VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

NÁVRH AUTOMATIZOVANÉ SESTAVY NÁSTROJŮ TESTOVACÍ STOLICE PRO VÝROBU STÍNÍCÍCH LAMEL

PROPOSAL FOR AUTOMATED ASSEMBLY TOOLS FOR PRODUCTION TESTING OF STOOL SCREENING SLATS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Roman Gricman

VEDOUCÍ PRÁCE
SUPERVISOR
prof. Ing. Zdeněk Kolíbal, CSc.

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Student:	**Roman Gricman**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**prof. Ing. Zdeněk Kolíbal, CSc.**
Akademický rok:	2015/16

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

Návrh automatizované sestavy nástrojů testovací stolice pro výrobu stínících lamel

Stručná charakteristika problematiky úkolu:

Jedná se o kreativní úkol spočívající v komplexním návrhu složení a naprogramování funkce činnosti sestavy potřebných nástrojů

Cíle bakalářské práce:

Bude vytvořen návrh části zkušební stolice pro automatické zakládání výrobních nástrojů a dílců.

Seznam literatury:

CHVÁLA, B.- NEDBAL, J.- DUNAY, G.: Automatizace. SNTL Praha, 1989

CHVÁLA,B.: Mechanizace a automatizace obráběcích strojů. SNTL Praha, 1970

JACHYMOVIČ, V.A.: Orientační mechanizmy montážních automatů a robotů. SNTL Praha,1980

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2015/16

V Brně, dne

L.S

prof. Ing. Jindřich Petruška, CSc.	doc. Ing. Jaroslav Katolický, Ph.D.
ředitel ústavu	děkan fakulty

ABSTRAKT

Práce se zabývá návrhem vývojové testovací stolice a naprogramováním funkce nových nástrojů a ovládacího panelu stolice. Jsou v ní popsány funkce a program nástrojů, návrh testovací stolice, komponenty stolice, aplikace vytvořena pro ovládací panel, řídící systém a vývojové prostředí potřebné pro naprogramování nástrojů. Tato stolice poskytuje možnost vyzkoušení nástrojů, což umožňuje kontrolu nástroje před namontováním na stroj.

Klíčová slova
Testovací stolice, Mosaic, Visual Studio

ABSTRACT

This bachelor thesis deals with the development of a testing stool, programming function of a new tools and control panel of the testing stool. It describes a function of tools, design of testing stool, stool components, application created for a control panel, the control system and development environment needed for programming of tools. This stool provides the ability to test the tools, which allows a control of a tool before it's mounted on the machine.

Keywords
Testing stool, Mosaic, Visual studio

Bibliografická citace práce:

GRICMAN, R. *Návrh automatizované sestavy nástrojů testovací stolice pro výrobu stínících lamel*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2016. 50 s. Vedoucí bakalářské práce prof. Ing. Zdeněk Kolíbal, CSc.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením prof. Ing. Zdeněk Kolíbal, CSc. a s použitím literatury uvedené v seznamu

……

Roman Gricman

BRNO 2015

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat společnosti Zebr s.r.o., že mi umožnila pracovat na tomto projektu a umožnila mi zdokonalit své znalosti. Zároveň chci poděkovat všem pracovníkům společnosti, kteří mi poskytli odborné dokumenty a potřebné doplňující informace. Rád bych ocenil svoji rodinu a přátelé za podporu během celého studia i při psaní této práce.

BRNO 2015

OBSAH

1 ÚVOD

Při splňování čím dále více náročných podmínek zákazníků na výrobní čas stínících lamel, je zapotřebí vyvíjet nové nástroje urychlující proces výroby lamely s požadovanými prvky (díry, háčky, podkovy atd.). K tomuto účelu slouží navržená vývojová testovací stolice schopná pojmout jakýkoliv nástroj používaný k výrobě lamel. Práce je vytvořena pro společnost Zebr s.r.o. V této společnosti probíhá moje praxe, kterou mi firma umožnila během mého studia na VUT.

Firma Zebr s.r.o. se soustředí především na výrobu strojů, které plně automatizují proces výroby stínících lamel. Ty se nejčastěji skládají ze tří různých částí. První část slouží k válcování lamely do požadovaného tvaru. Druhá část stroje se skládá s různých druhů nástrojů, které slouží k výrobě díry nebo k rozdělení lamely. Poslední část převážně slouží k odkládání vyrobených lamel.

2 CÍLE PRÁCE

Hlavním cílem bakalářské práce byl návrh jednoduché konstrukce testovací stolice a následné odzkoušení funkce nástrojů na testovací stolici. Tyto nástroje byly odzkoušeny pomocí ovládacího panelu obsahující nově vytvořenou aplikaci ve formulářovém prostředí Visual Studio.

První kapitoly práce jsou zaměřeny na popis konstrukce navržené testovací stolice a veškerých komponent elektro rozvaděče. V němž jsou umístěny všechny prvky potřebné pro chod stolice a nástrojů. Hlavním prvkem, který byl popsán je řídící systém umožňující naprogramování nástrojů pomocí prostředí Mosaic.

Druhá část práce se věnuje popisu funkce a naprogramování nově vyrobených nástrojů a následné ukázce činnosti těchto nástrojů. Poslední část práce se zaměřuje na aplikaci ovládacího panelu, kde se popisují jednotlivé funkce formulářových prvků.

3 NÁVRH TESTOVACÍ STOLICE

Testovací stolice má sloužit jako univerzální stůl, na který je možné připojit nástroje, které jsou ve vývoji a je nutné je odzkoušet a doladit než přijdou ve finální verzi na stroj zákazníka.

Hlavním úkolem bylo navržení konstrukce testovací stolice, umožňující snadnou montáž a manipulaci. Konstrukce byla navržena ze systému hliníkových profilů značky Item (od firmy Haberkorn Ulmer s.r.o.), které jsou ideálním řešením právě pro nejrůznějších jednoúčelové stroje a zařízení. Základem Item systému jsou přesné eloxované hliníkové profily s podélnými drážkami a otvory pro upevnění spojovacích prvků a rozsáhlého příslušenství. Povrchové plochy jsou odolné proti korozi a poškrábání. Koncepce systému umožňuje velkou pružnost, vysokou přesnost i pevnost, ale také rychlou přestavitelnost a možnost opakovaného použití prvků.

Stolice se skládá ze tří částí. Dvou bočních stolů a jednoho středního stolu. Ke stolici náleží i samostatně stojící ovládací panel, s jehož pomocí lze odzkoušet dané komponenty pomocí navržené aplikace viz kapitola 8.

3.1 Boční stoly

Boční stůl je složený pomocí jednoduchých hliníkových Itemů do obdélníkového tvaru, se čtyřmi nastavitelnými nohami pro nastavení rovinnosti a stejné výšky všech stolů testovací stolice. Pro spojování hliníkových Itemů byly použity speciálně tvarované matice, které lze koupit od výrobce Itemů.

Ke každému stolu je připevněn odkládací žlab, který slouží k odkládání používaného nářadí, například šroubováky, imbusové klíče, spojovací materiál atd.

Boční stoly jsou umístěny z každé strany středového stolu. Je na nich možné aplikovat různé komponenty jako například dělící či děrovací nástroje, nastřelování či posouvání. Využití však najdou také jako skladovací prostory pro lamely na testování posouvadel. Lze však použít i různá čidla na kontrolu rychlosti posunu lamely apod.

Každý boční stůl obsahuje namontovaný ventilový terminál, který bude sloužit k ovládání pneumatických komponentů namontovaných nástrojů. Přesnou konstrukci a seznam použitých prvků najdeme v příloze A1 Výkresová dokumentace.

3.2 Středový stůl

Středový stůl je nejdůležitější částí testovací stolice, jelikož se zde nachází veškeré vstupní parametry potřebné ke zkoušení komponentů.

Z hlediska konstrukce byl středový stůl navržen pro umístění těchto vstupních parametrů, ale i také pro umístění nového nýtovacího nástroje. Přesnou konstrukci a seznam použitých prvků najdeme v příloze A2 Výkresová dokumentace.

Vstupy:
- Vzdušník
- Hydraulický agregát
- Elektro rozvaděč
- Ovládací panel

3.2.1 Vzdušník

Vzdušník o objemu 4L je výrobkem firmy Zebr s.r.o.. Z něj je vedeno 12 vývodů, které slouží pro rozvod pracovního tlaku do ventilových terminálů.

3.2.2 Hydraulický agregát

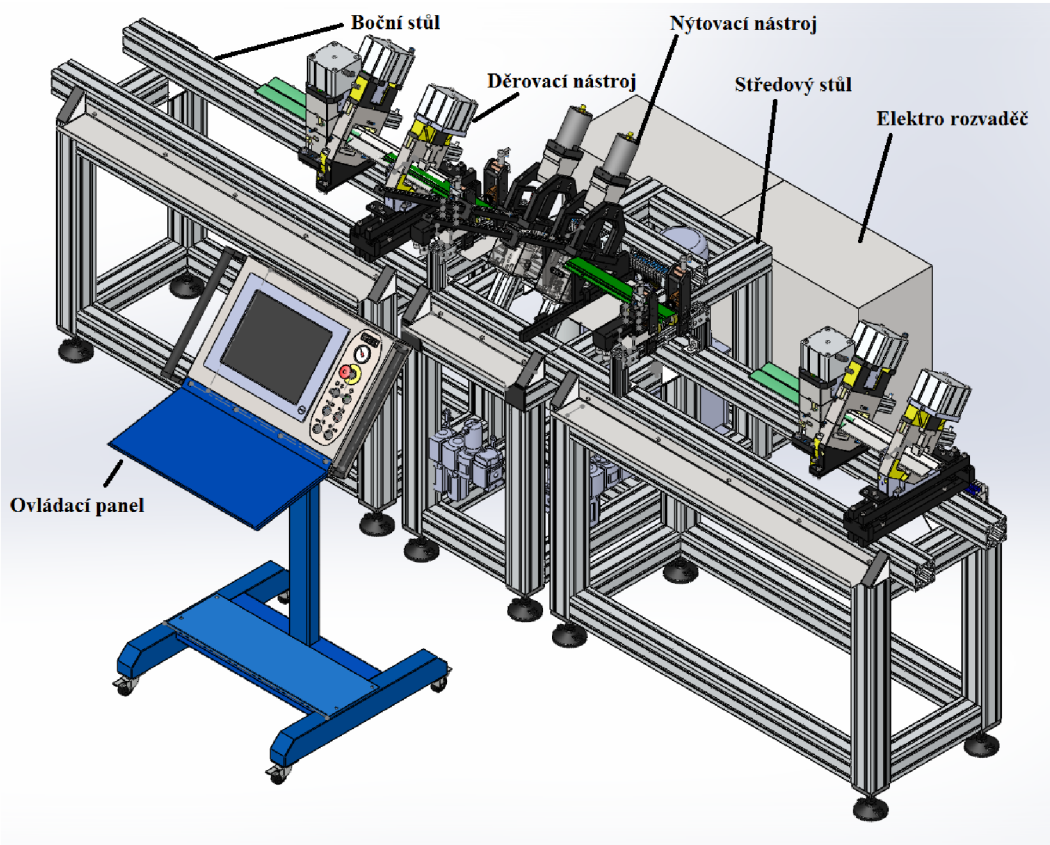
Hydraulické agregáty od firmy Argo-Hytos jsou určeny pro zástavbu do malých prostorů a mohou být použity u zdvihacích plošin, zvedacích stolů, manipulačních zařízení, malých lisů, obráběcích strojů a mobilních aplikací. Agregát se skládá z elektromotoru, zubového čerpadla, řídicího bloku a nádrže. Centrální blok, odlitý ze slitiny hliníku, tvoří zároveň nosnou část pro připevnění hydraulických prvků [9].

3.2.3 Elektro rozvaděč

Elektro rozvaděč je skříň, která obsahuje elektrické přístroje pro měření, ovládání a jištění elektroinstalace. Do rozvaděče je zavedeno několik kabelů, jenž jsou nutné pro jeho funkci a chod testovací stolice, více o obsahu elektro rozvaděče bude popsáno v kapitole 3.

3.2.4 Ovládací panel

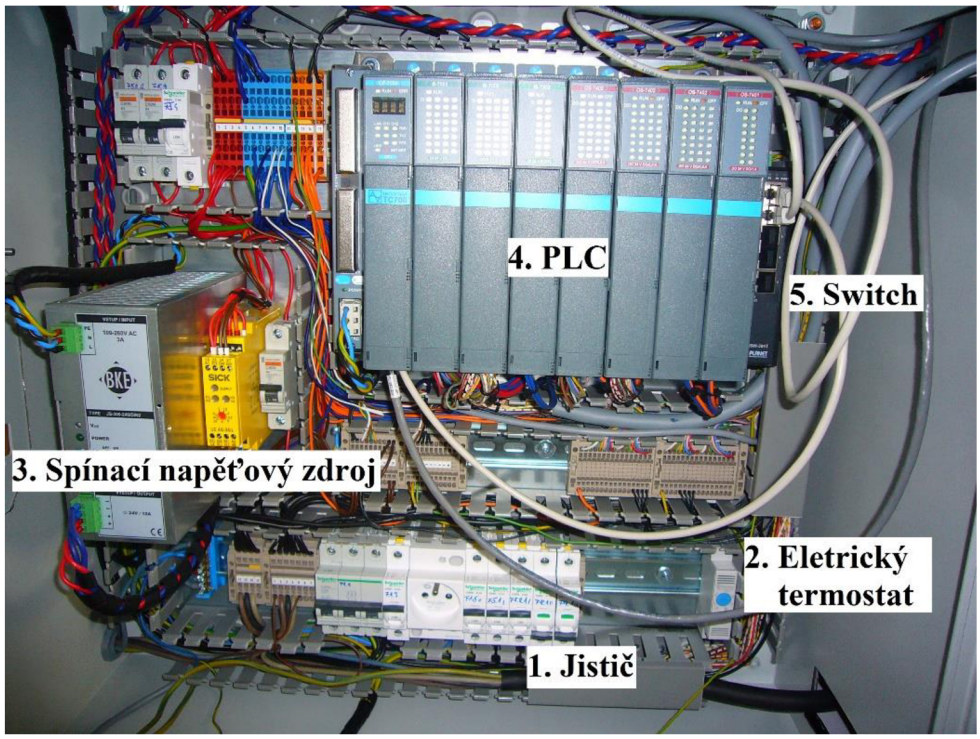
Rám ovládacího panelu je vyroben pomocí ocelových čtvercových profilů, které jsou navzájem spojeny pomocí svařování. Hlavní důvod použití tohoto rámu spočívá v jeho dutém tvaru, který nám poskytuje větší odolnost natažených kabelů. Hlavní komponentou panelu je dotykový počítač obsahující námi naprogramovanou aplikaci viz kapitola 7, která slouží k nastavení a ovládání nástrojů testovací stolici.



