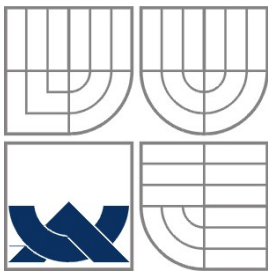


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY

FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

NÁVRH A REALIZACE LABORATORNÍ ÚLOHY ŘÍZENÍ ELEKTRO-PNEUMATICKÉHO MANIPULÁTORU FESTO

REALIZATION OF CONTROL SYSTEM FOR ELECTRO-PNEUMATIC MANIPULATOR FESTO

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE **BC. JAKUB JURNÍČEK**
AUTHOR

VEDOUCÍ PRÁCE **ING. STANISLAV VĚCHET, PH. D.**
SUPERVISOR

BRNO 2013

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2012/2013

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Jakub Jurníček

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

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

Návrh a realizace laboratorní úlohy řízení elektro-pneumatického manipulátoru FESTO

v anglickém jazyce:

Realization of control system for electro-pneumatic manipulator FESTO

Stručná charakteristika problematiky úkolu:

Hlavním cílem práce je návrh řídicího systému pro 3osý manipulátor FESTO. Manipulátor je osazen pneumatickými lineárními pohony a elektrickými snímači polohy. Požadovaný řídicí systém bude navržen s využitím programovatelných automatů. Systém by měl být dostatečně otevřený pro možnost připojení externího řídicího prvku např. od NI LabVIEW.

Cíle diplomové práce:

1) Seznamte se s možnostmi řízení průmyslových manipulátorů.

2) Podrobně prostudujte vlastnosti dostupného manipulátoru.

3) Navrhněte adekvátní řídicí systém.

4) Navržený řídicí systém prakticky realizujte a otestujte na vybrané laboratorní úloze.

Seznam odborné literatury:

[1] Kopáček, J.: Pneumatické mechanismy

[2] Balátě, J.: Technické prostředky automatického řízení

Vedoucí diplomové práce: Ing. Stanislav Věchet, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2012/2013.

V Brně, dne 15.11.2012

L.S.

_____	_____
Ing. Jan Roupec, Ph.D.	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c.
Ředitel ústavu	Děkan fakulty

ABSTRAKT

Práce se zabývá problematikou pneumatických pohonů, fyzikálními vlastnostmi stlačeného vzduchu a řízení pomocí programovatelných automatů. V práci jsou dále probírány základy programování v LabVIEW a se stručným přehledem jednotlivých prvků manipulátoru FESTO. Závěr práce je věnován návrhu řídícího systému pro 3osý manipulátor FESTO v programu NI LabVIEW.

ABSTRACT

Presented thesis deals with pneumatic actuators, PLC control systems and physical properties of compressed air. The thesis introducing short view of pneumatic components of FESTO manipulator and LabVIEW programming. Experimental results are presented on three axis FESTO manipultor programmed by NI LabVIEW.

KLÍČOVÁ SLOVA

Festo, LabVIEW, manipulátor, PLC, pneumatické pohony

KEYWORDS

Festo, LabVIEW, manipulator, PLC, pneumatic actuators

PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že jsem uvedenou práci zpracoval samostatně pod vedením Ing. Stanislava Věcheta, Ph.D. a použil jsem pouze literaturu uvedenou v bibliografii.

Květen 2013

Bc. Jakub Jurníček ….....

BIBLIOGRAFICKÁ CITACE

JURNÍČEK, J. *Návrh a realizace laboratorní úlohy řízení elektro-pneumatického manipulátoru FESTO.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013. 65 s. Vedoucí diplomové práce Ing. Stanislav Věchet, Ph.D..

PODĚKOVÁNÍ

Touto cestou chci poděkovat vedoucímu diplomové práce Ing. Stanislavu Věchetovi, Ph. D. za odborné vedení, cenné rady a podněty při zpracování této práce.

Poděkování také patří Ing. Pavlu Houškovi Ph. D. za profesionální pomoc při práci s programem LabVIEW.

Obsah:

SEZNAM POUŽITÝCH SYMBOLŮ

Schématické značení ovládání ventilů

- Lidskou silou


tlačítko


hřibové tlačítko


páka


páka s aretací polohy


pedál

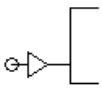
- Mechanicky


dřík

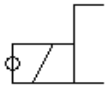

dřík s kladkou

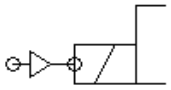

vratná pružina*

- Tlakem média


tlak média

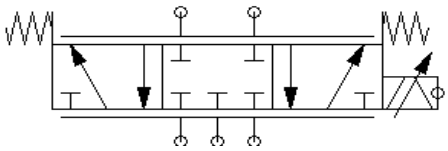
- Elektromagneticky


přímé ovládání


nepřímé ovládání

*) Nejedná se o samostatný ovládací prvek, ale o mechanismus vestavěný do těla ventilu

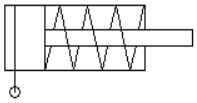
Proporcionální ventil



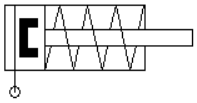
ventil 5/3, přímo ovládaný ve střední poloze uzavřený s regulovatelným napájením

Schematické značky lineárních pohonů

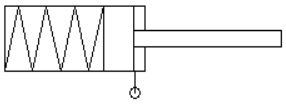
- Jednočinné pohony



jednočinný válec pružinou zasunutý

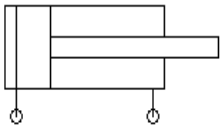


jednočinný válec pružinou zasunutý s magnetem

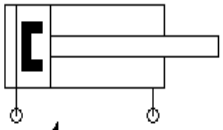


jednočinný válec pružinou vysunutý

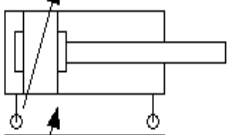
- Dvojčinné pohony



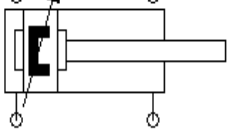
dvojčinný válec



dvojčinný válec s magnetem



dvojčinný válec s nastavitelným tlumením

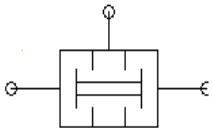


dvojčinný válec s nastavitelným tlumením a magnetem

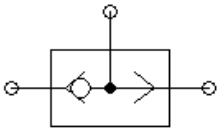
Ostatní pneumatické prvky

zdroj stlačeného vzduchu

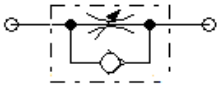
vzdušník



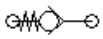
logický ventil AND



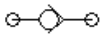
logický ventil OR



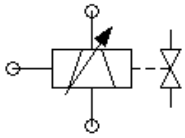
škrtící ventil v jednom směru



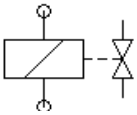
zpětný ventil s přítlačnou pružinou



zpětný ventil



elektromagnet ovládající proporcionální ventil



elektromagnet ovládající ventil

1 ÚVOD

Stlačený vzduch je vhodné médium pro malé a střední síly s pohybem o vysoké frekvenci a četnosti opakování. Elektronika je s výhodou nasazována na náročné výpočetní procesy a snadný přenos dat. Spojením vzniká spolehlivý a přesný elektropneumatický systém vhodný i pro ty nejnáročnější úkoly. Řízení pneumatických systému se převážně provádí pomocí programovatelných automatů. Zaručují univerzální použití, úsporu místa, kabeláže a celkových nákladů, jak pořizovacích, tak i provozních. Vhodným zpracováním vstupních a výstupních signálů lze řešit i ty nejobtížnější úlohy pomocí jednoho z normovaných programovacích jazyků.

Při výběru pneumatických válců vycházíme z hlavních parametrů jako jsou zdvih, vysouvací popř. zasouvací síla a pracovní tlak. Nejrozšířenějšími pneumatickými pohony různých konstrukcí a provedení slouží k realizaci lineárních pohybů. Většina konstrukcí vychází ze dvou základních provedení: Jednočinných s přívodem vzduchu pouze na jedné straně a dvojčinných s přívodem na obou stranách válce. Nepostradatelné prvky pneumatických obvodů tvoří ventily. Ventily určují směr toku média přestavěním do příslušné polohy. Volba ovládání ventilů je odvozena od složitosti realizovaného obvodu. I pro středně obtížné obvody je nejvhodnější použít elektromagnetické ovládání s připojením na některý z programovacích automatů. Máme tak zaručeně usnadněnou editaci, popřípadě rozšíření o další úlohy.

Cílem této diplomové práce je návrh univerzálního řídicího systému pro pneumatický manipulátor, který slouží pro praktickou laboratorní výuku. Pneumatický manipulátor je osazen pneumatickými prvky od fy. FESTO. Použitým pneumatickým a elektro-pneumatickým prvkům je věnována kapitola 7. Pro návrh řídicího systému bylo využito prostředí NI LabVIEW od firmy National Instruments, jež je vhodné pro svoji univerzálnost, otevřenost a snadné použití. Pneumatické prvky od fy. FESTO byly zvoleny pro svou didaktickou podstatu, která usnadňuje pochopení principu ovládání pneumatických prvků. Práci v tomto prostředí je věnována kapitola 6. Dílčím cílem bylo také navrhnout rozhraní mezi využitým hardware a vlastním programovým prostředím tak, aby toto rozhraní umožňovalo snadnější tvorbu vlastních řídicích úloh v budoucnu. Pomocí tohoto rozhraní je možné další pokračování práce na manipulátoru s přispěním jiných řídících prvků např. programovatelných automatů s možností otevřeného programování v jednom z jazyků zmiňovaných v této práci v kapitole 4.

2 PNEUMATICKÉ SYSTÉMY

2.1 Stlačený vzduch

Stlačený vzduch jako nositel energie lze považovat za nejstarší formu pro zvýšení fyzické výkonosti. Již ve starověku byl použit stlačený vzduch k pohánění pneumatického praku. Řecké slovo "pneuma" znamenající duše nebo také vítr, dalo název celému oboru. Dnes s oborem pneumatika lze studovat pohyby vzduchu popř. procesy, které ve vzduchu probíhají.

I když je tedy stlačený vzduch znám od starověku, trvalo celá staletí, než jsme jej začali systematicky využívat. Nedostatečná znalost fyzikálních zákonů plynů nás odsoudila k průmyslovému nasazení stlačeného vzduchu až na druhou polovinu minulého století. A i přes průmyslovou revoluci nedostal takový prostor pro uplatnění, jaký by si zasloužil. Plné nasazení lze považovat až v posledních desetiletích a to především stále častějšímu nasazování automatizace v technologických procesech. A i když stále o pohybu vzduchu nevíme vše, snažíme se naše vědomosti zvětšovat a technologie na zpracování a využití vzduchu stále zdokonalovat. V současné době si jen těžko lze představit nějaký průmyslový proces bez stlačeného vzduchu.

Faktory, které stojí za rychlém rozvoji stlačeného vzduchu je hned několik. Přibližme si je:

Dostupnost – vzduch vhodný ke stlačení je všude kolem nás v neomezeném množství.

Doprava – snadná doprava pomocí potrubí beze ztrát, kde není potřeba řešit problém zpětného vedení jako je tomu u hydrauliky.

Akumulace – není potřeba neustálé spotřeby, lze uskladnit v tlakových nádobách, stejně tak lze snadno přenášet v tlakových lahvích

Bezpečnost – nezpůsobuje žádné jiskření, tudíž je možnost je bezpečně používat i v prostředích s nebezpečím výbuchu nebo požáru

Čistota – jedna z velkých předností v dnešní environmentální době. Při výrobě nebo spotřebě stlačeného vzduchu nedochází k žádnému znečišťování okolního prostředí. Lze jej tedy použít např. v potravinářství, dřevozpracujícím průmyslu apod.

