference klestiny
REFEROVAT KLESTINU
FIND MATERIAL
STOP


Manualni pohyby
s: 0.00 mm
v: 0.00 m/min
25
1 2 3
4 5 6
7 8 9
± 0 .


Servisni parametry
Rychlost - Automat: 40.00 mm/s
Rychlost - referovací 1: 8.00 mm/s
Rychlost - referovací 2: 4.00 mm/s
Rychlost - rychle: 50.00 mm/s
Rychlost - nominální: 20.00 mm/s
Rychlost - pomalu: 10.00 mm/s
Akcelerace - nominální: 200.00 mm/s²
Decelerace - nominální: 200.00 mm/s²
Kp 0: 0.85
Tn 0: 0
Kp 1: 1.00
Tn 1: 0
Difference to change PID: 100
Negativní softwarový konec: 65.00 mm
Pozitivní softwarový konec: 6950.00 mm
Referenční pozice: 6940.00 mm
Omezení polohy vzadu: 0
Omezení polohy vpředu: 0
Encoder Increments: 4000
MaxAcceleration: 0
MaxLagError: 50000
MaxSpeed: 0
SettleTime: 1
TaskTime: 0.004
Units: 10000
TimeInPosition: 500
TimeMove: 500
TimeTimeout: 500


FM klestiny
OK
A
9999
Nm
9999
v:
9999
/
9999
3.00 min
Žádná chyba
Vstupní bod
Všechny stavy
NEPŘIPRAVEN PRO ZAPNUTÍ
CHYBOVÁ REAKCE
CHYBOVÁ REAKCE
ZAPNUTÍ ZAKÁZÁNO
PŘIPRAVENO PRO ZAPNUTÍ
ZAPNUTO
RYCHLÉ ZASTAVENÍ
OPERACE POVOLENY

D Přiložené CD

Přiložené CD obsahuje:

- Zdrojové kódy aplikace
- Závěrečnou práci v elektronické podobě
- Vývojový diagram
- Ukázky jednotlivých stránek vizualizace ve vysokém rozlišení