Obrázek 1 – Navržená Konstrukce testovací stolice

4 ELEKTRO ROZVADĚČ

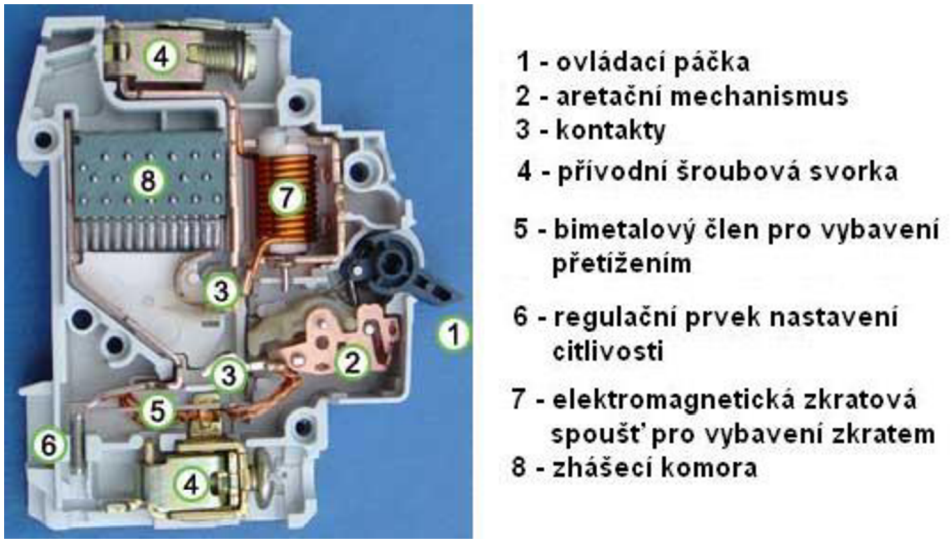
Elektro rozvaděč je skříň, která obsahuje elektrické přístroje ovládání a jištění elektroinstalace, které zajišťují správný chod nástrojů umístěných na testovací stolici. V našem případě elektro rozvaděč obsahuje tyto komponenty:



Obrázek 2 – První část elektro rozvaděče

4.1 Modulární jistič

Modulární jistič je elektrický přístroj, který chrání elektrický obvod bezpečným automatickým rozpojením obvodu pří zkratu či překročení jmenovitého proudu s reakcí až 4 - 25ms. Konstrukci modulárního jističe je možné vidět na obrázku 3.



Obrázek 3 – Konstrukce modulárního jističe

Správnou funkci jističe zajišťuje tepelná a zkratová spoušť. Tepelná spoušť je tvořena bimetalovým páskem, vloženým do proudové dráhy a chrání vedení proti přetížení proudem, který je větší než jmenovitý. Princip ochrany je založen na tepelné roztažnosti, kdy se v případě většího proudu bimetalový pásek prohne a uvolní západku. Dochází k vypnutí jističe.

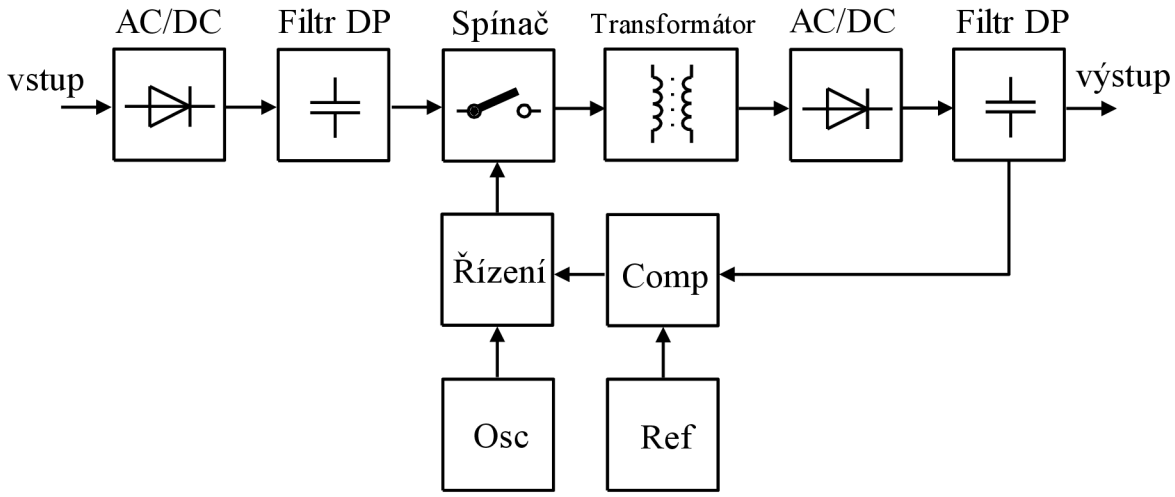
V případě zkratových proudů je zapotřebí co nejrychleji odpojit chráněný obvod, což zajišťuje elektromagnetická zkratová spoušť. Skládá se z cívky vložené do proudové dráhy a v případě, že dojde v obvodu ke zkratu a cívkou začne téct zkratový proud, vytvoří se magnetické pole, které přitáhne kotvu vypínajícího jističe. Tím dojde k vypnutí jističe.

4.2 Elektrický termostat

Slouží k udržování stále teploty v rozvaděči. Při dosažení nastavené teploty termostatu, se sepne ventilátor umístěný na dveřích rozvodné skříně, který odvede teplý vzduch z rozvaděče.

4.3 Spínací napěťový zdroj

Spínací zdroj slouží k transformaci síťového napětí na požadované napětí 24 V a 12 A. Výstupní napětí tohoto zdroje je získáno ze vstupního pomocí spínání. Principiální blokové schéma spínaného zdroje ukazuje obrázek 4.



Obrázek 4 - Obecné blokové schéma spínaného zdroje.

4.3.1 Princip činnosti spínacího napěťového zdroje

Vstupní napětí je v usměrňovači (AC/DC měniči) přeměněno na stejnosměrné. V praxi je často požadováno velmi dobré vyhlazení střídavé složky. Děje se tak buďto analogovým filtrem typu dolní propust (DP) nebo díky zpětné vazbě ve spínané části. Stejnosměrné napětí přichází na spínač, kterým bývá tranzistor spolu s dalšími pomocnými obvody. Pomocí spínače, jehož kmitočet je svázán s kmitočtem oscilátoru (Osc), je stejnosměrné napětí přeměněno na střídavé obdélníkového nebo trojúhelníkového průběhu. Kmitočet spínání bývá různě vysoký podle typu a určení zdroje. Velikost tohoto střídavého napětí je poté transformátorem upravena podle potřeby. Je-li požadováno na výstupu stejnosměrné napětí, může být výstupní napětí transformátoru dále

usměrněno. Zdroj obsahuje zpětnou vazbu, pomocí které je řízena logika spínání spínače a která zajišťuje rovněž stabilizaci (regulace). Zde se používají bloky jako komparátory (Comp), zdroje referenčních napětí (Ref) apod.

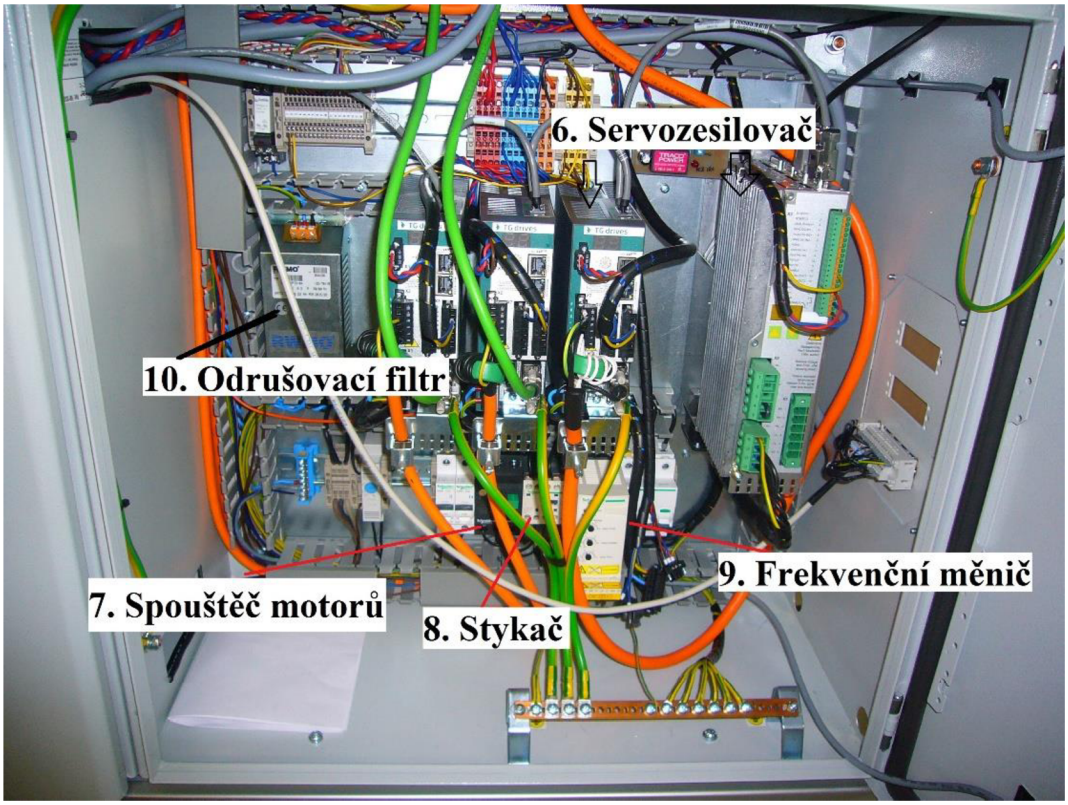
Navržený spínací zdroj je určený pro napájení zařízení průmyslové automatizace, řídící a regulační techniky, jelikož je odolný proti rušení [5].

4.4 Programovatelný automat

Dále PLC (Programmable Logic Controller = programovatelný automat) je malý průmyslový počítač používaný pro automatizaci procesů v reálném čase. V našem případě byl použit modulární řídící systém Tecomat TC700 viz kapitola 4.

4.5 Switch

Switch je zařízení sloužící k propojení ovládacího panelu s PLC či k napojení programátora na PLC.



Obrázek 5 – Druhá část elektro rozvaděče

4.6 Servozesilovač

Servozesilovač slouží jako ústřední člen pro ovládání servopohonu, který v sobě spojuje funkci napájení a řízení samotného motoru a funkci pro ovládání tohoto motoru z nadřazeného systému, v tomto případě PLC. Pro realizaci zpětné vazby lze použít řadu snímačů implementovaných přímo na osu motoru. Pro komunikaci mezi servozesilovačem a nadřazených systémem lze použít různé typy sítí (CAN, EtherCAT, Profinet a Ethernet IP). V tomto případě byly do rozvaděče umístěny dva druhy servozesilovačů od fimy TG Drives.

Tyto typy servozesilovačů byly určeny pro řízení synchronních servopohonů, obsahují prostředky pro napájení motoru a pro jeho řízení za použití tří kaskádně zapojených regulačních smyček. Na nejnižší úrovni je implementován regulátor proudu pro napájenou fázi. Nad tímto regulátorem funguje rychlostní a dále pak polohový regulátor. Pro napájení synchronního motoru se využívá pulsně šířkové modulace, což přináší vysokou flexibilitu pro generování napěťové charakteristiky.

Tyto servozesilovače jsou vybaveny vstupy pro komunikaci s mnoha druhy snímačů polohy, což nabízí velkou škálu řešení z hlediska použitého zpětnovazebního snímače i aplikace servopohonu [7], [8].

TGA 300	AKD 3006
Digitální vstupy (4) a výstupy (2)	Digitální vstupy (7) a výstupy (2)
Digitální proudová (32,25 µs), rychlostní (62,5 µs) a polohová (125 µs) regulační smyčka	Digitální proudová (0,67 µs), rychlostní (62,5 µs) a polohová (125 µs) regulační smyčka
Vybaveny odrušovacím síťovým filtrem	Vybaveny odrušovacím síťovým filtrem v případě napájení 3x400 V
Řízení pomocí komunikačních rozhraní RS232, CANopen	Řízení pomocí komunikačních rozhraní CAN, EtherCAT, Profinet a Ethernet IP
Levnější v případě řízení větších motorů	Dražší v případě řízení větších motorů

Tabulka 1 - Porovnání rozdílných funkcí servozesilovačů

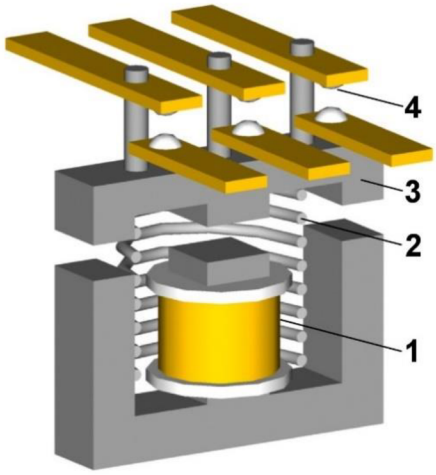
4.7 Spouštěč motorů

Tento typ jističů zaručuje ochranu motorů jak proti zkratům, tak i proti nadproudu rychlým přerušením obvodu s poruchou. Jde o kombinaci elektromagnetického jističe a nadproudového relé. V těchto spouštěčích má ochrana proti zkratům pevnou mez, obvykle třináctinásobek maximálního proudového nastavení tepelné ochrany.

Tepelná ochrana (proti nadproudu) je kompenzována na změny okolní teploty. Nastavení tepelné ochrany lze měnit zepředu spouštěče. Nastavená hodnota musí odpovídat jmenovitému proudu chráněného motoru. Kromě toho je vzdálenost mezi kontakty ve vypnutém stavu dostatečná pro zajištění izolace.

4.8 Stykač

Stykač slouží k spojení nebo rozpojení elektrického spojení. Na obr. je možné vidět princip a vnitřní konstrukci třífázového stykače.



Obrázek 6 - Popis funkce a vnitřní konstrukce stykače

1. – Cívka s jádrem
2. – Vratná pružina
3. – Kotva
4. – Silový spínací kontakt

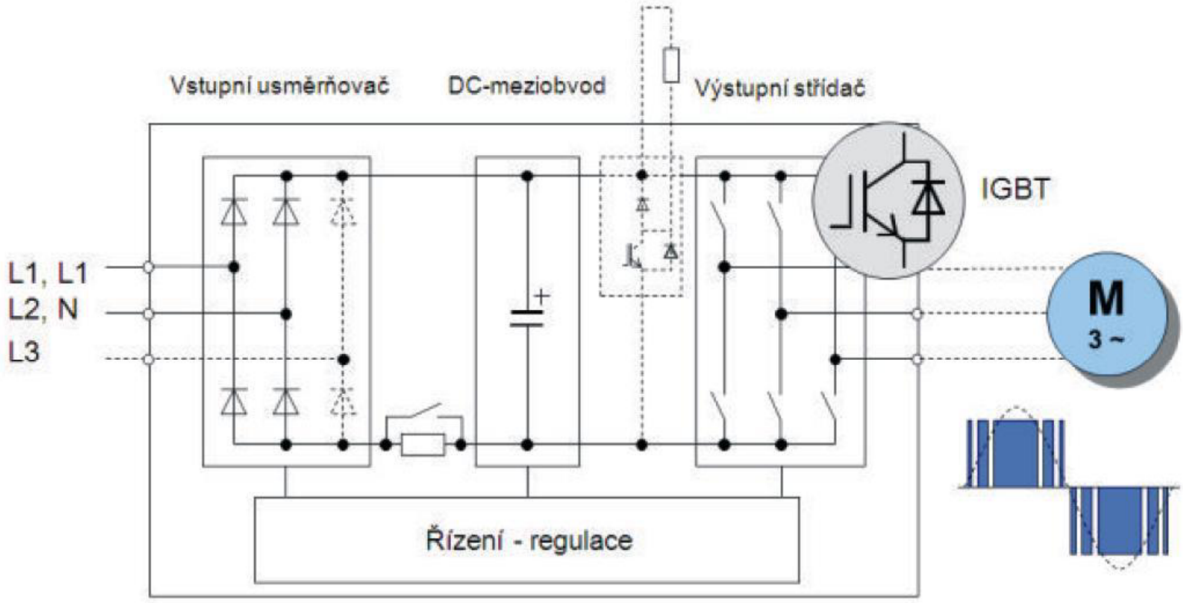
4.8.1 Princip činnosti stykače

Když přivedeme napětí na svorky cívky stykače, vznikne magnetické pole a kotva s pohyblivými kontakty je přitažena k pevným kontaktům. Tím dojde k elektrickému spojení.

4.9 Frekvenční měnič

Při napájení asynchronních elektromotorů, je zapotřebí použít frekvenční měnič (nepřímý měnič frekvence), který se skládá ze čtyř základních částí:

- Vstupní usměrňovač
- Stejnosměrný mezi obvod
- Střídač
- Řídící obvody



Obrázek 7 - Základní části frekvenčního měniče

Princip funkce nepřímého měniče frekvence spočívá v tom, že vstupní střídavé napětí je nejprve ve vstupním usměrňovači usměrněno na napětí stejnosměrné buď o konstantní, nebo proměnné velikosti a následně je z něj střídačem vytvořeno výstupní střídavé napětí o požadované efektivní hodnotě a frekvenci.