Jednoduchost – výkonové prvky jsou velmi jednoduché, neobsahují žádné složité pohyblivé prvky, které bývají náchylné k rychlému opotřebení a pak k následné nefunkčnosti. Z jednoduchosti vyplývá i konečná cena produktů pracujících na stlačený vzduch.

Rychlost – rychlost 1 až 2 ms^{-1} řadí stlačený vzduch mezi velmi rychlá pracovní media.

Přetížitelnost – v neposlední řadě velká přednost je v možnost přetížení. Pneumatický prvek se jen zastaví bez jeho následného poškození.

Samozřejmě má i své nevýhody:

Hlučnost – jak při výrobě, tak i při spotřebě (odfuk vzduchu do atmosféry) vzniká nepříjemný hluk. Ten je možné částečně odstranit tlumičem výfuku.

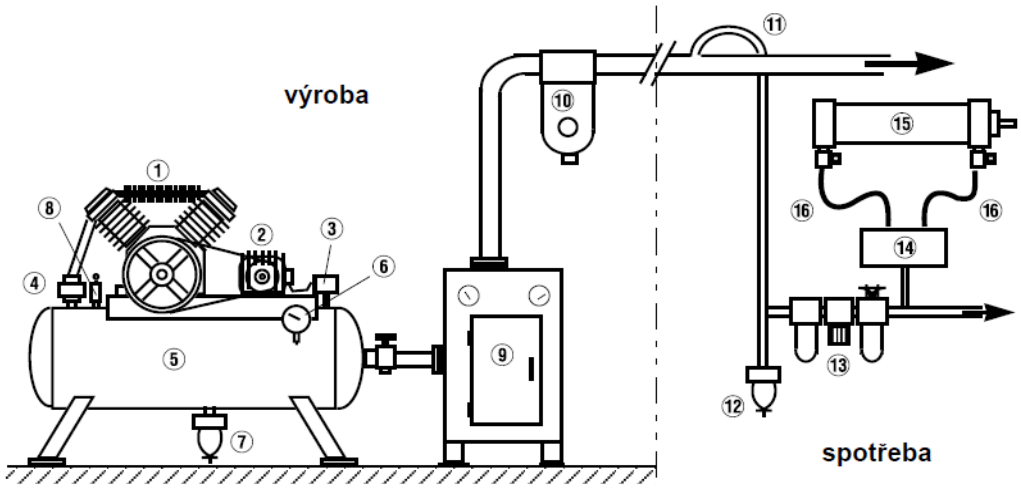
Úprava – úprava se nevyhne takřka žádnému pracovnímu mediu. Ani vzduch není výjimkou a tak je potřeba dbát na odstranění nečistot a vlhkosti způsobující opotřebení pneumatických prvků. Dalším krokem v úpravě vzduchu je obsah oleje, který se do něj dostane při stlačování v kompresoru. Dříve se olej odstraňoval úplně, dnes však převládá názor, že olej ve vzduchu pomáhá a snižuje tření

Náklady – Náklady na výrobu tlakového vzduchu z něj dělají vcelku drahý nosič energie, přesto vše kompenzuje nízkou cenou a vysokým výkonem jednotlivých prvků v soustavě (např. vysokým počtem pracovních cyklů)

2.2 Základní obvod pneumatického modelu

V pneumatickém obvodu využíváme energii stlačeného vzduchu, kterou měníme na energii mechanickou. Pro tuto přeměnu nalezneme v obvodu různé pneumatické motory, úchopné hlavice a podobné prvky, pro přepravu materiálu, montáže nebo zajištění polohy.

Výkonné pohony jsou ovládané a řízené pomocí dalších pneumatických prvků. V obvodu se také setkáme s jednotkami pro úpravu vzduchu, zajišťující filtraci, zbavení nečistot a zbytky olejů z kompresorů. Ventily a rozvaděče určují směr toku vzduchu a tím i pohyb pneumatických pohonů. Škrtící ventily nám usměrňují rychlost protékaného proudu vzduchu, které se promítne do rychlosti pohybu jednotlivých pneumatických pohonů.



Obr. 1 základní obvod [1]

1 – kompresor
2 – elektromotor
3 – tlakový spínač
4 – zpětný ventil
5 – vzdušník
6 – manometr
7 – vypouštění kondenzátu
8 – přetlakový pojistný ventil
9 – vysoušení vymrazováním
10 – filtr hlavní větve
11 – odbočka z hlavní větve
12 – vypouštění kondenzátu
13 – úprava stlačeného vzduchu
14 - ventil
15 – pneumatický pohon
16 – škrtící ventil

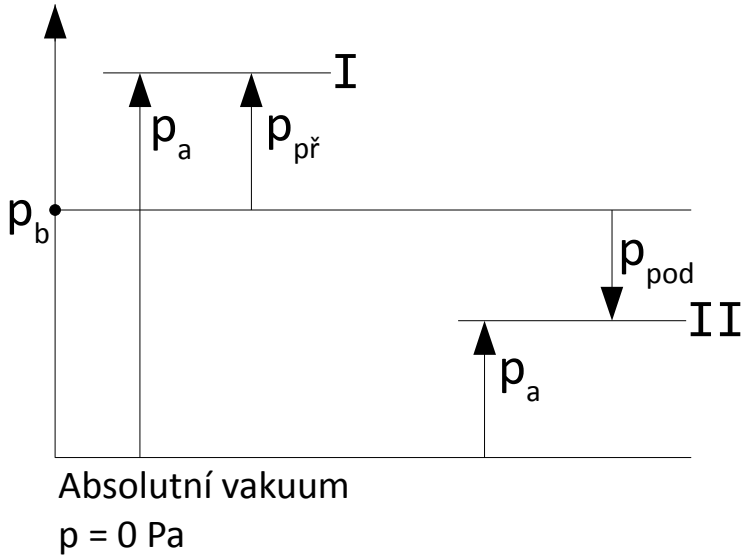
3 FYZIKÁLNÍ VLASTNOSTI STLAČENÉHO VZDUCHU

Úlohy pro řešení stlačeného vzduchu se neobejdou bez nutnosti znát podstatu fyzikálních vlastností, probíhajících v příslušných soustavách. Zde se dostáváme do teorie termodynamiky, kde lze relativně snadno popsat změnu tlaku vzduchu a další procesy spojené se stlačováním vzduchu a aplikovat je na příslušných modelech.

Základní mezinárodní soustava jednotek SI (Le **S**ystème **I**nternational d'Unité [2]) je od svého uvedení v roce 1960 stále více rozšiřována po světě. Soustava SI jednotek využívá 7 základních fyzikálních veličin a z nich vznikají tzv. odvozené jednotky.

3.1.1 Tlak

Na každém místě na Zemi měříme atmosférický tlak. Ten je dán hmotností sloupce vzduchu, který působí na zemský povrch. Barometrický tlak se mění s nadmořskou výškou, tahovým zrychlením a podle současných meteorologických podmínek. Nejvyšší hodnotu má na mořské hladině a s rostoucí výškou klesá. Dále rozeznáváme podtlak a přetlak. V případě podtlaku p_{pod} se hodnota pohybuje od nulového absolutního tlaku až po proměnlivý barometrický tlak p_b. A hodnoty pohybující se nad p_b nazýváme přetlak $p_{př}$. Důležité vědět, že do všech vztahů v termodynamice dosazujeme absolutní tlak p_a (nikdy přetlak ani podtlak).



Obr. 2 Tlak

Z obrázku 2 jednoduše určíme hodnotu absolutního tlaku. Z důvodu proměnlivosti barometrického tlaku byl zaveden tzv. normální tlak vzduchu, který definuje průměrnou hodnotu tlaku vzduchu při mořské hladině na 45° s.š. při teplotě 15°C a tíhovém zrychlení g = 9,80665 ms^{-2} [3].

$$p_b \approx 0{,}1\ MPa \approx 1000\ hPa \approx 1\ atm$$

$$p_a = p_b - |p_{pod}|$$
$$p_a = p_b + p_{př}$$

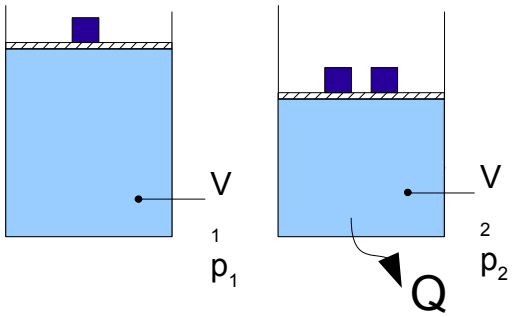
3.1.2 Termodynamické děje

Izotermická změna stavu plynu

Izotermickou změnu popisuje Boyle-Mariottův zákon. Jedná se o jev, kdy objem plynu za stálé teploty je nepřímo úměrný absolutnímu tlaku, resp. Součin objemu a absolutního tlaku je

konstantní. Izotermický děj zapisujeme vztahem:

$$p_1 \cdot V_1 = p_2 \cdot V_2 = konst.$$

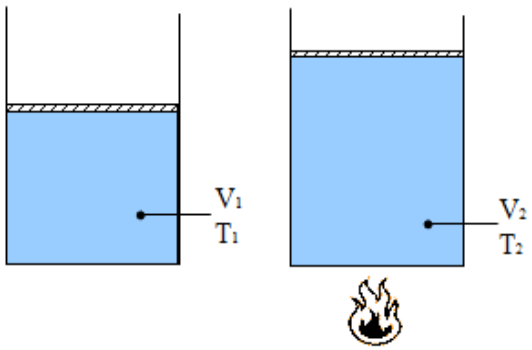


Obr. 3 Grafické znázornění Boyle-Mariottova zákona

Izobarická změna stavu plynů

Změnou stavu plynu při stálém tlaku se zabývá Gay-Lussacův zákon. Tlak v nádobě je konstantní a mění se teplota a tlak. Poměr objemů se rovná poměru teplot vyjádřených v Kelvinech. Matematická formulace:

$$\frac{V_1}{V_2} = \frac{T_1}{T_2}$$



Obr. 4 grafické znázornění Gay-Lussacova zákona

Izochorická změna stavu plynů

Charlesův zákon popisuje chování plynu resp. jeho změnu za konstantního objemu. Matematická formulace:

$$\frac{p_1}{T_1} = \frac{p_2}{T_2}$$

Obecná rovnice stavu plynů

Sloučením předcházejících rovnic lze odvodit obecnou rovnici pro stav plynů:

$$\frac{p_1 \cdot V_1}{T_1} = \frac{p_2 \cdot V_2}{T_2} = konst.$$

3.1.3 Průtok stlačeného vzduchu

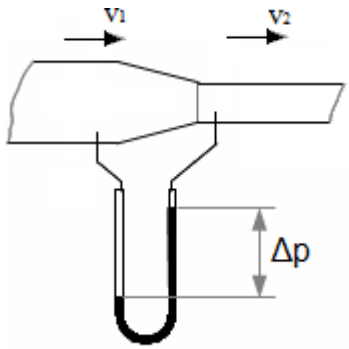
Druhou důležitou veličinou pro určení vhodných pneumatických prvků je průtok vzduchu. Určuje objem vzduchu, který proteče průřezem za jednotku času. Z toho vyplývá jednotka pro průtok vzduchu $m^3 \cdot s^{-1}$.