V našem případě je vstupní usměrňovač měniče neřízený, diodový, poskytuje tedy stejnosměrné napětí o konstantní velikosti. Stejnosměrný meziobvod obsahuje kondenzátor, který slouží ke stabilizaci napětí a také prvky s indukční zátěží k vyhlazení průběhu stejnosměrného napětí odebíraného ze sítě.

Střídač je zdroje výstupního napětí o proměnné velikosti a frekvenci čehož se dosahuje spínáním jednotlivých vypínatelných výkonových součástek pomocí některé z metod řízení, která pro nový takt modulační frekvence určuje tzv. poměrné otevření střídače. Průběh výstupního napětí a proudu je vyhlazován zátěži nebo pomocí LC sinusových filtrů.

Pro pohon asynchronních elektromotorů je nutno použít střídač, který je schopen zajistit komutaci svých součástek, nejběžněji se využívá třífázový střídač, který má šest větví. V každé větvi je vypínatelná součástka se zapojenou diodou, které dohromady tvoří spínač, jelikož v tomto případě tvoří dvě větve jednu fázi tak pak dva spínače ve fázi tvoří přepínač, ten připojuje fázi ke kladnému nebo zápornému přívodu napětí.

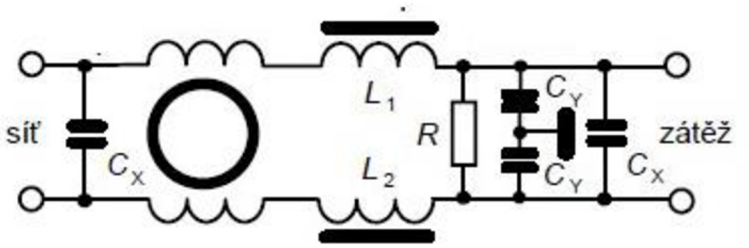
Z metod řízení střídačů se především používá metoda pulzně-širokého řízení (modulace), ale existuje i obdélníkové řízení, které se nevyužívá kvůli vysokému obsahu harmonických ve výstupních průbězích a špatným dynamickým vlastnostem.

Pulzně - široké řízení, PWM (Pulse Width Modulation), v PWM se okamžiky přepnutí spínače dané fáze určují jako průsečíky mezi trojúhelníkovým napětím a na stejné ose přidaným sinusovým referenčním napětím [3].

4.10 Jednofázové filtry pro měniče (odrušovací filtr)

Pro odrušení frekvenčních měničů se většinou používají odrušovací filtry. Filtr se připojuje mezi vstupní (síťové) svorky frekvenčního měniče a napájecí síť. Omezují tedy rušení, které generuje frekvenční měnič a jenž by se šířilo napájecí sítí. V našem případě byla použita varianta jednofázového filtru, jehož schéma zapojení je uvedeno na Obrázku 7.

Filtr obsahuje dvě proudově kompenzované tlumivky, jeden odrušovací kondenzátor třídy X zapojený mezi fázové vodiče, který odrušuje protifázové složky rušivých proudů. Součástí filtru jsou i dva bezpečnostní kondenzátory třídy Y mezi fázovými vodiči a zemnícím vodičem k odrušení soufázových složek rušivého signálu. Pro zvýšení útlumu je zde přidán další kondenzátor Cx a další dvě odrušovací tlumivky L1 a L2 do obou fází. Pro vybíjení kondenzátorů v době odpojení od napájecí sítě je použit odpor R, jehož hodnota se může pohybovat od stovek kΩ až po jednotky MΩ [4].



Obrázek 8 - Schéma zapojení odrušovacího filtru

5 PROGRAMOVATELNÝ AUTOMAT

Programovatelný automat, dále PLC (Programmable Logic Controller = programovatelný automat) je malý průmyslový počítač používaný pro automatizaci procesů v reálném čase. PLC se oproti normálnímu PC liší například tím, že vykonává program v cyklech. Jejich periferie jsou předem připraveny tak, aby mohly být propojeny s technologickými procesy.

Řídící algoritmus programovatelného automatu je zapsán jako posloupnost instrukcí v paměti uživatelského programu. Centrální jednotka postupně čte z této paměti jednotlivé instrukce, provádí příslušné operace s daty v zápisníkové paměti a zásobníku, případně provádí přechody v posloupnosti instrukcí, je-li instrukce ze skupiny organizačních instrukcí. Jsou-li provedeny všechny instrukce požadovaného algoritmu, provádí centrální jednotka aktualizaci výstupních proměnných do výstupních periferních modulů a aktualizuje stavy ze vstupních periferních modulů do zápisníkové paměti. Tento děj se stále opakuje a nazýváme jej cyklem programu.

5.1 PLC Tecomat

Programové automaty Tecomat jsou určené pro řízení nejrůznějších strojů z oblasti průmyslu, jako je strojírenství nebo hutnictví. Dále také z oblasti chemického, vodohospodářského a potravinářského průmyslu apod. Tyto automaty patří mezi světové standarty PLC. Jelikož náplní této práce není porovnávání PLC automatů různých značek, musí být vyzdviženy ty přednosti, kterými disponuje firma Teco a.s. a její automaty. Jeden z hlavních faktorů proč bylo rozhodnuto používat tuto variantu je jednoznačně software, který firma poskytuje zdarma ke svým automatům, a možnosti v tomto programovém vybavení jménem Mosaic.

V němž se programuje podle mezinárodního standartu IEC EN 61 131-3. Firma nabízí automaty pro široké využití, ať už jsou to automaty, které mají několik vstupů a výstupů (tzv. kompaktní systémy – TC400), přes větší kompaktní systémy TC500, TC600 a TC650, až po velké aplikace pro které jsou určeny PLC typu TC700 a NS950. Porovnání všech typů naleznete v tabulce 1 [2].

Typová řada	Binární I/O	Analogové I/O	Komunikační kanály	Displej/ klávesnice	Paměť programu
TC400	6/4	2/-	2 serial	externí	32 kB
TC500	20/20	4/4	2 serial	integrovaný	32 kB
TC600	48/44	24/8	3 serial	externí	32 kB
TC650	48/44	24/8	3 serial, 1 Ethernet	externí	64 + 64 kB
Foxtrot	až 134 DI/DO	až 80AI / 20AO	2× serial, 1×CIB, 1×Ethernet	externí	192 + 64 kB
TC700	přes 6500	až 3300/700	až 10 serial, 2× Ethernet, 1× USB	externí	192 + 64 kB

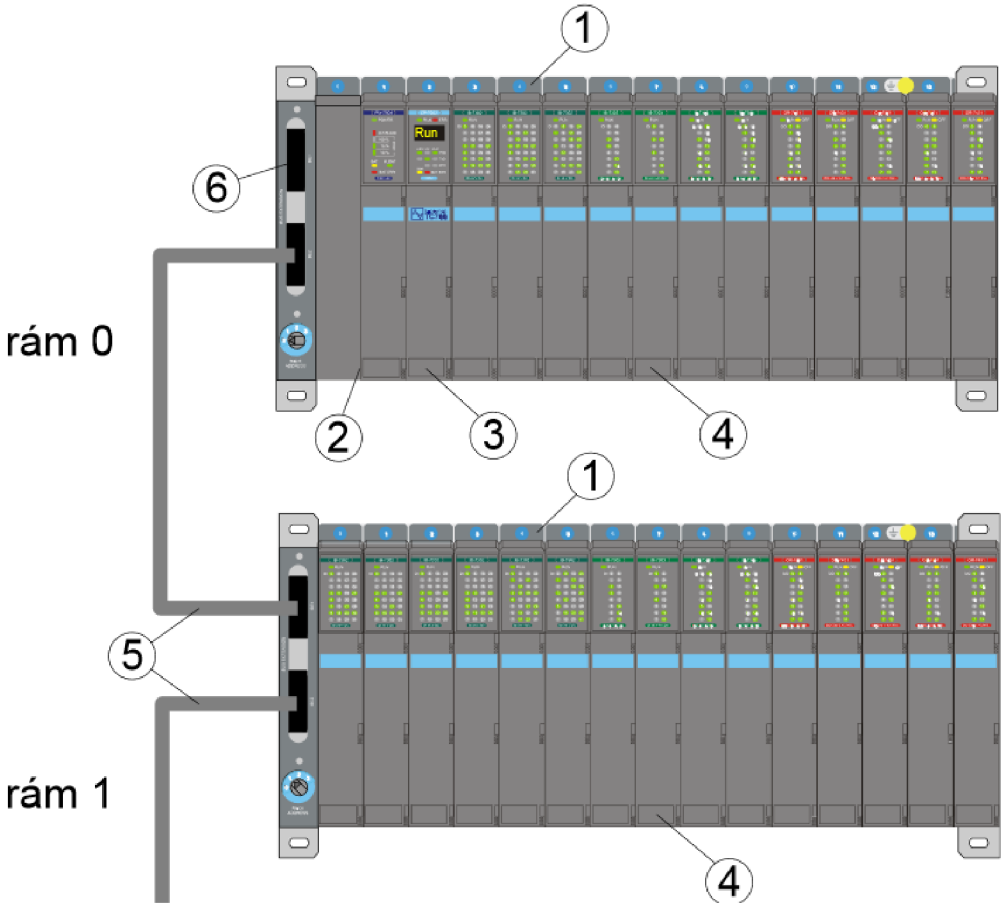
Tabulka 2 - Porovnávající tabulka typů systému

5.2 Tecomat TC700

V našem případě použijeme modulární systém Tecomat TC 700, který má širokou škálu využití od řízení technických zařízení budov až po složité řízení dopravy či strojů.

Základní sestava Tecomat TC 700 se skládá z nosného plochého rámu, v kterém jsou připojeny jednotlivé moduly. Každá taková soustava obsahuje jeden napájecí modul a centrální jednotku, zbytek modulů tvoří periferie. Nejdůležitějším modulem, který PLC musí obsahovat je centrální jednotka. Centrální jednotka provádí uživatelský program a obsahuje základní funkce bez, kterých se PLC neobejde. V našem případě využíváme centrální jednotku CP-7004 umožňující výstavbu PLC až do osmi rámů. Centrální jednotka CP – 7004 obsahuje 32 bitový RISC procesor s rychlostí až 0,2 ms/1000 instrukcí a integrovaný Web server.

Datové komunikace mezi PC a PLC nebo mezi PLC se obvykle realizovány sériovými přenosy. Systémy TC 700 podporují základní přenosy pomocí Ethernet nebo pomocí sítě EPSNET.



Obrázek 9 - Sestava PLC Tecomat TC700

Popis obrázku Sestava PLC Tecomat TC700 :
1 – nosný rám
2 – napájecí modul
3 – centrální jednotka
4 – periferní moduly
5 – propojovací kabel mezi rámy
6 – zakončení sběrnice

5.3 Vývojové prostředí Mosaic

Velká většina výrobců PLC má svůj vlastní software k programování automatů pomocí PC. Výjimkou není ani firma Teco a.s., která ovšem svůj software nabízí zdarma. Vývojové prostředí Mosaic poskytuje značný komfort při tvorbě vlastních programů, dokumentování, jejich ladění či diagnostice. Nejméně důležitou vlastností prostředí Mosaic je možnost programovat a ladit PLC bez jeho fyzické přítomnosti (SimPLC). Programová forma je ve shodě s mezinárodní normou IEC EN 61 131-3 v textových jazyce IL (Instruction List = firemní mnemokód) a ST (Structured Text = strukturovaný text) a grafických jazycích LD (Ladder Diagram = reléové schéma) a FBD (Function Block Diagram = funkční bloky). Program se skládá z programové organizační jednotky (POU = Program Organisation Unit). Těmito jednotkami jsou funkce, funkční bloky a nejvyšší jednotkou je program (IL, ST, LD nebo FBD). Tyto jazyky je možno při tvorbě programu mezi sebou kombinovat. Také jsou ve vývojovém prostředí integrovány standardní i uživatelské knihovny funkcí a funkčních bloků. Mosaic také umožňuje komunikaci s řídícím systémem přes sériovou linku, Ethernet či USB. V prostředí je zahrnuta i podpora pro vytáčené připojení přes telefonní nebo GSM modem a i spojení přes Wi-Fi, která umožňuje dálkovou správu [2].

Stručný obsah nástrojů, které obsahuje vývojové prostředí Mosaic:

 PanelMaker

- nástroj na tvorbu dialogů pro operátorské panely ID-07, ID-08 a PLC řady TC500 a TR200
- program pro panel je součástí programu pro PLC

 PanelSim

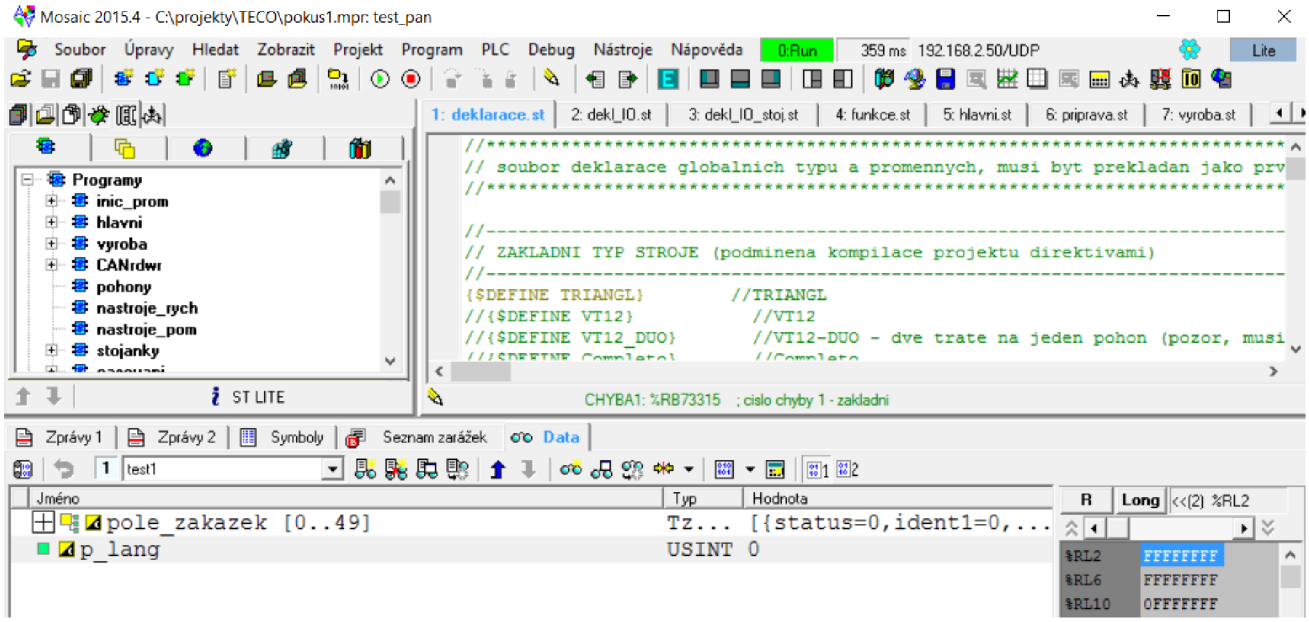
- simulátor operátorských panelů, dovoluje zkoušet dialogy vytvořené PanelMakerem bez připojení skutečného panelu, funguje jak s reálným, tak i simulovaným PLC

 PIDMaker

- nástroj pro ladění a návrh PID regulátorů
- nabízí interaktivní náhled na průběh regulace, usnadňuje správné nastavení parametrů regulátoru a generuje programový kód
- součástí je simulace jednoduchých soustav do třetího řádu s dopravním zpožděním

 GraphMaker

- nástroj pro podporu ladění a diagnostiku řízeného systému umožňuje zobrazení průběhů vybraných proměnných offline i v reálném čase
- dva sledovací kurzory, nastavitelná perioda vzorkování, umožňuje ukládání dat na disk i export do DB programů
- funkce digitálního osciloskopu (16 kanálů) a logického analyzátoru.



Obrázek 10 – Mosaic

5.4 Mezinárodní norma IEC EN 61 131-3

Norma IEC 61 131 pro programovatelné řídící systémy má pět základních částí a představuje souhrn požadavků na moderní řídící systémy. Je nezávislá na konkrétní organizaci či firmě a má širokou mezinárodní podporu. Jednotlivé části normy jsou věnovány jak technickému tak programovému vybavení těchto systémů.

V ČR byly přijaty některé části této normy pod následujícími čísly a názvy: ČSN EN 61 131-1 (Všeobecné informace), ČSN EN 61 131-2 (Požadavky na zařízení a zkoušky), ČSN EN 61 131-3 (Programovací jazyky), ČSN EN 61 131-4 (Podpora uživatelů), ČSN EN 61 131-5 (Komunikace), ČSN EN 61 131-7 (Programování fuzzy řízení). V Evropské unii jsou tyto normy přijaty pod číslem EN IEC 61 131.

Výsledkem normy IEC 61 131 je specifikace syntaxe, sémantiky unifikovaného souboru programovacích jazyků, včetně obecného softwarového modelu a strukturujícího jazyka. Tato norma byla přijata jako směrnice u většiny významných výrobců PLC.

Podrobné informace o mezinárodní normě IEC EN 61131-3 jsou uvedeny v [2].

5.5 Programovací jazyky

Ve vývojové prostředí Mosaic jsou definovány čtyři programovací jazyky. Jejich sémantika i syntaxe je přesně definována. Zvládnutím těchto jazyků se otevírá cesta k používání široké škály řídicích systémů, které jsou na tomto standardu založeny.

Programovací jazyky se dělí do dvou kategorií:

Textové jazyky

IL - Instruction List - jazyk seznamu instrukcí

ST - Structured Text - jazyk strukturovaného textu

Grafické jazyky

LD - Ladder Diagram - jazyk příčkového diagramu (jazyk kontaktních schémat)

FBD - Function Block Diagram - jazyk funkčního blokového schématu

Programovací jazyk volíme na základě typu řešeného problému, na úrovni popisu problému, na struktuře řídicího systému atd. Všechny čtyři základní jazyky (IL, ST, LD a FBD) jsou vzájemně

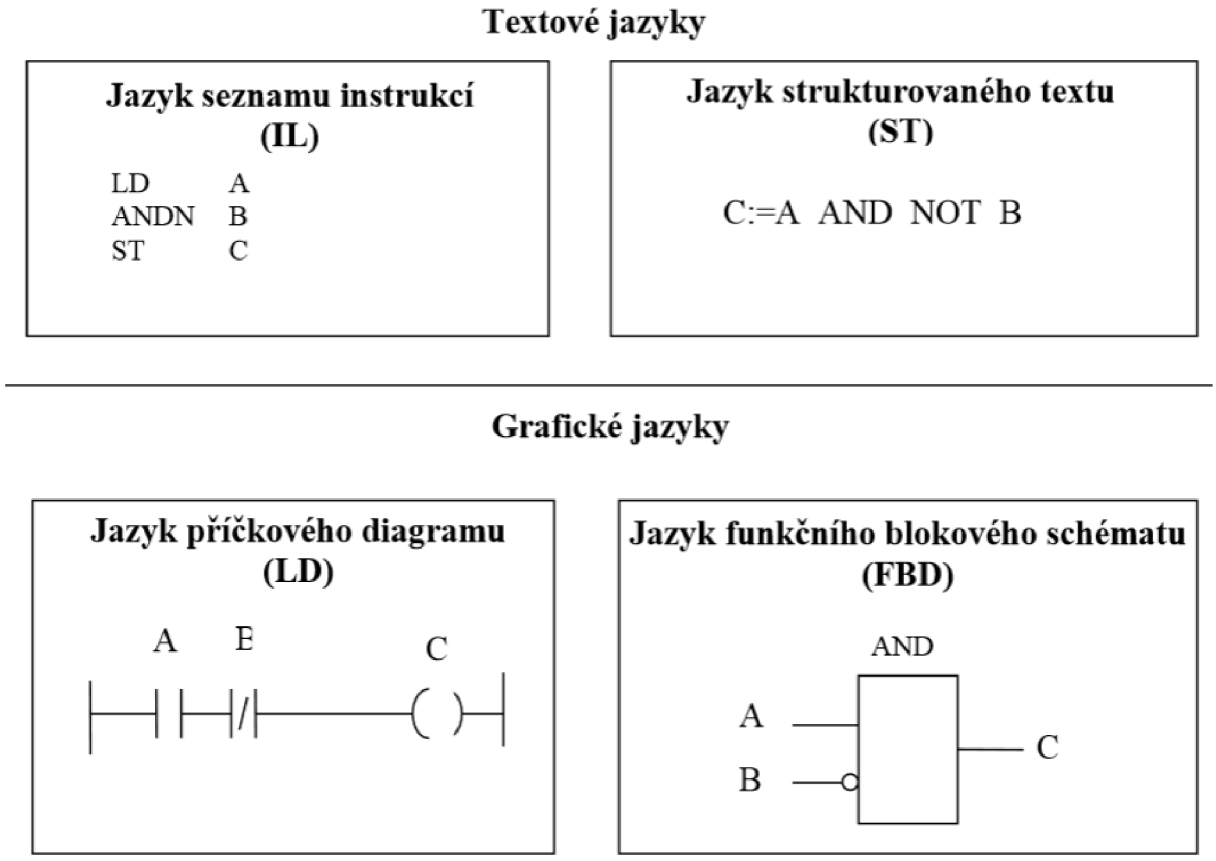
provázány, to znamená, že je možné část programu naprogramovat v jazyce ST a zbytek doprogramovat v jiném jazyce.

Jazyk LD - jazyk příčkového diagramu je založen na grafické reprezentaci reléové logiky.

Jazyk IL (jazyk seznamu instrukcí) - je textový jazyk připomínající assembler.

Jazyk FBD je jazyk funkčního blokového schématu. Vyjadřuje chování funkcí, funkčních bloků a programů. Je to určitý systém prvků (příkazů), které zpracovávají signály.

Jazyk ST - jazyk strukturovaného textu je vyšší programovací jazyk, podobný Pascalu nebo C. Obsahuje všechny podstatné prvky moderního programovacího jazyka. Větvení IF, ELSE, CASE a iterační smyčky FOR, WHILE a REPEAT. Tento jazyk se používá pro definování komplexních funkčních bloků, které mohou být použity v jiném programovacím jazyku.



Obrázek 11 - Ukázka jednotlivým programovacích jazyků

5.6 Společné prvky programovacích jazyků

5.6.1 Typy dat

V rámci společných prvků programovacích jazyků jsou definovány typy dat. Před naprogramováním funkce je nutné definovat typy všech použitých parametrů. Běžné datové typy jsou BOOL, BYTE, WORD, INT, REAL,DATE, TIME, STRING atd.

5.6.2 Proměnné

Ve vývojovém prostředí Mosaic je možné přiřadit proměnné k hardwarovým adresám (vstupům, výstupům) ve zdrojích nebo programech. Tímto způsobem je dosaženo vysokého stupně hardwarové nezávislosti a možnosti opakovaného využití softwaru. Existují dva typy proměnných lokální a globální. Lokální proměnné jsou deklarovány pouze v jedné POU, což znamená, že můžeme použít jména těchto proměnných v jiných POU bez omezení. Globální proměnné mají globální působnost, tedy název proměnné může být v rámci projektu použit pouze jednou.

Pro zajištění správného počátečního stavu stroje je možné přiřadit parametrům počáteční hodnoty při startu.

5.6.3 Programované organizační jednotky

Funkce, funkční bloky a programy jsou v rámci normy IEC 61 131-3 nazývány programové organizační jednotky (Program Organization Units, zkratka POU). POU mohou být dodávány od výrobce řídícího systému nebo mohou být napsány uživatel. Každá POU může volat další POU a při tomto volání může volitelně předávat volané POU parametry.

Existují tři základní typy POU:

- - funkce (function, FUN)
- - funkční blok (function block, FB)
- - program (program, PROG)

5.6.4 Funkce a její syntaxe

Nejjednodušší POU je funkce, jejíž hlavní charakteristikou je to, že výsledek funkce je jednoznačně určen vstupními parametry při volání dané funkce. Funkce může vracet pouze jeden výsledek. IEC 61 131-3 definuje standardní funkce a uživatelem definované funkce. Standardní funkce jsou funkce dodané výrobce řídícího systému např. ABS funkce pro absolutní hodnotu. Jakmile jsou jednou definovány nové uživatelské funkce, mohou být používány opakovaně.

5.6.5 Funkční bloky a syntaxe

Dalším typem POU je funkční blok, který si na rozdíl od funkce, může pamatovat hodnoty z předchozího volání funkčního bloku. Tyto parametry mohou ovlivňovat výsledek funkčního bloku. Hlavním rozdílem mezi funkcí a funkčním blokem je tedy schopnost funkčního bloku vlastnit paměť pro zapamatování hodnot proměnných. Funkční blok může také vracet více než jeden výsledek. Jakmile je jednou funkční blok definován, může být používán opakovaně v daném programu, nebo v jiném programu, i v jiném projektu. Je tedy univerzální a mnohonásobně použitelný. Funkční bloky mohou být zapsány v libovolném z jazyků definovaném v normě.

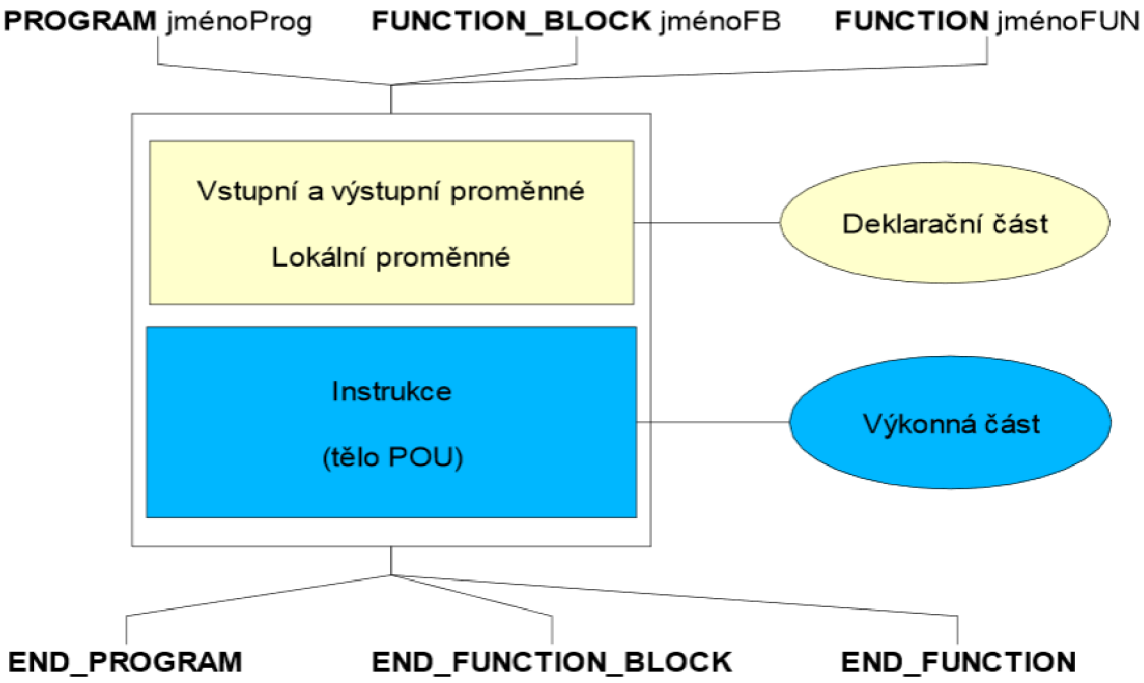
Odvozené funkční bloky jsou založeny na standardních funkčních blocích, ale v rámci pravidel normy je možné vytvářet i zcela nové funkční bloky. Funkční bloky obsahují algoritmy i data, takže mohu zachovávat informaci o minulosti. Mají jasně definované rozhraní a skryté vnitřní proměnné, podobně jako integrovaný obvod nebo černá skříňka.

5.6.6 Program

Program je nejvyšší jednotkou POU, z které je možné volat funkce nebo funkční bloky. Naopak to není možné. Program je definovaný jako logický chod příkazů nutných pro zajištění zamýšleného zpracování signálu.

5.6.7 Struktura programové organizační jednoty (POU)

Každá POU se skládá ze dvou základních částí: deklarační a výkonné, jak je vidět na obrázku 12. V deklarační části POU se definují proměnné potřebné pro činnost POU. Výkonná část pak obsahuje vlastní příkazy pro realizaci algoritmu.



Obrázek 12 – Struktůra POU

5.6.8 Deklarační POU

Deklarační část POU jsou k nadefinování proměnných potřebných pro činnost POU. Každá proměnná je definována jménem proměnné a datovým typem.

Proměnné lze rozdělit na 4 základní deklarační bloky: VAR_INPUT, VAR_OUTPUT, VAR a VAR_TEMP. Každý blok je ukončen END_VAR.

Na Obrázku 10 můžeme vidět, že vše začíná klíčovým slovem PROGRAM a je ukončeno klíčovým slovem END_PROGRAM. Tato klíčová slova vymezují rozsah POU. Za klíčovým slovem PROGRAM je uvedeno jméno POU. Poté následuje deklarační část POU, která obsahuje definici proměnných uvedené mezi VAR_INPUT a END_VAR.

5.6.9 Výkonová část POU

Výkonná část POU obvykle následuje za částí deklarační a obsahuje příkazy a instrukce, které zpracovává centrální jednotkou PLC. V případě použití pouze globálních proměnných nemusí POU obsahovat deklarační část, tedy POU začíná hned výkonovou částí. Z výkonové části POU jsme schopni volat jiné POU[2].

6 NÁSTROJE UMÍSTĚNÉ NA TESTOVACÍ STOLICI

V našem případě byly umístěny na testovací stolici dva nově navržené nástroje používané k výrobě žaluzií, kde hlavním úkolem bylo popsání funkce nástrojů a na základě toho vytvořit jednoduchý program pro odzkoušení těchto nástrojů na testovací stolici viz kapitola Program.

Nástroje:

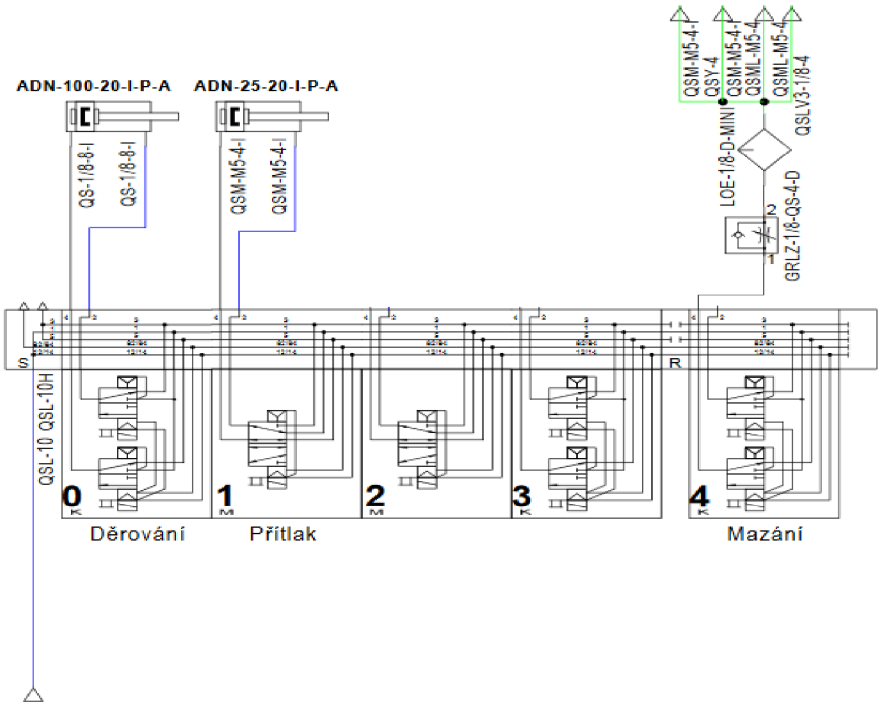
- Děrovací nástroj
- Nýtovací nástroj

6.1 Děrovací nástroj

Děrovací nástroj slouží k vytvoření díry do lamely. Děrování je zajištěno pneumatickými válci, jejichž funkci ovládáme přes ventilový terminál pomocí naprogramovaného PLC. Na obrázku 13 je možné vidět pneumatické schéma tohoto děrovacího nástroje.