Bernoulliho rovnice vyjadřuje zákon zachování mechanické energie pro ustálené proudění ideální kapaliny. „Součet kinetické a tlakové potencionální energie kapaliny o jednotkovém objemu je ve všech částech vodorovné trubice stejný.“ [4]

$$p_1 + \frac{1}{2}\rho \cdot v_1^2 = p_2 + \frac{1}{2}\rho \cdot v_2^2$$

ρ – hustota protékajícího media

Bernoulliho rovnice lze ukázat na konstrukci Venturiho trubice (obr. 5). Δp v tomto případě značí rozdíl statických tlaků. Rovnice se také používá pro ostatní plyny, kde není překročena kritická rychlost proudění (asi 300m/s).



Obr. 5 Venturiho trubice [5]

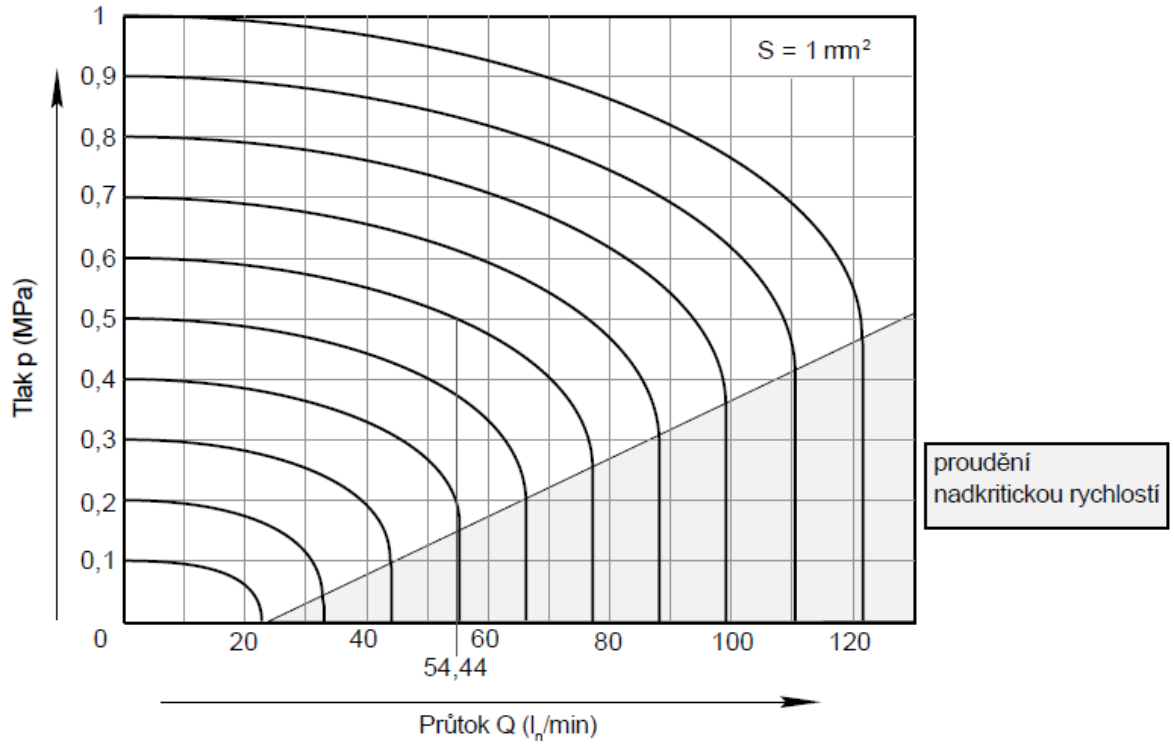
Průtok i tlak se navzájem ovlivňují. Pro výběr pneumatických prvků a konstrukci pneumatických obvodů jsou to nejdůležitější parametry.

Dokud v pneumatickém obvodu stlačený vzduch neproudí, je v každém místě stejný tlak. Začne-li vzduch protékat, musí být tlak na výstupu nižší než na vstupu. Tento rozdíl se nazývá diference tlaku nebo také tlakový spád. Značí se Δp a ovlivňují tři parametry: vstupní tlak, objem protékajícího vzduchu a odpory proudění.

Pro zjištění objemu protékajícího vzduchu potrubím nebo jednotlivými prvky, používáme tzv. P-Q diagram, který zobrazuje závislost vstupního a diferenčního tlaku na objemu vzduchu (obr. 6).

Na svislé ose jsou vyneseny hodnoty vstupních a výstupních tlaků. Když vzduch obvodem neprotéká tlaky jsou stejné, tzn. na vstupu i výstupu pneumatického prvku působí statický tlak. Křivky v diagramu demonstrují jak klesá vzduch na výstupu při zvětšujícím se průtoku stlačeného vzduchu.

Barevná část diagramu zobrazuje proudění nadkritickou rychlostí, tj. vzduch dosahuje nadzvukové rychlosti a průtok je pak možné zvýšit pouze zvýšením vstupního tlaku.



Obr. 6 P-Q diagram [6]

V diagramu můžeme vidět hodnotu objemu protékaného stlačeného vzduchu prvkem s průřezem S = 1,0 mm^2.

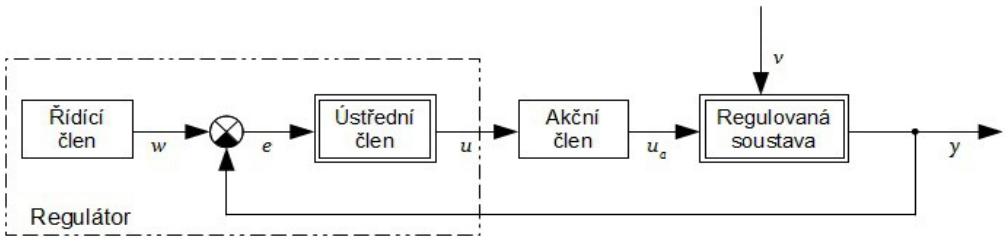
4 ŘÍZENÍ PNEUMATICKÝCH SYSTÉMŮ

Při realizaci jakéhokoliv pneumatického systému lze použít několik možností jak jej řídit. To kterou variantu zvolíme, určuje především složitost řízení dané úlohy. Jedná-li se o jednoduchou úlohu, můžeme použít čistě pneumatické prvky jak pro ovládání, tak i pro řízení. Takto sestavenou úlohu lze provozovat až do chvíle, než budeme chtít znát jakékoliv informace o stavu systému.

V případě, kdy už je zapotřebí mít informaci o stavu systému je nutné použít senzory. Systém, ve které jsou zapojeny senzory využívá zpětnovazebního řízení a ten reprezentuje typickou regulační úlohu. Protože při zpětnovazebním řízení se nejčastěji používají elektrické signály, mluvíme o elektropneumatickém řízení.

4.1.1 Regulace

Regulátor je obecnější označení pro soubor zařízení, který zajišťuje regulační funkci, tj. slouží k dosažení shody mezi žádanou a skutečnou hodnotou regulované veličiny. Jeho nejdůležitější částí je ústřední člen – zpracovává regulační odchylku *e(t)* a generuje signál akční veličiny *u(t).* Dosti často se ústřední člen zkráceně (a nepřesně) nazývá regulátorem. [7]



Obr. 7 Základní schéma regulačního obvodu

Úlohou regulace je udržovat regulovanou veličinu co nejpřesněji na žádané hodnotě a potlačovat nežádoucí vlivy poruchových veličin na regulovanou veličinu. Regulátory můžeme rozdělit podle způsobu regulace na lineární (akcí je lineární funkce odchylky) a nelineární (akce je nelineární funkcí odchylky). Dále rozlišujeme podle práce v čase na spojité (analogové) a diskrétní (číslicové).

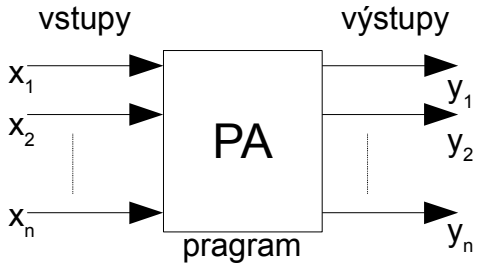
4.1.2 Elektropneumatická řízení

O elektropneumatickém řízení hovoříme tehdy, skládá-li se pneumatický systém z elektromagneticky ovladatelných pneumatických ventilů, které jsou řízeny elektronickými signály z jednotky a které ovládají různé pneumatické mechanismy systému. Elektromagnetické pneumatické ventily jsou ovládány signály z řídící jednotky, do které vstupují signály ze senzorů např. o poloze. Informace z těchto snímačů jsou zpracovány řídící jednotkou a po vyhodnocení jsou jako výstupní signály rozvedeny po celé soustavě k jednotlivým ovládacím ventilům. V elektropneumatickém řízení používáme pro jednodušší úlohy kontaktní prvky elektrického řízení jako jsou relé, spínače nebo stykače. Pro obtížnější úlohy, především pro svoji úsporu kabeláže a dalších předností, jsou voleny programovatelné automaty, kde výstupní hodnoty určuje samotný program.

Programovatelný automat PLC (Programmable Logic Controller) je číslicový elektronický systém navržený pro použití v průmyslovém prostředí, který používá programovatelnou paměť pro uložení uživatelsky orientovaných instrukcí sloužících k implementaci specifických funkcí, jako jsou logické funkce, funkce pro vytváření sekvencí, funkce pro časování, funkce pro čítání a funkce pro

aritmetické výpočty, a to za účelem řízení různých typů výrobních strojů a procesů pomocí číslicových a analogových vstupů a výstupů. [8]

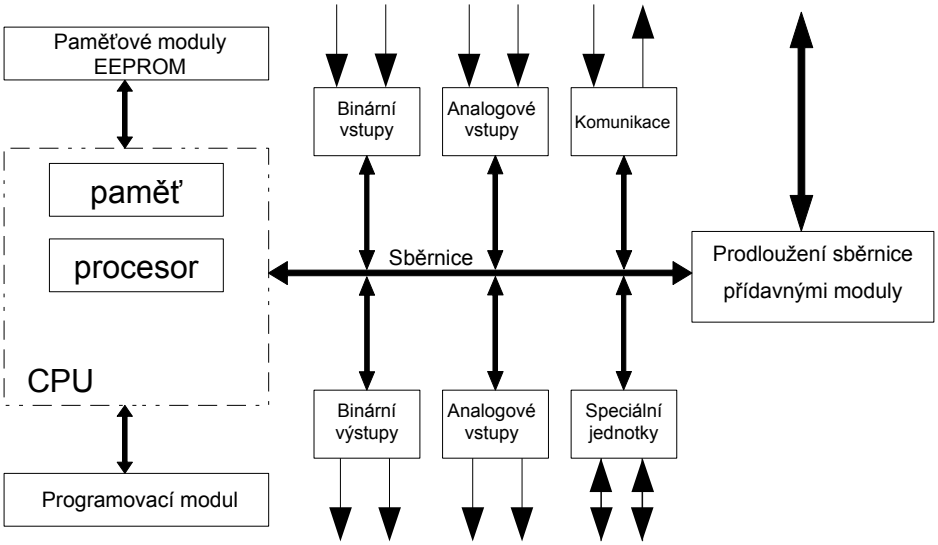
Podobně jako v počítačové historii i programovatelné automaty byly původně navrženy aby nahradily pevné obvody reléové logiky. Samotné chování PLC je snadno měnitelné programem a tím dokonale zastupuje pevné části dřívějších obvodů. V současné době se jim přiřazují složitější úlohy regulačního typu, monitorování a měření. Samotné řízení probíhá uživatelsky definovaným programem, který je napsán v různých jazycích a uložen v paměti PLC. Program obsahuje posloupnost instrukcí, které procesor vykonává cyklicky. Zkráceně řečeno, programovatelný automat zpracovává vstupní logické signály a za pomocí programu je mění na výstupní signály. Popis činnosti PLC lze popsat vztahem $Y = f(X)$, kde f je funkce daná vlastním programem.