V našem případě děrovací nástroj obsahuje dva pneumatické válce různých typů. Pro hlídání polohy pneumatického válce jsou použity dvě magnetické čidla. V tomto případě můžou nastat dvě polohy válce (vysunuto, zasunuto).

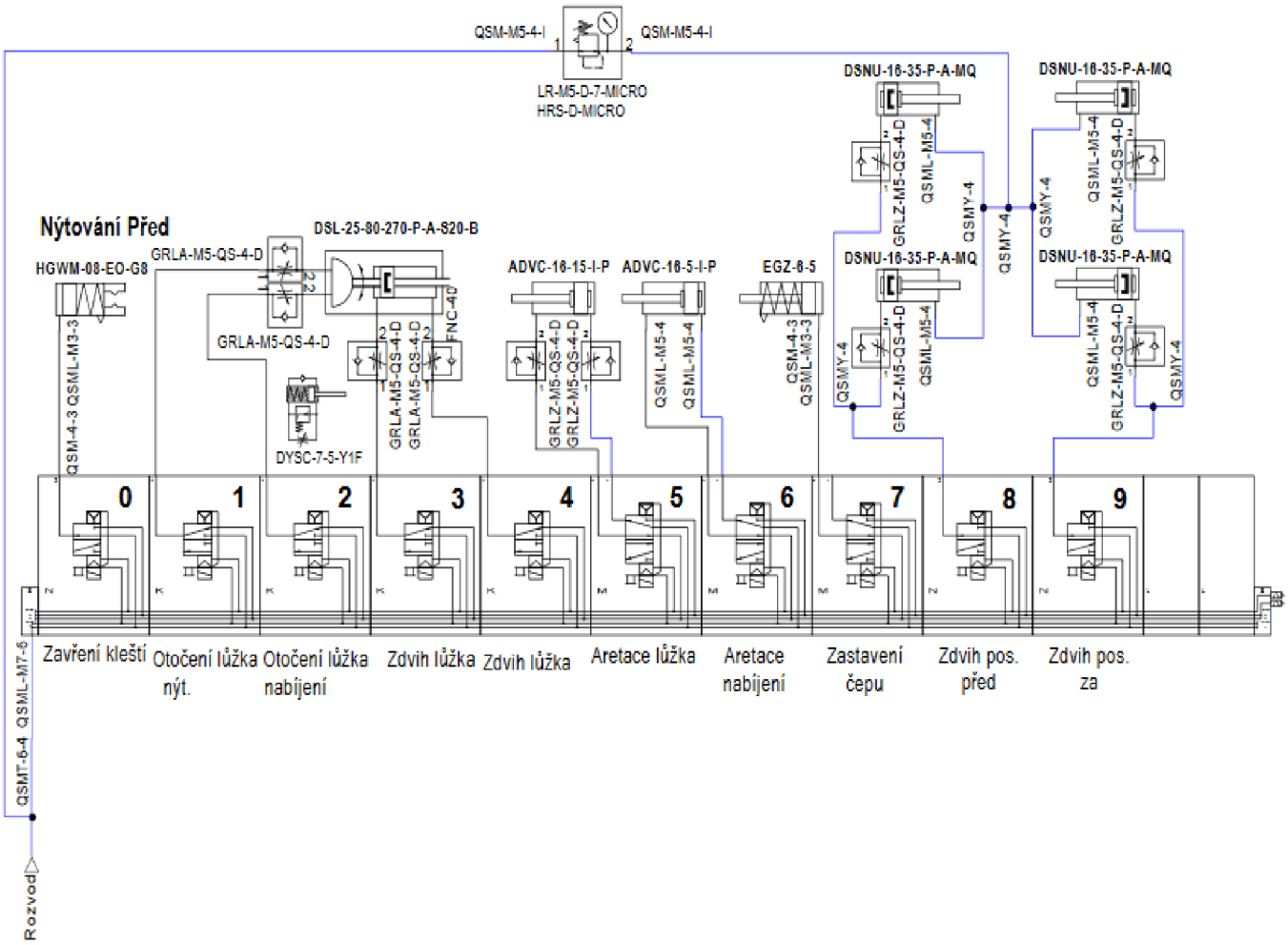
První pneumatický válec zajišťuje funkci přidržení lamely při děrování a je ovládaný pomocí jednoho ventilu. Druhý pneumatický válec slouží k vytvoření díry a jeho polohu ovládáme pomocí dvou ventilů.



Obrázek 13 - Pneumatické schéma děrovacího nástroje

6.2 Nýtování

Nýtovací nástroj slouží k nanýtování koncovek na začátek nebo konec lamely. Funkce nástroje je zajištěna pomocí několika senzorů, pneumatických válců a hydraulického agregátu. Na obrázku 14 je zobrazeno pneumatické schéma ventilového terminálu, který slouží k řízení pneumatických válců, které jsou součástí nýtovacího nástroje.



Obrázek 14 – Pneumatické schéma nýtovacího nástroje

Popis funkce jednotlivých ventilů:

0 - Slouží k ovládání kleští, které zajistí polohu koncovky při nýtování.

1,2 - Řídí otáčení nýtovacího lůžka, buď doprava nebo doleva podle druhu nýtování (na začátku nebo na konci). Při vypnutí obou ventilů zůstává lůžko v prostřední poloze.

3,4 - Pro řízení zdvihu nebo zasunutí lůžka.

5 - Ovládá píst, který slouží pro zajištění stejné polohy nýtovacího lůžka při zdvihu.

6 - K zajištění stále polohy pístu při nabíjení.

7 - ovládá ochranný píst, který slouží pro zastavení koncovek při otáčení lůžka.

8,9 – Slouží k ovládání zdvihových kol, které slouží k posunu lamely.

Proces nýtování začíná nabíráním koncovky. Nýtovací lůžko je ve své středové poloze, která slouží k naložení koncovky. Pro zajištění této polohy je využit pneumatický píst s malým zdvihem (aretace nabíjení). Po naložení koncovky se scvaknou kleštiny, které slouží pro upevnění koncovky v lůžku. Následně se lůžko otočí do polohy nýtování. Po otočení lůžka a dojedí lamely do správné polohy se lůžko vysune a zajistí pomocí dalšího pístu (aretace lůžka). Nýtování je v našem případě prováděno pomocí hydraulického válce, který se následně po zajištění polohy nýtovacího lůžka vysune.

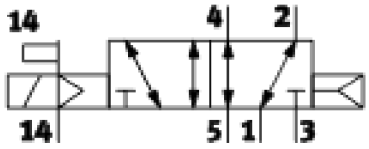
Nýtovací nástroj obsahuje mnoho senzorů, které především zajišťují plnou automatizaci tohoto nástroje. Nýtování obsahuje tři senzory hlídající průchod lamely, dále jeden senzor pro kontrolu zda je koncovka uložena v lůžku. Nýtování obsahuje stejný senzor jako u děrovacího nástroje. Senzor slouží k hlídání chodu pneumatického pístu, přesněji pístu pro zdvih či zasunutí nýtovacího lůžka.

6.2.1 Druhy ovládaných ventilů

Ventilový terminál může obsahovat mnoho druhů různých ventilů v našem případě, ventilový terminál obsahuje tři druhy elektricky ovládaných ventilů od firmy Festo.

6.2.1.1 Ventil M

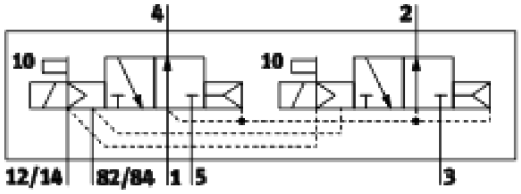
Je monostabilní ventil, tedy ventil, který se bez elektrické signálu vrací do původní polohy v tomto případě pomocí pneumatické pružiny. Tento ventil je zároveň reverzibilní, tedy v kanálu 3 je vyšší tlak, aby se pístnice při elektrickém signálu vysunula, a v kanálu 5 je nižší tlak, aby se pístnice zasunula s úsporou energie. Ventil M je označován jako typ 5/2, kde první číslo označuje počet všech přívodů, výstupu a odfuků ventilu. Druhé číslo určuje možný počet stavů daného ventilu.



Obrázek 15 - Schématická značka Ventil typu M

6.2.1.2 Ventil N

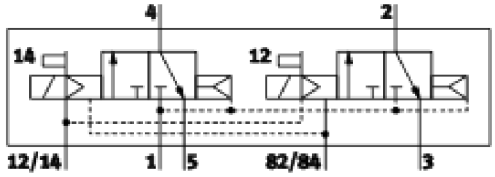
Ventil typu N, je monostabilní ventil, který se vrací do základní polohy pomocí pneumatické pružiny. Tento ventil je typ 3/2, který je v klidové poloze otevřen, pístnice je zasunutá. Při elektrickém signálu se ventil uzavře, což způsobí zasunutí pístnice.



Obrázek 16 - Schématická značka 2xVentil typu N

6.2.1.3 Ventil typu K

Ventil typu K, je monostabilní ventil, vracející se do základní polohy pomocí pneumatické pružiny. Ventil je v základní poloze uzavřený, tedy pístnice je zasunutá. Při elektrickém signálu se ventil otevře, což způsobí vysunutí pístnice.



Obrázek 17 - Schématická značka 2x Ventil typu K

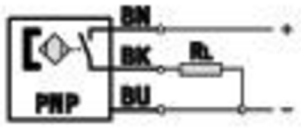
6.2.2 Druhy senzorů u nástrojů

U nástrojů byli použity 3 druhy senzorů:

- Magnetické čidla
- Fotoelektrický senzor
- Laserový fotoelektrický snímač

6.2.2.1 Magnetická čidla

Použité magnetická čidla jsou výrobkem firmy Festo, které jsou konstruovány a optimalizovány speciálně pro válce od firmy Festo, které jsou použité u nástrojů. Magnetické čidla detekují magnetické pole permanentních magnetu umístěných v pístu válce a tím nepřímo měří polohu pístnice. V našem případě je použité magnetické čidlo SMT, které funguje podle magneto-odporovém spínacím principu. Čidla jsou jištěna proti přepólování, odolná proti zkratu a přetížení.



Obrázek 18 - Princip spínaní magnetického čidla SMT

6.2.2.2 Fotoelektrický senzor

Fotoelektrický snímač od firmy Panasonic se skládá ze dvou samostatných jednotek vysílače a přijímače. Kde vysílač vysílá světelný paprsek do přijímače. V případě přerušení světelného paprsku mezi vysílačem a přijímačem detekujeme předmět.

Obrázek 19 - Fotoelektrický snímač od Firmy Panasonic

6.2.2.3 Laserový fotoelektrický snímač

Laserový snímač od firmy Balluff, se používá pro detekování předmětu procházející přes laserový paprsek. Laserový snímač obsahuje přijímač i vysílač jenom v jedné součástce, která má vidlicový tvar. Což nám zmenšuje vzdálenost mezi přijímačem a vysílačem, ale zase zjednodušuje montáž snímače.

Obrázek 20 - Laserový fotoelektrický snímač od firmy Balluff

7 NAPROGRAMOVÁNÍ NÁSTROJŮ

Pro odzkoušení nových nástrojů na testovací stolici bylo zapotřebí naprogramovat činnost nástrojů podle popsaného pneumatického schématu ventilového terminálu. Před samotným programováním chodu nástrojů bylo důležité nadefinovat ovládané vstupy a výstupu PLC a následně vytvořit dvě pomocné funkce (časování, zapnutí hydrauliky).

7.1 Deklarace

Deklarace slouží k vytvoření a nadefinování typů použitých proměnných v programu a k nastavení názvu proměnných pro vstupy a výstupy Modulu. Do těchto modulů zapojujeme jako vstupy senzory nebo centrál stopky a další bezpečnostní prvky a jako výstupy většinou ventilové terminály.

```
////////////                                    /////////////////
////////////// Nastaveni vstupu a vystupu modulů /////////////////
 IB_7302A : TBIN_DI;

  IB_7302A                      AT r0_p1_DI;

  h_s_valce                     AT IB_7302A.DI0;    //otvor nahore
  d_s_valce                     AT IB_7302A.DI1;    //otvor dole
  pridrz_vys_s                  AT IB_7302A.DI2;    //pridrzeni  (pro otvor) zasunuto
  pridrz_zas_s                  AT IB_7302A.DI3;    //pridrzeni  (pro otvor) vysunuto
  in_koncovka_nalozen           AT IB_7302A.DI4;    // senzor pro hlidani nalozeni koncovky
  in_luzko_vy                   AT IB_7302A.DI5;    //  luzka vysunuto senzor
  in_luzko_za                   AT IB_7302A.DI6;    //  luzko zasunuto senzor
  in_lam_pred_nyt               AT IB_7302A.DI7;    // kontrola zda je lamela před nytovani
  in_lam_v_nyt                  AT IB_7302A.DI8;    // kontrola zda je lamela v nytovani
  in_lam_za_nyt                  AT IB_7302A.DI8;    // kontrola zda je lamela za nytovani
  //                            AT IB_7302A.DI10;   //
  //                            AT IB_7302A.DI11;   //
  //                            AT IB_7302A.DI12;   //
  //                            AT IB_7302A.DI13;   //
  //                            AT IB_7302A.DI14;   //
  //                            AT IB_7302A.DI15;   //
```

Zde je zobrazena ukázka deklarace vstupů modulu. V našem případě senzorů umístěných na nástrojích. Deklarace všech vstupu a výstupu je součástí přílohy č.3

7.2 Pomocná funkce časovače

Vývojové prostředí Mosaic neobsahuje integrovanou funkci časovače, proto v našem případě je nezbytné si takovou funkci doplnit, jelikož je nedílnou součástí každého programování v jazyce ST. V Mosaicu jsme schopni odečítat, kolik desítek ms vypršelo od minulého cyklu, tuto hodnotu využíváme pro odečítání hodnoty, kterou časujeme. Časovač jsme vymysleli jako volaný funkční blok, který voláme pomocí pomocného programu vždy při překročení doby více jak 10 ms.

```
FUNCTION_BLOCK Tcasovac  //casovani - odecteni casu
  VAR_IN_OUT
    cas : UINT;
  END_VAR
  IF (cas > odecteni_casovace) THEN
    cas := (cas - odecteni_casovace);
  ELSE
    cas := 0;
  END_IF;
END_FUNCTION_BLOCK
VAR_GLOBAL  //promenne - instance funkcnich bloku
  casovac : Tcasovac;
END_VAR
```

```
PROGRAM casovani
  VAR_TEMP
    poc1 : INT;
  END_VAR
  odecteni_casovace := USINT_TO_UINT(%S3); //pro casovani - kolik desitek
ms vyprselo od minuleho cyklu
  IF (odecteni_casovace > 0) THEN  //vyprsela doba minimalne 10 ms
    casovac(cas := otvor.cas_primazavani);  //odecteni casovace
    casovac(cas := otvor.cas);  //odecteni casovace
    casovac(cas := cas_koncovka);  //odecteni casovace
    casovac(cas := cas_hydraulika);  //odecteni casovace
  END_IF;
END_PROGRAM  //pro casovani
```

7.3 Funkční blok děrování

Naprogramovaný chod nástroje jsme vložili do funkčního bloku, který může být vyvolán při automatickém chodu nebo pomocí zmáčknutého tlačítka zobrazeného na ovládacím panelu.