Obr. 8 Blokové schéma modulárního PA

Programovatelný automat se skládá z jednotlivých modulů, které jsou propojeny systémovou sběrnicí:

- centrální procesorové jednotky
- systémové paměti
- uživatelské paměti
- modul napájení a záložního zdroje
- vstupních a výstupních jednotek pro připojení řízeného systému
- komunikačních jednotek pro komunikaci se souřadnými i nadřazenými řídícími systémy



Obr. 9 Blokové schéma modulárního PA [9]

4.1.3 Způsob zpracování programu PLC

Jak bylo uvedeno výše, program v programovatelných automatech probíhá cyklicky. Program tedy probíhá opakovaně v tzv. Scanu.

Nejprve se nevzorkují vstupní proměnné tím dojde ke stanovení stavu jednotlivých vstupních veličin. Následně dochází ke zpracování programu a to po jednotlivých instrukcích. Nakonec po ukončení zpracování se předají výsledná data do výstupních jednotek. Vzhledem k rychlému zpracovávání se může zdát, že PLC zpracovává data paralelně. Ve skutečnosti se data zpracovávají sériově a tak je potřeba během probíhajícího programu eliminovat účinek změn vznikajících mezi vstupy a výstupy. Nerespektování těchto změň by mělo za následek nepředvidatelné chování PLC . K tomu PLC využívá svoji paměť. Vždy před každým zpracování jsou do paměti uloženy všechny stavy vstupů a výstupů.

Důvody pro používání skenování v PLC je hned několik. V prvé řadě se jedná o odolnost proti rušení, kdy poruchové signály působící mimo interval vstupního scanu se neprojevují. Chyby, které se projeví v nějaké části programu nemají vliv na ostatní instrukce (s vyjímkou logicky navazujících instrukcí). Známe průměrnou, minimální a maximální dobu zpracování a po překročení maximální doby se generuje chyba. To nám zaručuje nepřekročení doby cyklu a tím i periodicitu. Vždy když dojde k poruše, nám scan zaručí možnost spustitelnosti programu od posledního stavu těsně před poruchou. A v poslední řadě nám umožňuje snadnou programovatelnost.

Programování PLC vychází jako přepis původních logických kontaktních obvodů do logických funkcí v řídícím programu. PLC svými parametry zvládají i popis spojitých a nespojitých regulátorů a tak postupně vytlačují i klasické regulátory. Tomuto se přizpůsobují i programovací jazyky jednotlivých výrobců, kteří se snaží o největší kompatibilitu se svými výrobky a naopak nekompatibilitu s výrobky jiných firem. Proto v roce 1993 vznikla norma IEC 1131, která má na starosti problematiku zabývající se automatizační technikou. Jedna z částí této normy se také zabývá programováním automatizačních prostředků PLC, ale také jiných průmyslových počítačů. V rámci této normy se doporučují používat čtyři programovací jazyky. Dva grafické a dva textové. Mezi grafické patří kontaktní schéma LD nebo také LAD (Ladder Diagram) a logické schéma funkčních bloků FBD (Function Block Diagram). Mezi textové jazyky se řadí seznam příkazů IL (Instruction List) a strukturovaný text ST (Structured Text). Za pátý programovací jazyk lze považovat i SFC (Sequential Function Chart), avšak ten není v normě zařazen přímo mezi programovací jazyky, ale mezi tzv. společné prvky.

Obr. 10 Programovatelný automat firmy Siemens programovaný jazykem FBD [10]

5 PNEUMATICKÉ POHONY

Jak už bylo řečeno v úvodu této práce, pneumatika obecně je stále častěji nasazována v průmyslu na montážních a manipulačních linkách. Důvodů je hned několik. Může to být důsledkem drahé lidské práce, zlevňujících se pneumatických prvků nebo jen prosté zrychlení výrobní linky. Pneumatické pohony v některých činnostech nacházejí větší uplatnění než jejich alternativy a naopak v jiných zase zaostávají. Proto je potřeba nejprve zhodnotit vhodnost použití pro daný technologický proces. A to se neobejde bez alespoň základního shrnutí těch nejdůležitějších vlastností.

5.1 Porovnání hydraulických pohonů s pneumatickými

Dutý válec, ve kterém se pohybuje píst a buď hnací silou je nějaká kapalina nebo stlačený vzduch. Tak lze v jednoduchosti shrnout princip konání práce jak u pneumatických, tak i u hydraulických pohonů. Proto lze říci, že největším rivalem v průmyslových pohonech pneumatiky je hydraulika. Možnosti aplikace jak stlačeného vzduchu, tak i kapalin (nejčastěji použité minerální oleje) jsou téměř shodné. Přesto však nalezneme důležité a v některých ohledech i zásadní rozdíly, především v poddajnosti media nebo jeho viskozitě.

Přednosti hydraulických pohonů v porovnání s pneumatickými:

- dosažení vyšších tlaků
- velká tuhost
- pomalý chod, možnost plynulého a zároveň pomalého chodu bez převodů
- velká účinnost

Nedostatky hydraulických pohonů:

- menší průtoky => problematické dosáhnutí vyšších rychlostí
- rozvod tlakové kapaliny a těsnost soustavy
- viskozita kapaliny v soustavě se mění v závislosti na teplotě, následně se mění tlakové poměry a rychlosti pohybu.
- hořlavost některých použitých kapalin
- nutnost samostatný energetický zdroj na pohon čerpadla u každé soustavy

Přednosti pneumatických pohonů v porovnání s hydraulickými:

- možnost snadného připojení na centrální rozvod
- rychlé přímočaré pohyby
- bezpečné použití ve výbušném prostředí, při velkém teplotním rozsahu
- jednoduchý rozvod bez nutnosti zpětného vracení vzduchu (není zapotřebí odpadové větve)
- přetížitelnost motoru bez poškození
- čistota provozu

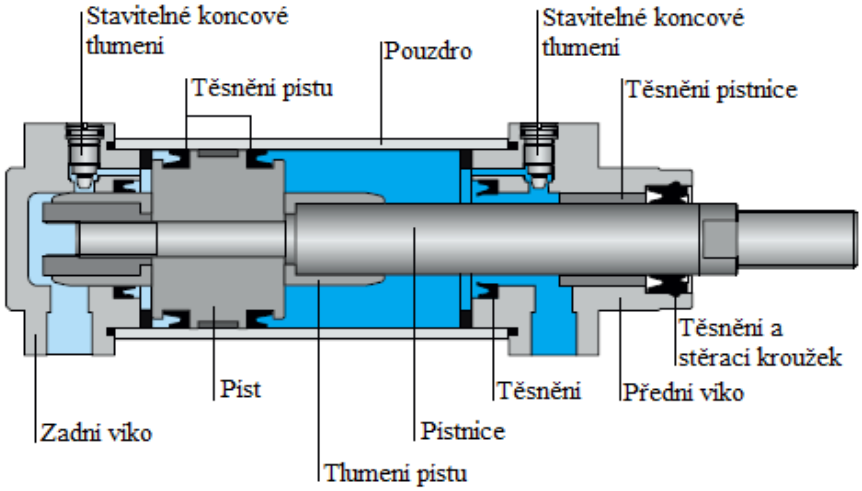
Nedostatky hydraulických pohonů:

- obtížná regulace polohy, zejména při krátkém posuvu
- nerovnoměrnost pohybu při pomalých rychlostech
- pomalá odezva na regulační zásah z důvodu stlačitelnosti media
- výroba stlačeného vzduchu je zpravidla dražší než výroba elektřiny nebo tlakové kapaliny

Z našeho porovnání vyplývá použití pneumatického pohonu. Použijeme je tam, kde jsou zapotřebí menší výkony, řádově asi do 1 kW, jednodušší cykly, snadné nastavení polohy dorazem a tam, kde neobtěžuje nerovnoměrnost pohybu při nižších rychlostech.

5.2 Lineární pohony

V průmyslu hojně využívaný pohon k přemisťování, zvedání nebo podávání nejrůznějších výrobků. Lineární neboli posuvný pohyb má v pneumatice velmi důležitý význam především v jednoduchosti principu pneumatického válce. Síla působí přímo na válec a tak nedochází k jiným ztrátám jako je tomu např. u elektromotoru, kde se musí převádět točivý pohyb na přímočarý přes různé mechanismy. Pohony můžeme dělit do několika skupin, ale v zásadě je dělíme do dvou základních. První skupinou jsou jednočinné válce, kde se stlačený vzduch stará o pohyb pouze v jednom směru a v druhém směru je vracen pružinou. A druhou skupinou jsou dvojčinné válce, zde stlačený vzduch zabezpečuje pohyb v obou směrech.



Obr. 1 Popis pístu [11]

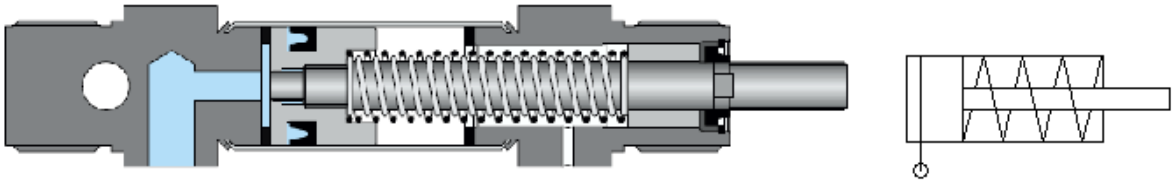
5.3 Pneumatické válce

Pneumatické válce jsou nejrozšířenějším prvkem pro realizaci lineárního pohybu v pneumatice. Válce jsou různých konstrukcí a provedení. Základní konstrukční provedení:

- válce jednočinné – vzduch je přiváděn pouze z jedné strany a vratný pohyb koná pružina
- válce dvojčinné – vzduch je přiveden z obou stran válce

Jednočinné válce

Práce pístu je vykonána tlakovou silou na plochu pístu pouze v jednom směru. Píst se vrací do základní polohy silou pružiny. Síla působící na píst může také vykonávat tažnou nebo tlačnou sílu a to podle provedení. Existují dvě provedení jednočinného válce. Buď s pístnicí v základní poloze zasunutou nebo s pístnicí v základní poloze vysunutou. Nejčastější použití jednočinných pneumatických válců je pro různá podávání polotovarů, vyhazovače a podobné aplikace. Výhodou této konstrukce je menší spotřeba vzduchu oproti dvojčinným válcům. Nevýhodou je využitelná síla, která je menší o sílu pružiny. Tyto válce také obsahují dorazy, které brání dosednutí závitů pružiny a tím bývají delší než válce dvojčinné se stejným zdvihem.

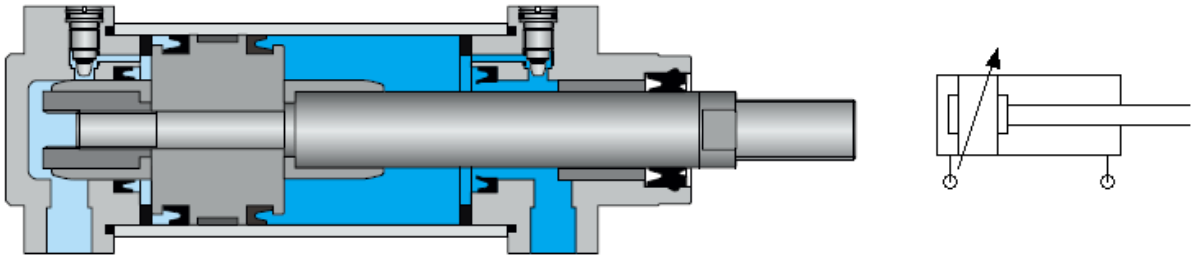


Obr. 11 Funkční řez jednočinného válce a jeho schématická značka [11]

Dvojčinné válce

Síla působící na hlavu pístu působí střídavě v obou směrech podle přísunu vzduchu. Dvojčinné pneumatické válce se použijí u takových aplikacích, u kterých je potřeba vykonat práci i při zasouvání

pístnice. Je důležité pamatovat, že při zasouvání píst nedokáže vykonat tak velkou práci jako při vysouvání. Důvodem je zmenšená plocha, na kterou stlačený vzduch působí, daná průměrem pístnice.



Obr. 12 Funkční řez dvojčinným válcem a jeho schématická značka [11]

5.3.1 Parametry pneumatických válců

Potřebná síla, kterou je pneumatický válec schopen vyvinout se určí z plochy válce, která je dána průměrem, tlaku vzduchu a odpory způsobené třením těsnění pístu a pístnice. Pneumatické válce mají podle normy ISO 4393 a 487 R10 doporučeny tyto průměry: 8, 10, 12, 16, 20, 25, 32, 40, 50, 63, 80, 100, 125, 140, 160, 200, 250 a 320 mm. V tabulce jsou základní rozměry některých pneumatický válců.

Průměr válce [mm]	12	16	20	32	40	50	80	100
Průměr pístnice [mm]	6	8	8	12	16	20	25	25
Délka zdvihu jednočinného Válce [mm]	10, 25, 50				25, 50, 80, 100			
Délka zdvihu dvojčinného Válce [mm]	do 160	do 200	do 320	10, 25, 50, 80, 100, 160, 200, 250, 320, 400, 500				

Tab. 1 Parametry některých pneumatických válců [9]

Mimo tyto rozměry se pneumatické válce vyrábějí také v miniaturním provedení o průměrech 2,5; 4 a 6 mm. Parametry válců se mohou u jednotlivých výrobců lišit.

Výpočet teoretické síly pneumatického válce

$$F = S \cdot p \quad [N]$$

S [m^2] – plocha pístu
p [Pa] – tlak vzduchu ve válci

Výpočet dvojčinného pneumatického válce

- Při vysouvání pístnice

$$F_{vys} = \frac{\pi \cdot D^2}{4} \cdot p \quad [N]$$

D [m] – průměr pístu

- Při zasouvání pístnice

$$F_{zas} = \frac{\pi \cdot (D^2 - d^2)}{4} \cdot p \quad [N]$$

d [m] – průměr pístnice

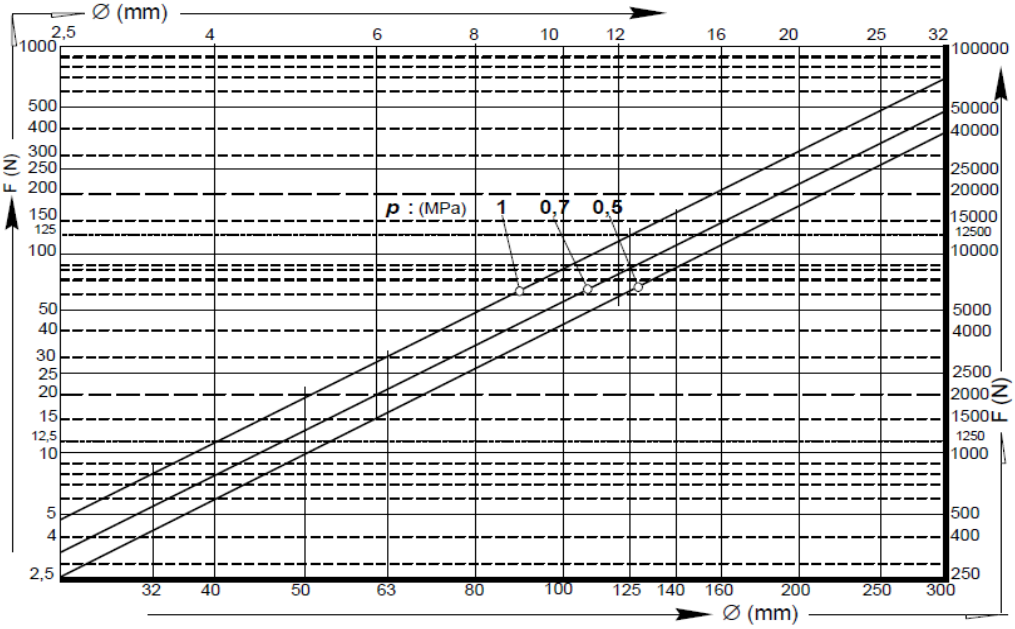
Výpočet pro jednočinné pneumatické válce

$$F_E = \left(\frac{\pi \cdot D^2}{4} \cdot p\right) - F_S \quad [N]$$

F_S [N] – síla stlačené pružiny na konci zdvihu

Při vypočtené výsledné síle zohledníme účinnost pneumatického válce, která se pohybuje mezí 80 až 95 %. Proto volíme vždy nejbližší vyšší průměr válce.

Průměr válce lze také volit z následující tabulky. Levá svislá stupnice nám určuje průměr od 2,5 až po průměr válce 32 mm. Pravá svislá stupnice do průměru 300 mm. Vše pro tlaky 0,5; 0,7 a 1 MPa.



Obr. 13 Teoretická síla F [N] pneumatických válců [12]

Výpočet spotřeby vzduchu pneumatického válce

U výpočtu spotřeby vzduchu vycházíme ze dvou parametrů. Tím prvním je *průměrná spotřeba* vzduchu za minutu. Tento údaj je důležitý při volbě kompresoru, průměru přívodního potrubí, nákladech na energii a celkových nákladů na provoz.

Druhým parametrem je *okamžitá maximální spotřeba* vzduchu. Tuto hodnotu určuje rychlost pístnice a to buď samostatného válce nebo součet maximální spotřeby všech současně pracujících válců. Maximální spotřeba je rozhodující pro určení správné velikosti ventilů, průřezu hadic nebo jednotek pro úpravu vzduchu.

Výpočet spotřeby vzduchu je dán součinem plochy pístu, zdvihu pístu, absolutního tlaku vzduchu a počtem zdvihů za minutu.

$$Q = \frac{1{,}4 \cdot \frac{\pi \cdot D^2}{4} \cdot H \cdot (p + 0{,}1) \cdot n}{10^5} \quad [l \cdot min^{-1}]$$

1,4 – konstanta, nutná pro kompenzaci termodynamických ztrát
D [mm] – průměr pístu
H [mm] – zdvih pístu
p [MPa] – tlak vzduchu ve válci
n [min^{-1}] – počet zdvihů za minutu

Ve výpočtu uvedeném výše není zohledněno několik faktorů ovlivňujících spotřebu vzduchu. Jedná se o mrtvé objemy vyskytujících se v čele válce, dále u velmi dlouhých přívodních hadic je to jejich objem. Spotřeba vzduchu daná objemem a délkou hadic je pak dána vztahem:

$$Q = \frac{1{,}4 \cdot \frac{\pi \cdot d_H^2}{4} \cdot L \cdot p \cdot n}{10^5} \quad [l \cdot min^{-1}]$$

1,4 – konstanta, nutná pro kompenzaci termodynamických ztrát
d_H [mm] – vnitřní průměr hadice
L [mm] – délka hadice
p [MPa] – tlak vzduchu v hadici
n [min^{-1}] – počet zdvihů za minutu

Výpočet okamžité maximální spotřeby vzduchu pneumatického válce

$$Q = \frac{1{,}4 \cdot \frac{\pi \cdot D^2}{4} \cdot v \cdot (p + 0{,}1) \cdot 60}{10^5} \quad [l \cdot min^{-1}]$$

1,4 – konstanta, nutná pro kompenzaci termodynamických ztrát
D [mm] – průměr pístu
v [$mm \cdot s^{-1}$] – rychlost pístu
p [MPa] – tlak vzduchu v hadici

5.4 Pneumatické rozvaděče

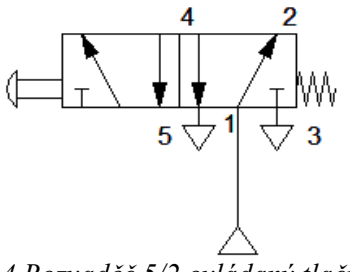
Rozvaděče slouží k ovládání směru průtoku tlakového vzduchu k daným spotřebičům.

Znázornění ve schématech využíváme normalizovaných značek, které nám ukazují pouze funkci rozvaděče. Každý stav je zakreslen v čtverci. Počet takovýchto čtverců nám udává počet funkčních stavů. V každém čtverci jsou nakresleny čáry se šipkou, znázorňující směr průtoku. Každý vstup nebo výstup se značí číslicemi (dříve velkými písmeny), dle následující tabulky.

	dnešní značení	starší značení
Pracovní výstupy	2, 4, 6 (sudé)	A, B, C
Přívod vzduchu	1	P
Odfuk do atm.	3, 5, 7 (liché)	R, S, T
Řídící vstupy	12, 14 (1x – ovládaný výstup)	Z, X, Y

Tab. 2 Normalizované značení vstupů a výstupů

Základním schématickou značkou ventilu určující řízení směru vzduchu je čtverec. Každý čtverec značí jednu polohu ventilu. Šipky uvnitř udávají směr průtoku a značka tvaru T značí uzavření příslušného kanálu. Přívod vzduchu a odfuky se kreslí ke čtverci, který značí polohu ventilu v klidovém stavu a je vždy na spodní straně. Výstupy vzduchu se značí na čtverci klidového stavu na horní straně.



Obr. 14 Rozvaděč 5/2 ovládaný tlačítkem

Ovládání rozvaděčů

Ovládání rozvaděčů lze provádět několika způsoby:

- ruční ovládání (tlačítkem, pedálem, pákou)
- mechanické ovládání (narážkou, kladkou)
- elektrické ovládání (elektromagnet)
- pneumatické ovládání – ovládání tlakem (přímé, nepřímé)
- kombinací výše uvedených ovládání

Konstrukce rozvaděčů

Konstrukce rozvaděčů určuje nejen způsob ovládání, ale také výsledné rozměry nebo celkovou životnost. Dle konstrukce rozdělujeme na:

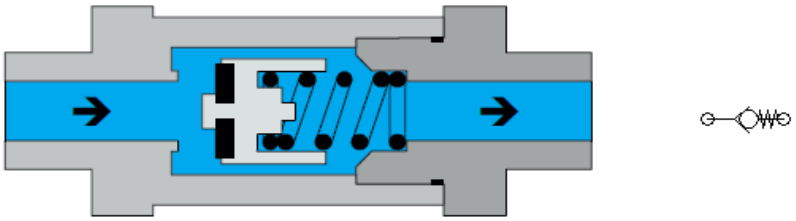
- ventilové (kuličkové nebo talířové)
- šoupátkové (s válcovým šoupátkem, s plochým přímočarým šoupátkem, s rotačním šoupátkem)

5.4.1 Ventilová hradla

Jedná se o konstrukčně velmi jednoduché pneumatické prvky. Jejich úkol většinou spočívá v hrazení průtoku vzduchu v jednom směru, zatímco směr opačný zůstává průchozí. Tlak vzduchu působí na závěrnou stranu a zvětšuje tak těsnost ventilu.