Správnou posloupnost chodu jsme zajistili pomocí příkazu “case“, který zajišťuje provedení seznamů příkazů, do jejichž mezí patří hodnota selektu (vstupní proměnné). Pro hlídání podmínek potřebných pro správný chod nástroje se používá příkaz IF, který při pravdě či nepravdě umožní provedení daných příkazů.

Prvním krokem funkčního bloku je nadefinování pomocných proměnných a nastavení časovačů. Po nadefinování pomocných proměnných následuje nastavení nástroje do základní polohy, tedy válec přidržení lamely a děrovací válec jsou zasunuté. Pro kontrolu polohy děrovacího válce a přidržení se používají magnetické čidla. V následujícím kroku se kontroluje stav zasunutého

válce. V případě nerozsvíceného horního čidla po vypršení časování se ohlásí chyba programu a ukončí se proces děrování. Pokud horní čidlo svítí, vyšle se signál pro vysunutí přidržení lamely a zapne se časování. V dalším kroku se zkontroluje stav přidržení, jestliže je vysunuté, provede se děrování. Následně se zkontroluje poloha děrovacího válce, pokud je válec vysunutý znamená to, že se provedlo děrování. Po tomto signálu se nastaví nástroj opět do základní polohy.

Zdrojový kód celého funkčního bloku děrování je obsažený v příloze B1.

7.4 Funkce zapnutí hydrauliky

Při procesu nýtování je zapotřebí zaručit zapnutí hydrauliky pro naběhnutí dostatečného tlaku k nýtování. Pro tento stav byla vytvořena funkce, která v případě vypnuté hydrauliky, zapíná časování, tedy zapíná hydrauliku a udržuje hydrauliku zapnutou. Tato funkce nám vrací zpět boolovskou proměnnou nabehly_tlak_hydrauliky, která určuje, zda je dostatečně naběhlý tlak hydrauliky.

```
FUNCTION nabehly_tlak_hydrauliky : BOOL  //vraci, zda bezi  hydraulika
  IF (cas_hydraulika = 0) THEN  //hydraulika vypla
    cas_hydraulika := c_cas_hydraulika;  //zapnuti hydrauliky
    nabehly_tlak_hydrauliky := 0;  //nastaveni vystupu funkce
  ELSE  //hydraulika zapla
    IF (cas_hydraulika <= (c_cas_hydraulika - c_cas_nabehu_hydr)) THEN
        //nabehly jiz tlak hydrauliky
cas_hydraulika:=(c_cas_hydraulika-c_cas_nabehu_hydr);//udrzovani hydrauliky
      nabehly_tlak_hydrauliky := 1;  //nastaveni vystupu funkce
    ELSE  //jeste nenabehly tlak hydrauliky
      nabehly_tlak_hydrauliky := 0;  //nastaveni vystupu funkce
    END_IF;
  END_IF;
END_FUNCTION
Zap_hydraulika:=(cas_hydraulika>0);// zapnuti hydrauliky + podmínka zapnuti
```

7.5 Funkční blok nýtování

Funkční blok nýtování popisuje celý proces nýtování, který má dvě části. První část zajišťuje nabrání koncovky do nýtovacího lůžka při zadání požadavku naložení koncovky.

Tento požadavek může mít tři stavy (pozadavek_koncovka):

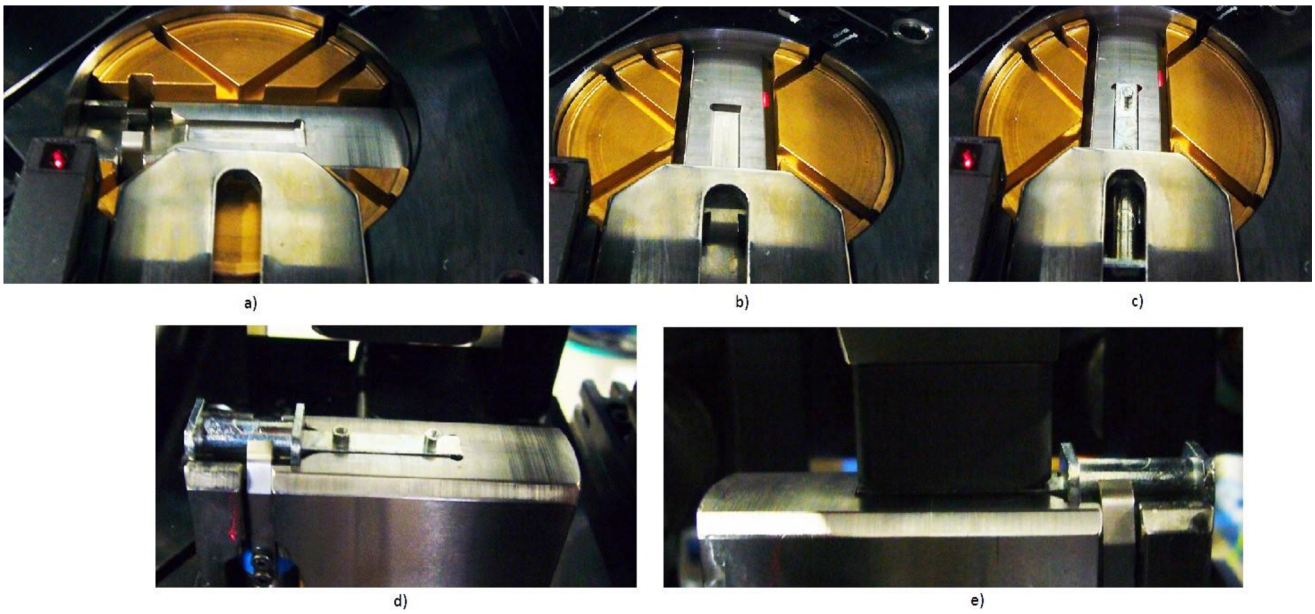
0 - nenabírat koncovku

1 - nabrat koncovku a otočit lůžko doleva

2 - nabrat koncovku a otočit lůžko doprava

Po zadání požadavku naložení koncovky se zkontroluje stav nýtovacího lůžka, při zasunutém nýtovacím lůžku se vyšle signál pro otočení lůžka do polohy nabrání koncovky a nastaví se časovač na dostatečnou hodnotu, za kterou se lůžko otočí. Při dokončení časování a kontrole zda není

nabraná koncovka, odbrzdí se koncovky a rozevřou kleště. V dalším kroku se čeká na nabrání koncovky, v případě nabrání koncovky zabrzdí se koncovky a pomocí kleští se přidrží koncovka v lůžku. Následně se podle požadavku naložení koncovky otočí lůžko doprava nebo doleva. Po otočení lůžka se vyšle signál pro jeho vysunutí, toto vysunutí se kontroluje pomocí magnetického čidla. V případě rozsvíceného horního čidla dojde k vysunutí nýtovací hlavy a tedy k nanýtovaní koncovky. Po nanýtování koncovky se zasune nýtovací hlava a lůžko. Následně se zkontroluje stav koncovky, která by neměla být umístěná v lůžku. Po zkontrolování lůžka se následně vrátí nástroj do základní polohy. Celý zdrojový kód je zobrazený v příloze B2.



Obrázek 21 – Proces nýtování: a) základní poloha nýtování, b) otočení lůžka pro nabíjení koncovky, c) zasunutí koncovky do lůžka d) vysunutí lůžka e) vysunutí nýtovací hlavy

8 OVLÁDACÍ PANEL

Ovládací panel slouží k nastavování funkcí nástrojů umístěných na stroji nebo jako pomocný prvek při seřizování chodu stroje. Naším úkolem bylo vytvoření aplikace ovládacího panelu pro ovládání a seřizování nástrojů testovací stolice pomocí tlačítek zobrazených na dotykové obrazovce.

Pro větší přehlednost panelu bylo rozhodnuto zakoupení většího display a vytvoření nové aplikace pro dotykovou obrazovku. Předchozí ovládací panel obsahoval dotykovou obrazovku zakoupenou od firmy Schneider, která podporovala pouze program od této firmy, který měl omezené možnosti. Proto bylo rozhodnuto zakoupení nového dotykové obrazovky, obsahující operační systém Windows 7. Což nám umožňovalo vytvoření aplikace v operačním systému Windows pro ovládací panel. Tato aplikace byla vytvořena ve vývojovém prostředí Visual Studio.

8.1 Visual studio

Visual Studio je rozšířené vývojové prostředí od firmy Microsoft, určené pro vývoj konzolových a formulářových aplikací v programovacích jazycích C, C++, C# a další. Při použití tohoto vývojového prostředí jsme využili programovací v jazyce C# v kombinaci s knihovnami .NET.

8.2 C#

Jazyk C# je objektový programovací jazyk vyvíjený firmou Microsoft. Program v jazyce C# je kompilátorem překládán do pseodokódu, který je poté spuštěn pomocí CLR(viz níže) Při spuštění programu je však nutné, aby na počítači, který program spouští, byl nainstalován příslušný .NET Framework.[1]

8.3 .NET framework

Aplikace napsané v jazyce C# jsou spuštěny pomocí .NET frameworku. To je vnitřní součást operačního systému Microsoft Windows. .NET obsahuje virtuální spouštěcí mechanismus CLR (common language runtime), který provádí pseodokód, do kterého jsou aplikace v jazyce C# kompilovány. Dále CLR provádí automatické uvolňování již nepoužívané paměti, řízení vyjímek a správu zdrojů. Také obsahuje sadu knihoven, které poskytují řadu již definovaných tříd, včetně tříd pro práci s řetězci, pro práci se soubory, parsování XML, pro vytváření formulářových aplikací atd. [1]

8.4 Komunikace

Prvním úkolem při vytvoření aplikace, bylo zapotřebí zajistit komunikaci mezi naprogramovaným PLC a aplikací běžící na dotykovém zařízení. Pro vytvoření komunikace mezi panelem a naprogramovaným PLC byl použit komunikační server PLCComS vytvořený firmou Teco a.s..

Další krok spočíval v zajištění přijímání a zasílaní dat mezi vytvořenou aplikací a serverem PLCComs, toho jsme dosáhli použitím třídy TCPclient a NetworkStrem.

8.4.1 PLCComS

Komunikační server PLCComS poskytuje TCP/IP spojení mezi klientským zařízením a PLC. Komunikace je řešena pomocí textově orientovaného protokolu typu DOTAZ/ODPOVĚĎ. Klient se tedy serveru dotazuje příkazy na hodnoty proměnných, jejichž jména jsou symbolická a jsou

popsána v public souboru. Server posílá pouze chybová hlášení a hodnoty proměnných, jejichž hodnota se změnila nebo zasílá hodnoty proměnných, na které byl dotázán.

Konfigurace je řešena pomoc *ini* souboru. Konfigurační soubor obsahuje globální nastavení komunikace a nastavení pro konkrétní PLC. Nastavení konfiguračního souboru, pro naši komunikaci je uvedena na obrázku 20.

```
# Configuration file for communication server

[*]

COMM_LOOP_DELAY    = 1 # Delay time in main loop ([1 - 1000]ms)
NET_CONNECT_MAX    = 128   # Maximal number of client connections (Maximum is
1024)
MEM_BLKSIZE        = 4096  # Block size in bytes for transfer PLC memory (Maximum
is 65536)
FFILE_BLKSIZE      = 16384 # Block size in bytes for transfer files (Maximum is
65536)
FFILE_TIMEOUT      = 300   # Time in seconds for holding files in memory (Maximum
is 3600)
FFILE_MAXRECS      = 256   # Number of files stored in memory (Maximum is 1024)
END_LINE_CRLF      = Yes   # End line character (Yes = DOS [\r\n], No = UNIX [\n])
PF_VAR_DISABLED    = Yes   # Default status for variables
DIFF_VAR_ENABLED   = Yes   # Disable or enable sending DIFF: messages
SYNC_VAR_ENABLED   = Yes   # Enable or disable synchronization variables with PLC
while downloading files
LIM_OF_DECIMALS    = No    # Limited precision of decimal numbers to generate
DIFF: messages (Yes = limited)
NUM_OF_DECIMALS    = 10    # Number of decimal digits to be displayed or limited
([0 - 6] REAL, [0 - 15] LREAL)
SCIENT_NOTATION    = No    # Scientific notation for REAL and LREAL (Yes =
scientific [-]d.ddd e[+/-]ddd, No = normal [-]ddd.ddd)

[TC700]

IPADDR         = 192.168.2.50
SERVER_PORT    = 61682
PUBFILE_CRC    = No
PUBFILE_WRITE  = No
PUBFILE_FIXED  = FIXED_TC700.PUB
PUBFILE        = sklad1.pub
```

Obrázek 22 - Nastavení konfiguračního souboru

8.4.2 TcpClient a NetworkStream

K připojení k serveru PLCComs byla vytvořena metoda osahující dvě třídy pro připojení k serveru. První třida TcpClient zajišťuje připojení clienta k zadané IP adrese a portu. Pro zajištění odesílání a přijímání dat ze serveru jsme využili metodu client.GetStream(), která vrací třídu networkStream používající se pro odesílání a přijímání dat.

```
    public void pripojenikplccoms()
            {
                Chybovehlasky1.Hide();
                try
                {client = new TcpClient("192.168.2.46", 61682);//připojení k PLCComs
                 networkStream = client.GetStream(); //nastavení třídy která čte a zapisuje
do PLC           Nacitani();  }
                catch
                {   chybapripojenikPLCComS = true;  }
```

8.5 Čtení a zasílaní dat

Pro čtení příchozích proměnných jsme využili časovač, který nám každou milisekundu kontroluje, zda nepřišla nějaká proměnná z komunikačního serveru. K této kontrole se používá funkce networkStream.DataAvailable, která pří možnosti čtení dat vrátí jedničku. Pro následné čtení dat a jejich dekódování jsou využité dvě funkce, kde funkce networkStream.Read vrací počet přečtených dat, ale i zároveň zapisuje data do pomocné proměnné. Pro dekódování dat se využívá Encoding typu ASCII, který převádí čísla desítkové soustavy na text.

Proměnné ze serveru chodí v určitém tvaru, který musíme upravit a následně uložit do řetězce v případě že došlo více proměnných, aby bylo možné dále sněmi pracovat. K tomu slouží funkce Split.