Jednosměrný ventil

Ventil umožňující průchod vzduchu pouze v jednom směru. Konstrukce bývá kuželová, desková nebo membránová. Ventil může být osazen pružinou pro uzavření průchodu i při nízkém tlaku.

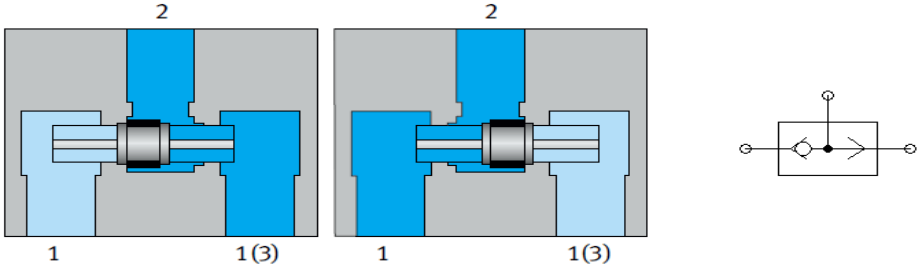


Obr. 15 Funkční řez jednosměrného ventilu a jeho schématická značka [11]

Zpětné ventily

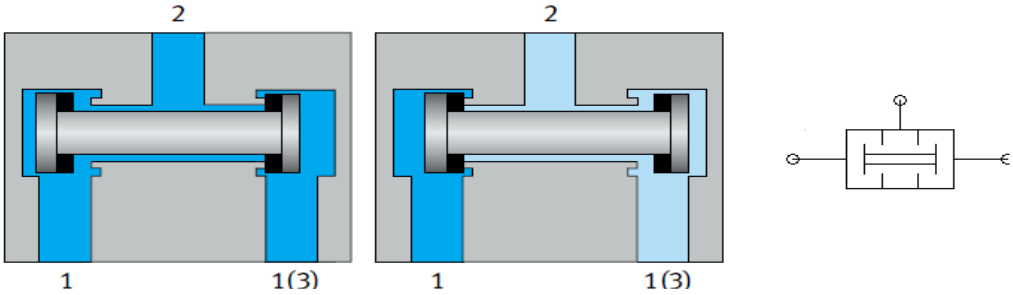
Ventily pro logické řízení průchodu vzduchu jsou osazeny dvěma vstupy a jedním výstupem. Zpětné ventily jsou dvojího druhu:

- Logická funkce OR je zastoupena přepínacím ventilem. Přivedeme-li vzduch na vstup A, dojde k odtlačení kuličky do druhého sedla a uzavření vstupu B. V tento okamžik je propojen vstup A a výstupem Y. Přivedeme-li vzduch do vstupu B, kulička se přemístí na sedlo vstupu A a dojde k propojení vstupu B a výstupu Y.



Obr. 16 Funkční řez ventilu logické funkce OR a jeho schématická značka [11]

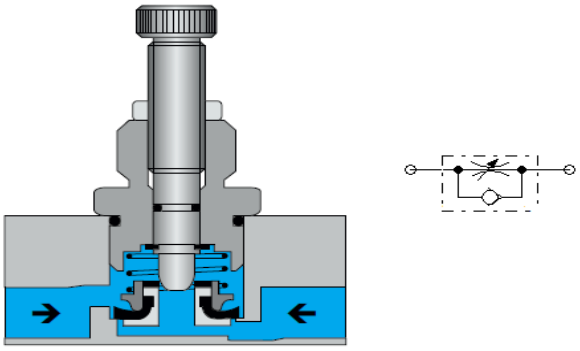
- Logická funkce AND určuje, že na obou vstupech musí být signál aby byl logický signál i na výstupu. Logický ventil AND tedy pracuje na principu dvou sedel. Přivede-li se vzduch pouze na jeden ze vstupů, sedlo se posune na jednu stranu a uzavře průchod vzduchu. Vzduch tedy musí být přiveden do obou vstupů. Tím se sedla udrží uprostřed ventilu a nedochází k jeho uzavření. Vzduch může proudit skrz ventil.



Obr. 17 Funkční řez ventilu logické funkce AND a jeho schématická značka [11]

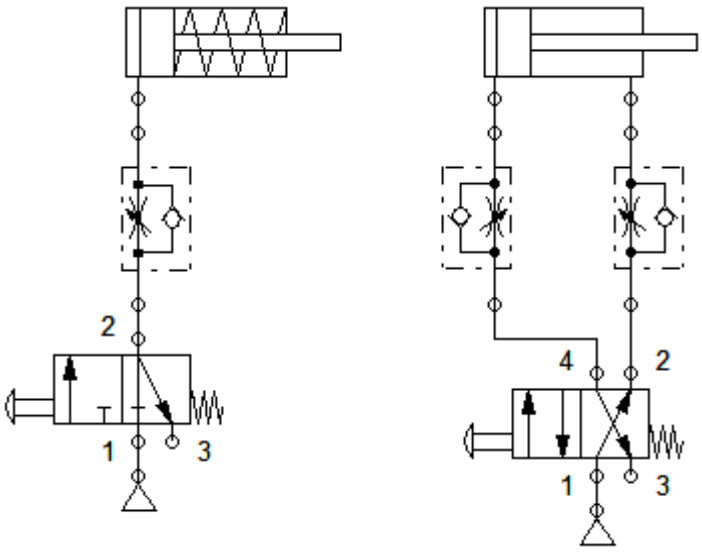
Škrtící ventil

Tento ventil je určen pro řízení rychlosti průtoku vzduchu a tím i pohybu pístu. Průchod vzduchu je v neprůchozím směru přiškrcen a v opačném směru je ventil otevřený a vzduch může procházet mimo přiškrcený průřez.



Obr. 18 Funkční řez škrtícího ventilu a jeho schématická značka [11]

V podstatě jsou dva způsoby jak řídit rychlost pneumotoru pomocí škrtícího ventilu. Jeden způsob je škrcením na vstupu (primárně). Škrtící ventil je umístěn při vstupu do pneumotoru. Tento způsob má velkou nevýhodu v nerovnoměrném pohybu při zatížení motoru. Mnohem výhodnější je způsob zapojení sekundárním škrcením. Vzduch unikající z válce musí projít škrtícím ventilem. Tím je pohyb pístu rovnoměrný i při zátěži.



Obr. 19 Škrcení na vstupu a na výstupu

6 PROGRAMOVÁNÍ V LABVIEW

V historii počítačového softwaru se setkáváme s různými typy programovacích jazyků. Přitom každý jazyk má své specifikace a z toho plynou i jeho výhody popřípadě nevýhody. Jednou velmi důležitou složkou programovacího jazyka je vývojové prostředí. Vývojové prostředí určuje nejen jak rychle je schopen programátor pochopit základní principy, ale také jak je schopen se v programu následně orientovat. Mnoho moderních vývojových prostředí má společnou vlastnost rychlé a efektivní tvorby programu. V angličtině jsou takováto prostředí pojmenována Rapid Application Development. Též se můžeme setkat se zkratkou RAD. A podobně jako v Delphi má vývojové prostředí Object Pascal, jehož zdrojovým kódem je text, tak i LabVIEW má vývojové prostředí, ale jeho zdrojovým kódem není text, nýbrž obrázek.

Grafické programování, které si firma National Instruments nechala patentovat v roce 1990, ušel řádný kus cesty a dnes jej lze považovat za plnohodnotný programovací jazyk se všemi náležitostmi, které k jazyku, tak i k vývojovému prostředí patří. Součástí Vývojového prostředí je i kompilátor, který generuje samostatně spustitelný program. Při programování v LabVIEW má uživatel k dispozici jak rychlé funkce nizkých úrovní, tak i před-připravené podprogramy pro nejrůznější aplikace jako jsou např. Matematická analýza, statistika nebo komunikace se standardizovanými periferiemi.

Protože většina prováděných akcí se při programování uskutečňuje myší, budeme v tomto textu uvažovat nastavení myši pro praváky. Dále je předpoklad alespoň základních znalostí programování jiných jazycích a vhodné uživatelské znalosti používání PC. Všechny programy uložené na CD jsou vytvořeny v programu LabVIEW ve verzi 12.0f5 (32-bit).

6.1 Vývojové prostředí

Při práci v programu LabVIEW se často setkáváme s označením VI. Jedná se o akronym Virtual Instruments. Takto označujeme uživatelská rozhraní podobající se skutečným přístrojům. I v této práci budeme nazývat program LabVIEW označením VI.

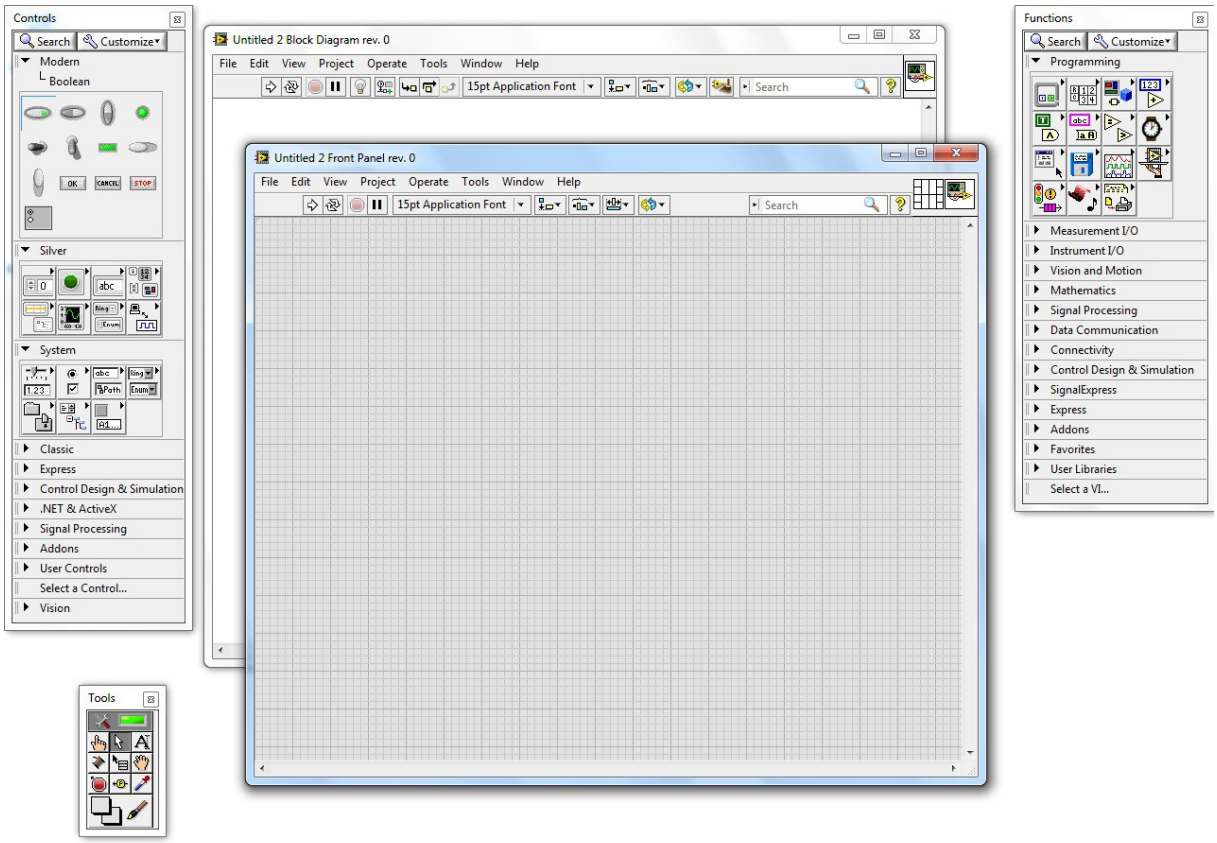
Vytváření samotného VI se provádí ve vývojovém prostředí LabVIEW. Sestává se z několika oken. Při spuštění se objeví první, které nám dává na výběr vytvoření nového projektu, popřípadě otevření již vytvořeného. Také máme na výběr vytvoření samostatného VI. Po zvolení *New VI* se otevře již zmíněná dvojice hlavních oken jako tomu je na obrázku. Další okna jsou palety funkcí, nástrojů nebo kontrolních prvků.