Tvar příchozí proměnné:

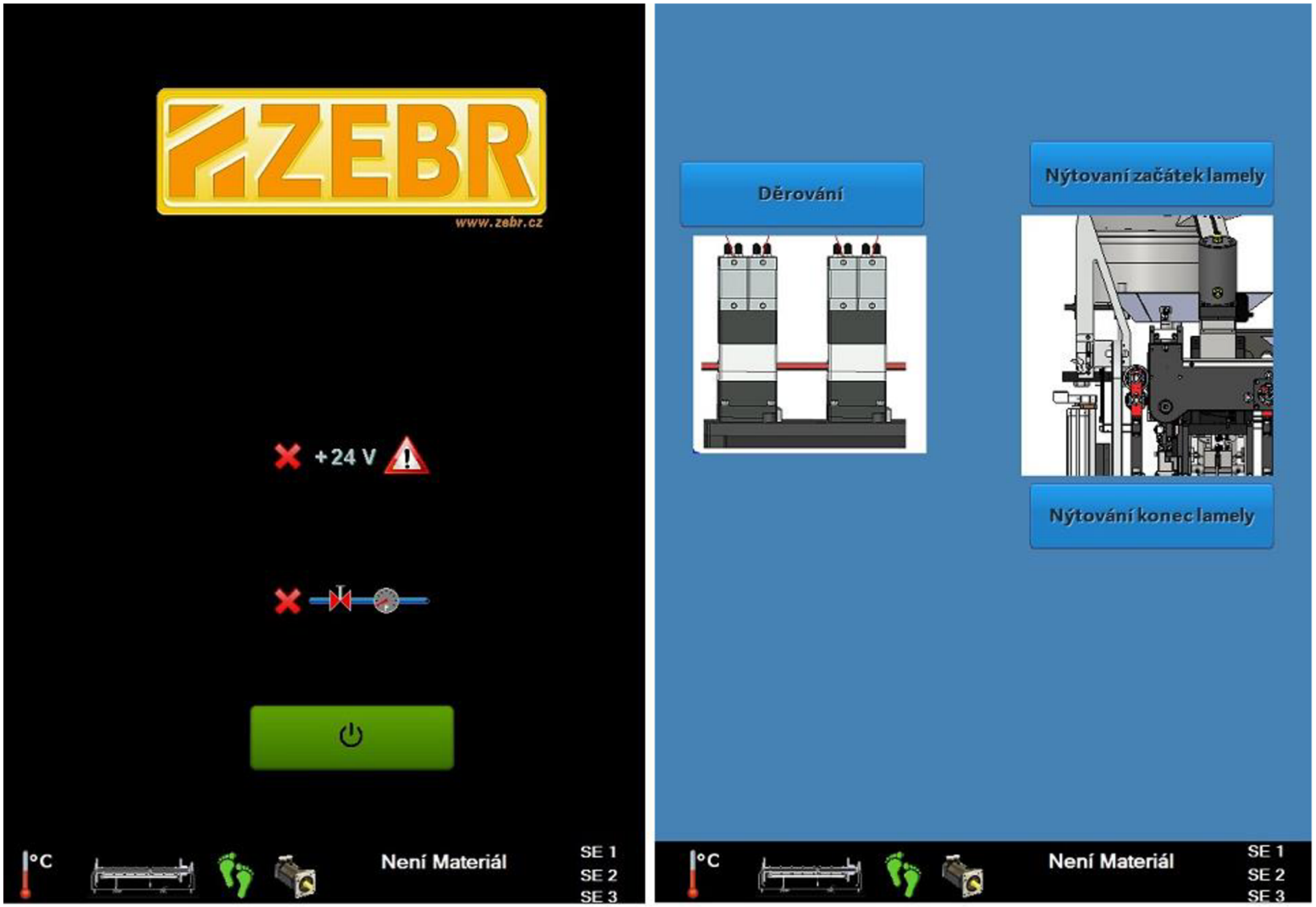

```
    private void timer1_Tick(object sender, EventArgs e) // timer, který slouží pro čtení
            {
    try
                {
                    read = networkStream.DataAvailable; // toto určuje jestli došlo k změné
pokud jo nahodí se 1  (došla proměnná)
                }
                catch
                {
                    read = false;
                }
                if (read)
                {
               int size = networkStream.Read(data, 0, data.Length); // délka dat

     recieved = Encoding.ASCII.GetString(data, 0, size); //encoding dat na string

    string[] source = recieved.Split(new string[] { "SET:", "GET:", "DIFF:", "\r\n" },
StringSplitOptions.RemoveEmptyEntries); // rozdělení zpráv do řetězce
```

Pro odesílání dat byla vytvořena jednoduchá metoda write(). Tyto data mají velice podobný tvar jako příchozí, rozdíl je v druhu příkazu, kdy pří odesílání používáme příkaz **GET:<jméno_proměnné>,<hodnota_proměnné>**.

```
    public void write(string message) // poslani promenych do PLC
            { try
                {
                networkStream.Write(Encoding.ASCII.GetBytes(message), 0, message.Length);
                }
                catch {  }
             }
```



Obrázek 23 - Obrazovky ovládacího panelu testovací stolice

8.6 Použité prvky - jejich využití a nastavení

Na obrázku 21, jsou zobrazeny obrazovky ovládacího panelu testovací stolice. Pro vytvoření těchto obrazovek jsme využili ve formulářové aplikaci 4 prvky:

- Button (tlačítko)
- Label (text)
- PictureBox (Box pro obrázky)
- TabPage (obrazovky)

8.6.1 Button(Tlačítko)

Funkce tlačítka: Při zmáčknutí tlačítka dojde k odeslání proměnné (p_touch_akce) o určité hodnotě do PLC. Abychom nemuseli vytvářet pro každé tlačítko vlastní metodu zamáčknutí. Byla vytvořena obecná metoda, kde se hodnota nastavená v položce Tag tlačítka rovná hodnotě proměnné potřebné pro vyvolání akce:

- Přepnutí obrazovku panelu (tl_inicializace, Tag = 1)
- Zapnutí nýtování (Tag = 2)
- Zapnutí děrování (Tag = 3)

```
    public void Tlacitko_Touch_mousedown(object sender, MouseEventArgs e) //
touch
            {
      Button btn = (Button)sender;

      string[]   btntag   =   btn.Tag.ToString().Split(new   char[]{   ','   },
      StringSplitOptions.RemoveEmptyEntries);

write("SET:pg_touch_akce,"+btntag[0]+"\nSET:pg_touch,1\nSET:pg_touch_pulz,1\n"
);

            } // konec pro mousedown
```

8.6.2 Label

Prvek Label slouží jako informační text, pro uživatele. Kde se daný text načítá z textového souboru, přiloženého k aplikaci, do pomocné proměnné. Labelu nastavíme vlastnost Tag na hodnotu příslušného textu. Pomocí funkce foreach projdeme všechny umístěné Labely a zapíšeme do nich načtený text ze souboru podle hodnoty Tag.

Příklad: Hodnota Tag = 10

Načtený text: 10#Ahoj, 20#Dobrý den.

Text Label = Ahoj

```
     foreach (Control tbx in tb.TabPages[a].Controls) // procházeni prvku v
obrazovce
       {
        if (tbx is Label)
         {
          if (tbx.Tag != null)
           {
             for (int i = 0; i < radky.Length; i++)
               {
                string[]   texty   =   radky[i].Split(new   string[]   {   "#"   },
StringSplitOptions.RemoveEmptyEntries);
```

```
    if (texty[0] == tbx.Tag.ToString())
                    {
                     if (texty.Length == 2)
                  {
                    tbx.Text = texty[1];
                    break;
                  }
                   else tbx.Text = "";
                    }
                    }
                }
            }
          }
        }
```

8.6.3 TabPage (obrazovka)

Do formulářové aplikace můžeme přidat TapPage, pomocí vyšší komponenty TabControl u které lze nastavit přesný počet obrazovek a další možnosti.

Do obrazovek je možné libovolně přidat ostatní prvky popsané v předchozích kapitolách. Pro testovací stolici jsme potřebovali pouze dvě obrazovky (inicializace, hlavní menu). Při prvním spuštění programu nebo při chybě se nám zobrazí obrazovka inicializace. Pro přepnutí obrazovky využijeme tlačítko inicializace, které vyvolá v PLC proces přepnutí obrazovky. Výstupem procesu je změna hodnoty proměnné “pg_picture“.

Pro přepínání obrazovek je vytvořená metoda obrazovka (), která reaguje na změnu hodnoty proměnné “pg_picture“, kde hodnota proměnné určuje zobrazenou obrazovku.

```
    public void obrazovka(string c_obr, TabControl tb)
            {
                foreach (TabPage tp in tb.Controls)
                {

                    if (tp.Tag != null)
                    {
                        if (c_obr == tp.Tag.ToString())
                        {
                      if (tabControl1.SelectedIndex !=Convert.ToInt32(tp.AccessibleName))
                            {
                                tb.SelectTab(Convert.ToInt32(tp.AccessibleName));
                                write("GET:*\n");
                            }
                        }
                    }
                }
            } // nacteni obrazovky
```

8.6.4 Chybové hlášky

Při procesu jakéhokoliv stroje může nastat chyba, proto je důležité tuto chybu zobrazit na ovládacím panelu. Chybové hlášky se zobrazují na samostatném formuláři obsahující dva dynamické Labely, kteří oznamují danou chybu a tlačítko pro potvrzení chyby. Chybová hláška se

zobrazuje při příchodu proměnné pg_picture hodnotě 255. Tedy bylo zapotřebí upravit metodu obrazovku (), pro nastání této možnosti.

```
    public void obrazovka(string c_obr, TabControl tb)
            {
                foreach (TabPage tp in tb.Controls)
                {
                    if (c_obr == "255")  // nastala chyba
                    {
                        Chybovehlasky1.Show();// zobrazeni chyby
                        break;
                    }
                    if (tp.Tag != null)
                    {
                        if (c_obr == tp.Tag.ToString())
                        {
                      if (tabControl1.SelectedIndex !=Convert.ToInt32(tp.AccessibleName))
                            {
                                tb.SelectTab(Convert.ToInt32(tp.AccessibleName));
                                write("GET:*\n");
                            }
                        }
                    }
                }
            } // nacteni obrazovky
```

V tomto případě se nám tedy zobrazí okno s chybou, ale texty jsou prázdné. Proto musíme vytvořit novou metodu, která bude nastavovat texty chybové hláška a to podle dvou proměnných.(pg_chyba1 a pg_chyba2). Chyba obsahuje dvě proměnné jelikož, první proměnná zobrazuje hlavní chybu a druhá proměnná obsahuje doplňující informace. Text labelu jako v předchozím případě načítáme z pomocné proměnné, kde chyby jsou číslované od 10000 a doplňkové chyby od 12000. Při rovnosti hodnoty Tag a čísla chyby se do příslušného labelu zapíše daný text chyby.

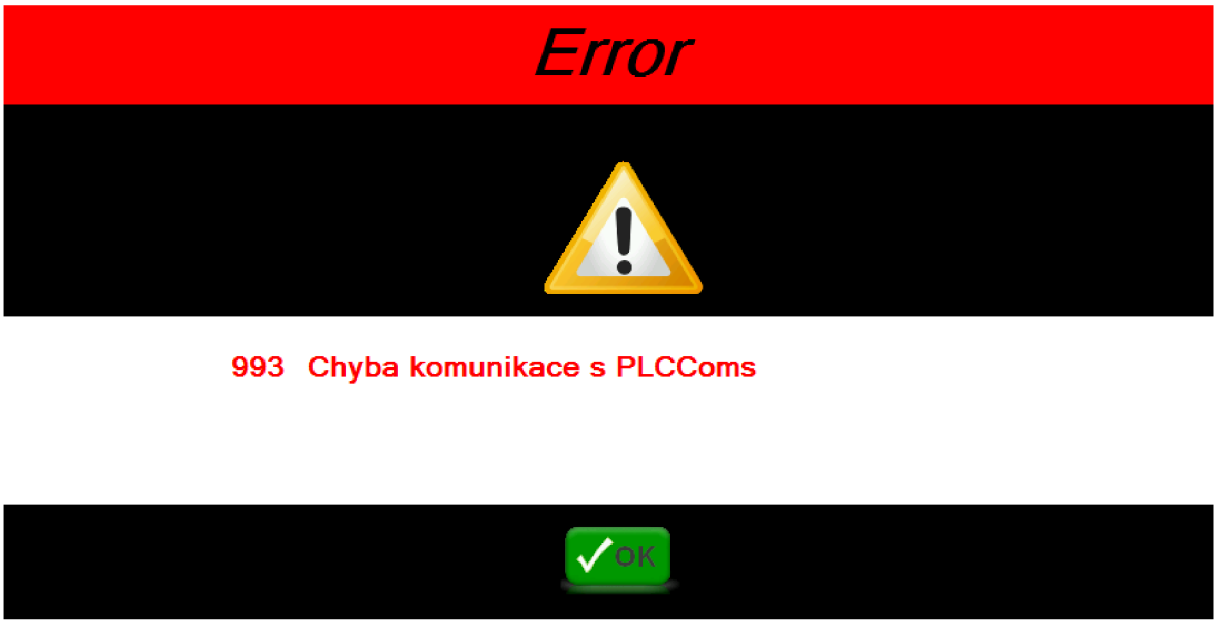
```
    public void chyba(string nazevchyby, string hodnotachyby)
            {
                string hodnotachybyprotext = "";

                if (nazevchyby == "pg_chyba1")
                {
                hodnotachybyprotext = (10000 + Convert.ToInt32(hodnotachyby)).ToString();
                }
                else if (nazevchyby == "pg_chyba2") { hodnotachybyprotext = (12000 +
Convert.ToInt32(hodnotachyby)).ToString(); }
                if (radky.Length > 1)
                    for (int i = 0; i < radky.Length; i++)
                    {
                        string[] texty = radky[i].Split(new string[] { "#" },
StringSplitOptions.RemoveEmptyEntries);

                        switch (nazevchyby)
                        {
                            case "pg_chyba1":
                                if (texty[0] == (hodnotachybyprotext))
                                {
                                    if (texty.Length == 2)
                                    {
```

```
                                Chybovehlasky1.chyba1.Text = texty[1];
                                Chybovehlasky1.cislochyby1.Text = hodnotachyby;
                                break;
                            }
                            else
                            {
                                Chybovehlasky1.chyba1.Text = "";
                                Chybovehlasky1.cislochyby1.Text = hodnotachyby;
                                break;
                            }
                        }
                        break;

                    case "pg_chyba2":
                        if (texty[0] == (hodnotachybyprotext))
                        {
                            if (texty.Length == 2)
                            {
                                Chybovehlasky1.chyba2.Text = texty[1];
                                Chybovehlasky1.cislochyby2.Text = hodnotachyby;
                                break;
                            }
                            else
                            {
                                Chybovehlasky1.chyba2.Text = "";
                                Chybovehlasky1.cislochyby2.Text = hodnotachyby;
                                break;
                            }
                        }
                        break;
                }
            }
        }
```



Obrázek 24 - Zobrazení chybové hlášky

9 ZÁVĚR

Hlavním úkolem bakalářské práce byl návrh testovací stolice. Důvodem vytvoření testovací stolice je neustálý vývoj nových nástrojů pro výrobní stroj, které je potřeba vyzkoušet před namontováním na stroj. Tento návrh byl zpracován v první části práce, kde byly popsány veškeré komponenty testovací stolice.

V druhá část práce byla zaměřena především na možnost odzkoušení nových automatizovaných nástrojů pomocí ovládacího panelu. Pro odzkoušení nástrojů bylo zapotřebí vytvořit funkční program nástrojů a program pro ovládací panel. Program nástrojů byl vytvořen ve vývojovém prostředí Mosaic, jenž bylo v práci popsáno. Ovládací panel byl naprogramovaný v prostředí Visual studio, které slouží pro vytvoření formulářové aplikace běžící v operačním systému Windows. Výsledkem této části práce byl odzkoušený chod nástrojů, jenž bylo možné ovládat pomocí ovládacího panelu nebo počítače.

Hlavním přínosem práce nespočívá jen v navržení univerzálního stroje, umožňující odzkoušení jakýkoliv nástrojů, senzorů, motorů atd., ale i v naprogramované aplikaci ovládacího panelu, která může být v budoucnu využita na ovládání veškeré funkce stroje používaného pro výrobu lamel.

Tato práce mi přinesla mnoho praktických zkušeností v návrhu a následné realizaci projektu. Možnost naučit se pracovat ve dvou nových programovacích jazycích, které byly pro splnění zadání nezbytné. Práce mi umožnila získat větší znalosti v pneumatice, elektronice a praktické zkušenosti v zapojování senzorů, motorů a dalších řídících komponent.