- V popředí je okno, které nazýváme čelní panel (Untitled 2 Front Panel). V tomto okně provádíme návrh uživatelského rozhraní. S čelním panelem bezprostředně souvisí paleta *Controls* popsaná podrobněji níže. Toto okno slouží jako "hlavní" a tudíž pokud chceme jakkoliv s VI pracovat nesmíme jej zavřít jinak dojde k uzavření celého VI. Pro přepínání mezi čelním panelem a blokovým diagramem můžeme použít klávesovou zkratku CTRL+E.

- Blokový diagram (Untitled 2 Block Diagram) je druhé okno pro práci s programem LabVIEW. Slouží k tvorbě blokového diagramu VI. Podobně jako čelní panel má svoji paletu, tak i blokový diagram má svoji paletu *Functions*. Okno po editaci můžeme zavřít a dále pracovat pouze s čelním panelem. Opětovné otevření provedeme pomocí CTRL+E nebo v čelním panelu klepneme pravým tlačítkem myší na nějaký prvek a z menu vybereme *Find Terminal.*

- Paleta *Controls* je spojena s čelním panelem, takže je viditelná pouze pokud je okno čelního panelu aktivní. Ze stromové struktury, tvořící paletu, si lze vybírat objekty pro vložení do okna čelního panelu. Paletu můžeme skrýt kliknutím na křížek a opět zobrazit zvolením v

horním menu okna View → *Controls Palette.* Zvolíme-li aktivním oknem blokový diagram, paleta *Controls* se nám skryje a zobrazí se paleta *Function.*



Obr. 20 Vývojové prostředí LabVIEW

- Paleta *Controls* je spojena s čelním panelem, takže je viditelná pouze pokud je okno čelního panelu aktivní. Ze stromové struktury, tvořící paletu, si lze vybírat objekty pro vložení do okna čelního panelu. Paletu můžeme skrýt kliknutím na křížek a opět zobrazit zvolením v horním menu okna View → *Controls Palette.* Zvolíme-li aktivním oknem blokový diagram, paleta *Controls* se nám skryje a zobrazí se paleta *Function.*

- Paleta *Functions* je viditelná jen v případě aktivního okna blokového diagramu. Stejná stromová struktura, jako u palety *Controls,* usnadňuje orientaci při výběru prvků do okna blokového diagramu. Paletu lze zavřít klepnutím na křížek, popřípadě zobrazit v horní nabídce View → *Functions Palette*.

- Paleta *Tools* může být pro některé uživatele vhodnější náhradou za automatické měnění funkcí kurzoru myši. Paletu zavoláme z menu View → *Tools Palette* a uzavřeme klepnutím na křížek. Jak již bylo naznačeno, paleta slouží ke změně funkcí kurzoru myši, které si následně popíšeme.

Myš je pro práci v programu LabVIEW nepostradatelná a bude s ní provádět většinu akcí. Kurzor myši má v LabVIEW několik režimů. Z palety *Tools* můžeme vybrat jeden z deseti režimů nebo pomocí tabulátoru lze efektivně přepínat mezi čtyřmi nejčastějšími.

- *Positioning* je základní režim reprezentující kurzor podobou šipky . Slouží především k přemisťování objektů umístěných na pracovní ploše.

- *Wiring*. Při zapnutém režimu se kurzor změní na cívku s drátem . Tímto režimem lze propojovat jednotlivé prvky umístěné na pracovní ploše.

- Režim *Operating* nám umožňuje výběr z více předdefinovaných možností. Signalizuje ho podoba ručičky .

- Labelling slouží k úpravě nebo vytváření nových textu a popisků. Kurzor v tomto režimu má podobu obdélníčku s kurzorem psaní .

Ukládání VI se provádí do jediného souboru označeného příponou .vi a obsahuje veškeré informace ke spuštění programu. Může nastat situace, kdy VI volá jiný VI, nebo se dožaduje nějaké knihovny, pak je nutná přítomnost knihovny nebo volaného VI. Uložení provedeme v menu *File → Save/Seve As.../Seve All* nebo tak jak jsme zvyklí CTRL+S. VI nám umožňuje také uložit soubor tak, aby jej bylo možné otevřít ve starších verzích LabVIEW. Tuto možnost nalezneme v menu *File → Seve for Previous Version.* Otevřít VI můžeme dvojklikem na uloženy soubor, v horním menu zvolením *File → Open* nebo klávesovou zkratkou CTRL+O.

6.2 Programování

6.2.1 Datové typy

Datové typy známe z běžných textových programovacích jazyků. Ani LabVIEW se neobejde bez datových proměnných. V LabVIEW znázorňuje datový typ jako nějakou vlastnost hrany a terminálu. Tzn. každý výstupní terminál generuje určitý datový typ a každý vstupní terminál umí pracovat s určitým datovým typem. Kromě klasických datových typů jako integer, string apod. najdeme v LabVIEW i netypické, které lze nalézt v nápovědě.

Datový typ hrany je v LabVIEW určen grafickým typem čáry, která určuje hranu. Datový typ terminálu je určen grafickou ikonkou s trojicí písmen, reprezentující daný typ. Aby program fungoval správně je nutné aby všechny spojené uzly měly stejný datový typ. Přesto, že je nutné aby se datový typ shodoval grafické znázornění nemusí být vypovídající.

Obr. 21 Různé zobrazení se shodným datovým typem

Na obrázku 21 vidíme dva uzly spojené hranou. Výstupní terminál na pravé straně není nějak grafický označený, proto nemůžeme s určitostí říci, jakého datového typu je. Ani z hrany, kterou jsou oba uzly spojené nelze s přesnou určitostí říci jakého typu je. Z hrany je pouze patrné, podle tenké oranžové barvy, že se jedná o typ hrany floating-point-numeric. Až z pravého uzlu se dovídáme, že se jedná o datový typ Double-precision floating-point-numeric.

Vybrané datové typy terminálů uvádí tabulka [4] a datové typy pro hrany tabulka [3]. Více lze nalézt v nápovědě LabVIEW pod heslem 'data types'.

6.2.2 Konstanty

Konstanty jsou při programování velmi důležitým prvkem. V LabVIEW se můžeme setkat s několika druhy konstant a to vždy v závislosti na datovém typu, který daná konstanta reprezentuje. Je označována ikonou obdélníku, jehož barva určuje datový typ a jedná se vždy o výstupní uzel.

Konstanta číselná

Asi nejčastěji se při vytváření programu setkáme s číselnou konstantou. Nalezneme ji v paletě blokového diagramu *Functions → Numeric → Numeric Constant.* Datový typ konstanty určuje číslo, které do obdélníku vepíšeme. Pokud napíšeme celé číslo, datový typ konstanty pak bude celočíselný. Číselné konstanty lze vybírat i z předdefinovaných v nabídce *Functions → Numeric → Additional Numeric Constants.*

Např. vložíme do blokového diagramu z palety *Functions → Numeric → Numeric Constant* konstantu a do obdélníku napíšeme číselnou hodnotu 7. V tu chvíli vidíme, že obdélník má modrou barvu a obsahuje hodnotu 7. To nám značí, že datový typ je celé číslo.

Konstanta pole

Konstantu typu pole vložíme do blokového diagramu *Function → Array → Array Constant.* Po vložení je potřeba ještě nastavit typ pole. Provede stejným způsobem jako vložení jiné konstanty, tedy přes paletu *Function.* Do takto vytvořeného pole už však nelze vložit další konstantu typu pole. Pokud se nám nehodí velikost zvoleného pole, je možné jej změnit po kliknutí pravým tlačítkem na konstantu.

Např. pro vytvoření konstanty pole, která bude obsahovat celá čísla, vložíme do blokového diagramu *Function → Array → Array Constant.* Následně musíme zvolit typ prvků v poli na celočíselné. To provedeme vložením *Functions → Numeric → Numeric Constant.*

Konstanta Boolean

Pro určení vstupní hodnoty pravdy nebo nepravdy, tedy logických funkcí v LabVIEW používáme konstanty True resp. False, které nalezneme v *Functions → Boolean → True Constant* resp. *Functions → Boolean → False Constant.*

Konstanta řetězce

Poslední konstantou kterou je dobré zmínit je poměrně složitější typ řetězce. Nalezneme ji v blokovém diagramu v paletě *Functions → String.* Po vložení vidíme, že typ řetězec má růžovou barvu. Samotné vkládání nebo nastavení není složitější než předešlé konstanty. Vkládání textu probíhá stejným způsobem. Klikneme do obdélníku a můžeme psát text, který má konstanta obsahovat. Problém nastává při vkládání speciálních znaků, které je potřeba vkládat přes obrácené lomítko 'backslash' a za lomítkem je potřeba uvést vkládaný znak v hexadecimálním kódu. Číslo za lomítkem musí být v rozsahu 00 až FF a to pouze velkým písmem. Daný kód pro speciální znak je určen z ASCII tabulky. Podobně jako u číselných konstant i zde je možné vybírat z předdefinovaných konstant jako např. prázdný řetězec apod.

Např. chceme-li vložit konstantu s textem 'Hello World', nejprve vybereme z palety blokového diagramu *Functions → String* konstantu typu řetězec a poté do obdélníčku napíšeme daný text.

6.2.3 Control a Indicator

Předešlou kapitolu jsme se věnovali konstantám. V této kapitole si přiblížíme další dva prvky, z trojice *Constant, Control a Indicator.* Prvek *Control* je podobný konstantě. Tvoří jej výstupní terminál. Opak je pak *Indicator,* který reprezentuje vstupní terminál.

Tato trojice nám velmi usnadňuje tvorbu programu. Při tvorbě programu máme několik možností jak postupovat. Logickou a pro začátek vhodnou variantou tvorby programu je vkládat postupně prvky do čelního panelu z palety nástrojů. Cesta jistě správná, však má několik úskalí. V prvé řadě je potřeba si rozmyslet vstupní a výstupní terminály a hlavně je třeba brát zřetel na vhodný datový typ.

Vhodnější varianta tvorby programu má opačný průběh. Nejprve do blokového diagramu vložíme terminály, které jsou potřebné ke správné funkci programu a poté začneme vkládat vytvářet čelní panel. Zde nám velmi usnadňuje práci samotné prostředí LabVIEW. Vkládáme-li ovládací prvky k jednotlivým terminálům, můžeme si usnadnit práci kliknutím pravým tlačítkem na daný terminál a v roletové nabídce zvolit *Create Constant, Create Control* nebo *Create Indicator.* U takto vytvořeného prvku se automaticky zvolí i vhodný datový typ.