SEZNAM OBRÁZKŮ A TABULEK

SEZNAM SYMBOLŮ A ZKRATEK

Zkratka	Význam
DP	Dolní propust
OSC	Oscilátor
COMP	Komparátor
REF	Referenční napětí
PLC	Programovatelný logický automat
PWM	Pulzně šířková modulace
CP	Centrální jednotka
ST	Strukturovaný text
FBD	Funkční blok
IL	Jazyk reléové schéma
LD	Jazyk kontaktních schéma
POU	Programová organizační jednotka
CLR	Common language runtime
TCP	Transmission control protocol
IP	Internetový protokol

POUŽITÁ LITERATURA

[1] Introduction to the C sharp Language and the .NET Framework. Microsoft. Visual Studio[online]. 2013. Dostupné z : http://msdn.microsoft.com/en-us/library/vstudio/z1zx9t92.aspx

[2] Programování PLC podle normy IEC 61 131-3. *Tecomat* [online]. Praha: Tecomat, 2007 [cit. 2016-03-8]. Dostupné z: http://www.tecomat.com/wpimages/other/DOCS/cze/TXV00321_01_Mosaic_ProgIEC_cz.pdf

[3] PAVLÍK, Ondřej. *Měnič pro malý 3f asynchronní motor* [online]. BRNO, 2011 [cit. 2016-03-15]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=39761. Bakalářská práce. Vedoucí práce Ing. Dalibor Červinka

[4] *Základy elektroniky*. 2009. Praha: CVUT Praha, 2009. ISBN 9788001042366.

[5] Základy elektromagnetické kompatibility (EMC): Způsoby omezování rušení - odrušovací prostředky a elektromagnetické stínění. *Elektrorevue* [online]. 2000, **2000**, 5 [cit. 2016-04-12]. DOI: 2000/41-28.11.2000. Dostupné z: http://www.elektrorevue.cz/clanky/00041/index.html#kap3.3.

[6] BKE. *Spínané napájecí zdroje* [online]. ČR: BKE, 2009 [cit. 2016-04-12]. Dostupné z: http://www.bke.cz/cs/produkty/js-12-xxx-xxx-din

[7] *TGDRIVES* [online]. ČR: omega design, 2006 [cit. 2016-02-15]. Dostupné z: http://www.tgdrives.cz/digitalni-servozesilovace/tga-300/

[8] *TGDRIVES* [online]. ČR: omega design, 2006 [cit. 2016-02-15]. Dostupné z: http://www.tgdrives.cz/digitalni-servozesilovace/akd/

[9] *Hydraulický agregát s motorem ponořeným v kapalině* [online]. Argo-Hytos, 2016 [cit. 2016-05-03]. Dostupné z: http://www.argo-hytos.com/fileadmin/user_upload/Katalog_SPA_01_hc7111_CZ.pdf

SEZNAM PŘILOH

A. Výkresová dokumentace

A1.Testovací stolice – 6113-32-20

A2.Boční stůl – 6113-32-01-00

A3. Středový stůl – 6113-32-02-00

B. Zdrojový kód

B1. Funkční blok děrovacího nástroje

B2. Funkční blok nýtovacího nástroje

CD disk

Roman_Gricman_BP2016.pdf	Bakalářská práce v elektronické podobě.
Vykresova_dokumentace	Složka obsahující výkresovou dokumentaci testovací stolice
Zdrojovy_kod_panel_Visual	Složka obsahující zdrojová kód a naprogramovanou aplikaci pro ovládací panel
Zdrojovy_kod_Mosaic	Složka obsahující zdrojový kód naprogramovaných nástrojů

B. Zdrojový kód

B1. Funkční blok děrovacího nástroje

```
FUNCTION_BLOCK Tfb_nastroj_otvor
VAR_IN_OUT
   otvor : Totvor;
  END_VAR
  casovac(cas := otvor.cas);//nastaveni promenych pro casovac
  casovac(cas:=otvor.cas_primazavani);//nastavuje se otvor.cas_primazavani
  out_primazavani_otvoru := ((otvor.cas_primazavani > 0) // primazavani

  CASE otvor.Scase OF

    1 :  // nazacatku musi byt valec nahore a musi byt zasunuté přidrzeni
      out_valec_nah := 1; // jen pro kontrolu zda je valec nahoře
      out_valec_d := 0; // jen pro kontrolu aby do obou komor nešel vzduch
      ovl_pridrzeni := 0; // jen pro kontrolu zda je zasunute pridrzeni
      otvor.cas := 0;
      otvor.Scase := 2;  //na dalsi akci

    2:    // primazavani po určitém počtu a vysunuti pridrzeni
        IF(pridrz_zas_s)THEN
          IF (pocet_cyklu_primazavani >= 100) THEN
            pocet_cyklu_primazavani := 0;
            otvor.cas_primazavani := c_cas_primazavani;  //primazavani
           END_IF;
         otvor.cas := 200;  // nahozeni casovace
         ovl_pridrzeni := 1; // vysunuti pridrzeni
         otvor.Scase:=3;
  ELSE // neni zasunuté přidrzeni
   chyba(ch1 := 66, ch2 := 91); //nezasunulo se prdrzeni
   otvor.Scase := 255; // do chyby
  END_IF;
     IF NOT (h_s_valce) THEN  // válec neni v horní pozici
```

```
      chyba(ch1 := 66, ch2 := 31);  //"Chyba polohy nastroje otvoru"
           otvor.Scase := 255;  //do chyby
         END_IF

        3 :  // kontrola vysunuti pridrz , + pohyb valce dolu
      IF(pridrz_vys_s) THEN // senzor vysunuti přidrzeni
             out_valec_d := 1;   // válec jede dolu
             out_valec_nah := 0; //pro vyfouknutí vzduchu v horní pozici pístu
             otvor.cas := 200;  //nahozeni casovace
             otvor.Scase := 4;  //na dalsi akci
      ELSE // nevysunul se pridrzeni
        IF (otvor.cas = 0) THEN  //vyprsel cas a nevysunulo se pridrzeni = chyba
              chyba(ch1 := 66, ch2 := 90); // Nevysunulo se pridrzeni
               otvor.Scase := 255;  //do chyby
        END_IF;
      END_IF;

        4 :  //otvor už je vytvořený, nahoru vyjede válec
          IF (d_s_valce) THEN  //dojeto dolu (senzor)
            out_valec_d := 0; // pro vyfouknutí vzduchu (uzavření ventilu)
            out_valec_nah := 1; // pohyb pístu nahoru
            otvor.cas := 200;  //[10 ms] nahozeni casovace
            otvor.Scase := 5;  //na dalsi akci
          ELSE
        IF (otvor.cas = 0) THEN  //vyprsel cas - chyba
               chyba(ch1 := 66, ch2 := 30);  //"-Nedojel dolu"
               otvor.Scase := 255;  //do chyby
        END_IF;
          END_IF;
        5 :  // kontrola dojeti nahoru válce a zasunutí pridrzeni
          IF (h_s_valce) THEN  //dojeto válec nahoře
            ovl_pridrz := 0; // zasunuti pridrzeni
             otvor.cas = 100; // nahozeni casovac
   pocet_cyklu_primazavani := pocet_cyklu_primazavani + 1; //vykonaný cyklus
přičtem
            otvor.Scase := 6;  //na dalsi akci
          ELSE
```

```
            IF (otvor.cas = 0) THEN  //vyprsel cas - chyba
                chyba(ch1 := 66, ch2 := 31);  //"Chyba polohy nastroje otvoru"
/ "-Nedojel nahoru"
                otvor.Scase := 255;  //do chyby
            END_IF;
          END_IF;


        6 :  //pridzeni zpet
       IF(pridrz_zas_s) THEN    // senzor kontrolu zasunuti pridrzeni
            otvor.cas := 50;  // nahozeni casovace
            otvor.Scase := 99;  //ukonceni
       ELSE
         IF (otvor.cas = 0) THEN  //vyprsel cas + neni zasunute pridrzeni = chyba
              chyba(ch1 := 66, ch2 := 91); // chyba polohy nastoje otvoru //
Nezasunulo se pridrzeni
                otvor.Scase := 255;  //do chyby
         END_IF;
       END_IF;


         99 :  //ukonceni
           IF (otvor.cas = 0) THEN  //vyprsel cas
              out_valec_nah := 1; // pro kontrolu zajistime polohu nahore
             out_valec_d := 0;
           END_IF;


       END_CASE;
    END_FUNCTION_BLOCK
    VAR_GLOBAL
      fb_nastroj_otvor : Tfb_nastroj_otvor;
    END_VAR
```

B2. Funkční blok nýtovacího nástroje

```
FUNCTION_BLOCK Tfb_nytovacka
  VAR_TEMP
    poc1 : INT;
    pom_pozad_naloz_konc : USINT;
    pom_kusu_pridani : UINT;
  END_VAR

//-------------------------------------------------------------------------
------
//
// -pozadavek_koncovka = pozadavek nanytovani (pokud koncovka neni na luzku,
tak se nabere) 0 ne , 1 vlevo 2 vpravo
//
//-------------------------------------------------------------------------
------
        out_aretace_luzka = (out_nytov_luzko_vy AND in_luzko_vy);
        out_ofuk_nyt = (cas_ofuk > 0);

CASE koncovka OF
    0 :
          out_nytov_luzko_za = 1; // luzko je zasunute
   IF(pozadek_koncovka > 0) THEN // pokud se má nabrat koncovka
      IF NOT(in_koncovka_nalozena) THEN // neni  v lůžku , můžeme naložit
         koncovka = 10;
      ELSE
      chyba(ch1 := 50, ch2 := 80); // v nastroji zustala stara koncovka
       koncovka := 255;
      END_IF;
   ELSE
      IF NOT(in_koncovka_nalozena) THEN // koncovka je nalozena
      koncovka = 1;
      END_IF;
   END_IF;

    1: // lužko nahoru
```

```
       IF (in_koncovka_nalozena) THEN  //neni nabrana koncovka
                pozadek_koncovka :=  minuly_pozadavek_nalozeni_koncovky;
                koncovka_za := 50;  //na dalsi akci
        END_IF;
      IF (koncovka_za <> 50) THEN  //pokud se nema nejdrive nabirat koncovka
            IF (nabehly_tlak_hydrauliky()) THEN    //nabehly tlak hydrauliky
BOOL funkce pokud pravda muzem pokracovat

            out_nytov_luzko_vy = 1; // vysunutí lužka
            out_nytov_luzko_zas = 0; // odfouknutí vzduchu
            cas_koncovka := cas_vysunuti_luzka;  // nahozeni casovace
        IF ((nalozeni_koncovky = 1)) THEN  //pridzeni pri nytovani vlevo
            out_pritlacne_kola_vystup := 1; // pridrzeni na vystupu
        ELSE
          IF ((nalozeni_koncovky = 2)) THEN  // pridzeni pri nytovani vpravo

            out_pritlacne_kola_vstup := 1;  //pridrzeni lamely na zacatku
          END_IF;
        END_IF;
            koncovka = 2;
        END_IF;
       END_IF;

       2: // kontrola luzka nahoře pokud jo vysunutí nýtovací hlavy
      IF(in_luzko_vy) THEN // luzko je vysunuté
        out_nyt_hlava_d = 1; nytovaci hlava dolu
        out_kleste_koncovky := 1;  //otevreni klesti
        cas_koncovka = cas_koncovka_nyt;
        koncovka = 3; // další akce
      ELSE
       IF(cas_koncovka = 0) THEN
        IF NOT(in_luzko_vy) THEN // nevysunulo se luzko + dosel cas = chyba
          koncovka = 255;
            chyba(ch1 := 63, ch2 := 97);  //"-Nevysunulo se luzko"
        END_IF;
       END_IF;
      END_IF;
```

```
  3: //vracení do původnich poloh
IF(cas_koncovka = 0)THEN // konec nytovani
  out_nyt_hlava_d = 0; nytovaci hlava do puvodni polohy
  out_pritlacne_kola_vystup := 0;
  out_pritlacne_kola_vstup := 0;
  out_nytov_luzko_vy = 0;
  out_nytov_luzko_zas = 1;
  cas_koncovka = 200; // nahozeni casovače
  koncovka = 4;
 END_IF;

  4: // kontrola luzka
IF(in_luzko_za)THEN
  out_nyt_luzko_za = 1;
  IF(je_vyroba)THEN
    IF(nalozeni_koncovky = 1) THEN // vlevo
      pozadek_koncovka= 1;
      koncovka = 0;
     ELSE // vpravo
        pozadek_koncovka= 2;
        koncovka = 0; // novy plán
    END_IF;
  END_IF;
ELSE
 IF(cas_koncovka = 0) THEN
   IF NOT(in_luzko_za) THEN // dosel cas a luzko nezasunuto = chyba
    chyba(ch1 := 63, ch2 := 98);  //"-Nezasunulo se luzko"
     koncovka = 255; // chyba
   END_IF;
 END_IF;



10:
    IF ((pozadavek_koncovka = 1)) THEN  //koncovka vlevo
      nalozeni_koncovky := 1; // vlevo pomocna promenna
```

```
    ELSE // koncovka vpravo
IF ((pozadavek_koncovka = 2)) THEN
nalozeni_koncovky := 2; // pravo pomocna promenna
    END_IF;
    out_nytov_luzko_vy := 0;  //vychozi klidova poloha
    out_nytov_luzko_za := 1; // zasunuté lůžko, provede se kontrola
    out_brzda_koncovky := 0;  //zabrzdene koncovky
    out_kleste_koncovky := 0;  //zavrene klesti
    out_pretoc_luzka_nyt := 0;  // poloha luzka je 0 - vlevo, 1 - vpravo
    out_pretoc_luzka_nab := 0;
    cas_ofuk := 250;  // ofuk koncovek
    cas_koncovka := 200;  //50;  //[10 ms] nahozeni casovace
    koncovka := 11;  //na dalsi akci

 11 :  // kontrola zasunuti luzka a otoceni luzka do nabijeni
    IF (in_luzko_za) THEN  //luzko zasunuto

      out_pretoc_luzka_nab := 1; // přesunutí do nabijeni
      out_aretace_nab= 1;
      cas_koncovka_za := 70;  //[10 ms] nahozeni casovace
      koncovka_za := 12;  //na dalsi akci
    ELSE
      IF (cas_koncovka_za = 0) THEN  // luzko neni zasunute = chyba
        chyba(ch1 := 63, ch2 := 98);  // "-Nezasunulo se luzko"
      END_IF;
    END_IF;

  12 :
    IF (cas_koncovka = 0) THEN  //vyprsel cas
      cas_ofuk := 250;  //dofouknuti koncovek okamzite
      cas_koncovka := 120;  //[10 ms] nahozeni casovace
      IF (in_koncovka_nalozena) THEN  //koncovka neni na luzku
        out_brzda_koncovky := 1;  //odbrzdeni koncovky
        out_kleste_koncovky := 1;  //otevreni klesti
      END_IF;
      koncovka_za := 13;  //na dalsi akci
```

```
        END_IF;

      13: // kontrola koncovky v luzku
         IF (cas_koncovka = 0) THEN  //vyprsel cas
          IF (in_koncovka_nalozena) THEN  //koncovka neni na luzku
               chyba(ch1 := 62, ch2 := 81);  // Koncovka neni v nastroji"
           END_IF;
         ELSE
          IF NOT(in_koncovka_nalozena) THEN
        out_aretace_nab = 0; // zruseni aretace
         out_kleste_koncovky := 0;  //zavreni klesti
        out_brzda_koncovky := 0;  //zabrzdeni koncovky
        pozadek_koncovka = 0; // koncovka je nalozena
        koncovka = 14;
          END_IF;
       END_IF;

      14 :  // pretoceni luzka do nytovani
       IF (cas_koncovka_za = 0) THEN  //vyprsel cas
         out_nytov_luzko_za := 1;   // porad zasunuté luzko,(mohli otáčet)

    IF(nalozeni_koncovky = 1)THEN // otoceni vlevo
     out_pretoc_luzka_nab = 0;
     out_pretoc_luzka_nyt := 0;
          cas_koncovka := 50;
   ELSE // otoceni vpravo
     out_pretoc_luzka_nab = 0;
     out_pretoc_luzka_nyt := 1;
     cas_koncovka := 50;
   END_IF;
         koncovka := 99;  //na dalsi akci
      END_IF;


99 :  //ukonceni
      IF (cas_koncovka_za = 0) THEN  //vyprsel cas
```

```
            koncovka := 0;  // nazacatek
        END_IF;
    END_CASE;


END_FUNCTION_BLOCK;


VAR_GLOBAL  //promenna - instance funkcniho bloku
  fb_nytovacka : Tfb_nytovacka;
END_VAR
```