Pokud se rozhodneme změnit prvek v blokovém diagramu není nutné jej mazat a vkládat nový. Postačí, když na něj klikneme pravým tlačítkem a v menu vybereme *Change to Constant* resp. *Change to Control* resp. *Change to Indicator*. Po provedení změny nám datový typ zůstane zachován.

V závěru je potřeba upřesnit zákonitostí při zaměňování těchto tří prvků. Je potřeba si uvědomit, že zaměníme-li konstantu za prvek typu *Control* nebo opačně, v čelním panelu vznikne resp. zanikne ovládací prvek. Po této změně není VI nějak narušeno a je stále funkční. Zaměníme-li ovšem konstantu nebo prvek typu *Control* za *Indicator* a opačně, nastává zásadní zásah do funkčnosti VI a následně je potřeba program upravit v blokovém diagramu.

	Skalár	Pole 1D	Pole 2D
Floating-point numeric (Single-precision floating-point numeric, Double-precision floating-point numeric, Extended-precision floating-point numeric, Complex single-precision floating-point numeric, Complex double-precision floating-point numeric, Complex extended-precision floating-point numeric)			
Integer numeric (Signed 8-bit integer numeric, Signed 16-bit integer numeric, Signed 32-bit integer numeric, Unsigned 8-bit integer numeric, Unsigned 16-bit integer numeric, Unsigned 32-bit integer numeric, Enumerated type)			
Boolean			
String			

Tab. 3 Datové typy hran v LabVIEW

Control	Indicator	Popis
		*Single-precision floating-point numeric* Datový typ reálných čísel. Číslo je v paměti uloženo binárně v 32 bitech. Nejmenší zobrazitelné kladné číslo: 1.40e-45, největší zobrazitelné kladné číslo: 3.40e+38, nejmenší zobrazitelné záporné číslo: -1.40e-45, největší zobrazitelné záporné číslo: -3.40e+38. Počet platných desetinných míst: 6.
		Double-precision floating-point numeric Datový typ reálných čísel. Číslo je v paměti uloženo binárně v 64 bitech. Nejmenší zobrazitelné kladné číslo: 4.94e-324, největší zobrazitelné kladné číslo: 1.79e+308, nejmenší zobrazitelné záporné číslo: -4.94e-324, největší zobrazitelné záporné číslo: -1.79e+308, Počet platných desetinných míst: 15.
		Extended-precision floating-point numeric Datový typ reálných čísel. Číslo je v paměti uloženo binárně v 128 bitech. Nejmenší zobrazitelné kladné číslo: 6.48e-4966, největší zobrazitelné kladné číslo: 1.19e+4932, nejmenší zobrazitelné záporné číslo: -6.48e-4966, největší zobrazitelné záporné číslo: -1.19e+4932. Počet platných desetinných míst je závislý na operačním systému pod kterým LabVIEW provozujeme a pohybuje se mezi 15 a 30.
		Signed 8-bit integer numeric Celočíselný datový typ. Číslo je v paměti uloženo binárně v 8 bitech. Rozsah zobrazení: -128 . . 127.
		Signed 16-bit integer numeric Celočíselný datový typ. Číslo je v paměti uloženo binárně v 16 bitech. Rozsah zobrazení: -32768 . . 32767.
		Signed 32-bit integer numeric Celočíselný datový typ. Číslo je v paměti uloženo binárně v 32 bitech. Rozsah zobrazení: -2147483648 . . 2147483647.
		Unsigned 8-bit integer numeric Celočíselný datový typ bez znaménka. Číslo je v paměti uloženo binárně v 8 bitech. Rozsah zobrazení: 0. .255.
		Unsigned 16-bit integer numeric Celočíselný datový typ bez znaménka. Číslo je v paměti uloženo binárně v 16 bitech. Rozsah zobrazení: 0..65535.
		Unsigned 32-bit integer numeric Celočíselný datový typ bez znaménka. Číslo je v paměti uloženo binárně v 32 bitech. Rozsah zobrazení: 0..4294967295.
		Boolean Logický datový typ.
		String Datový typ řetězec.

Tab. 4 Datové typy některých terminálních uzlů [13]

6.2.4 Programové struktury

Ani v graficky orientovaných vývojových prostředích se neobejdeme bez klasických, programátorům známých, řídících struktur. Základem pro programování je cyklus FOR a WHILE. Dále se v LabVIEW setkáme se strukturou CASE a Sequence, která patří mezi nezvyklé struktury. Všechny řídící struktury tvoří rámeček. Vše co se nachází uvnitř rámečku je vykonáváno podle typu řídící struktury.

Cyklus FOR (FOR Loop)

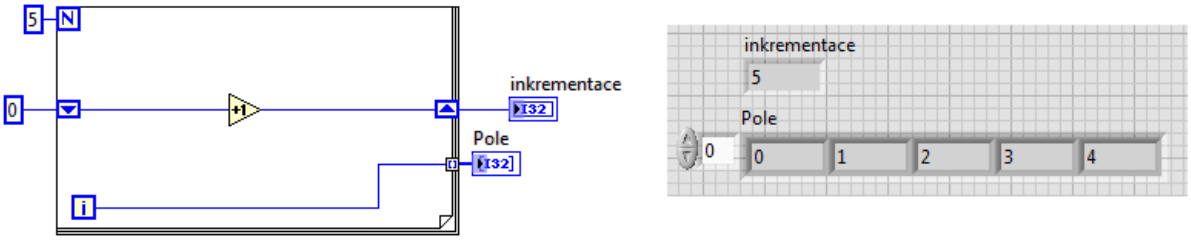
Cyklus FOR stejně jako je tomu u textově orientovaných jazycích slouží k N-krát opakování takové části blokového diagramu, která se nachází uvnitř rámečku. Cyklus nalezneme v paletě blokového diagramu Functions → Structures → For Loop.

Obr. 22 Reprezentaci cyklu FOR

Velikost rámečku lze samozřejmě upravit chycením a tažením za některý roh. Po vložení do blokového diagramu vidíme, že nedílnou součástí cyklu FOR jsou i dva terminály. V levém horním rohu se nachází terminál pro určení počtu cyklů. Jedná se o vstupní terminál, takže do něj lze připojit např. konstantu s hodnotou počtu opakování. V levém dolním rohu se nachází terminál výstupní, ze kterého můžeme číst hodnotu již vykonaných cyklů. Při prvním průchodu se hodnota nastaví na 0 a tomu odpovídá i první průchod. Poslední bude mít tedy hodnotu i = N-1.

Jak již bylo řečeno v úvodu, cyklicky se bude opakovat pouze ta část programu, která je obklopena rámečkem cyklu FOR. K propojení neopakující se části programu s programem nenacházející se ve smyčce FOR, slouží tzv. tunel. Tunel není potřeba nějak zvlášť vytvářet, stačí pouze hranou propojit prvek nacházející se ve smyčce s prvkem mimo ni a tunel se vytvoří automaticky. Tunel, přes který do cyklu vstupují data nazýváme vstupní tunel a naopak výstupní tunel nazýváme ten, přes který data vystupují ven z cyklu FOR.

Mimo tunel, se může ve smyčce nacházet posuvný registr *(Shift Register)*. Přenáší hodnoty z jednoho kroku iterace (oběhu cyklu) smyčky FOR do následujícího. Je to taková obdoba lokální proměnné, která předá výstupní hodnotu na konci cyklu jako vstupní hodnotu na začátek dalšího. Posuvný registr vložíme po kliknutí pravým tlačítkem myši na rámeček a zvolíme v menu *Add Shift Register*. Více si posuvný registr přiblížíme na jednoduchém příkladu.



Obr. 23 Praktický příklad cyklu FOR

Posuvný registr vidíme na obr. 23 jako párové šipky na okraji rámečku cyklu FOR. Konstanta do něj vstupující značí, že v první iteraci bude posuvný registr nastaven na hodnotu 0. Druhá konstanta vstupující do terminálu N značí, že cyklus bude mít 5 iterací V programu se bude opakovat pouze uzel, ve kterém se při každé iteraci přičte jednička. Výsledky se vypíší do výstupních terminálů tak jak je můžeme vidět vpravo na obr. 23. Příklad lze nalézt na přiloženém CD pod názvem 01.vi.

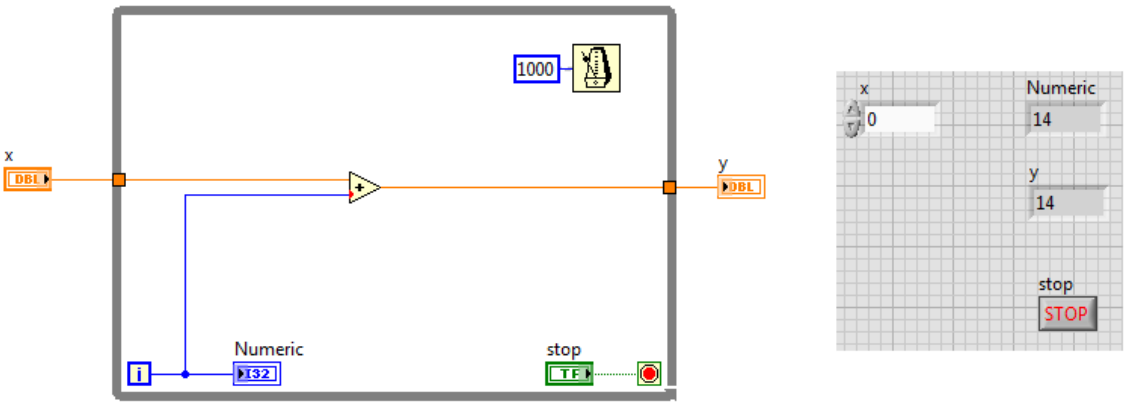
Cyklus WHILE

Další z volby cyklů, který známe především z textových jazyků. Nalezneme jej v paletě blokového diagramu *Functions → Structures → While Loop.* Cyklus WHILE využíváme k neustálému provádění té části programu uvnitř cyklu. Jak můžeme vidět z obrázku, cyklus WHILE reprezentuje rámeček jež je tvořen šipkou. Stejně jako tomu je u FOR cyklu i zde vidíme v levém dolním rohu ikonku značící počet opakování. Změnou oproti předchozímu je ikonka vpravo dole  . Jedná se o standardní řídící terminál. Cyklus trvá dokud na tento terminál přivádíme logickou jedničku, tedy hodnotu *TRUE.* Testování se provádí vždy až na konci cyklu. Z toho plyne, že první průchod se provede vždy. Ikonka lze změnit na  , pak cyklus probíhá tak dlouho dokud je na terminál přiváděna logická hodnota *FALSE.* Ikonky lze vzájemně zaměňovat kliknutím pravým tlačítkem a zvolením položky *Stop if True* resp. *Continue if True.*

Obr. 24 Reprezentace cyklu WHILE

Cykly FOR i WHILE pracují podobným způsobem, tak pro ně platí i stejná pravidla. Lze zde využívat vstupní a výstupní tunely a posuvné registry, které se chovají stejně jako je tomu u cyklu FOR.

Často se můžeme setkat s požadavkem, aby jednotlivé průchody probíhaly s odstupem. K tomu nám poslouží vstupní terminál z palety blokového diagramu *Functions → Time&Dialog → Wait Until Next ms Multiple.* Hodnota konstanty vstupujícího do terminálu určuje, o kolik milisekund se daná iterace opozdí. Takto lze jednoduše určit posloupnost několika vytvořených cyklů.

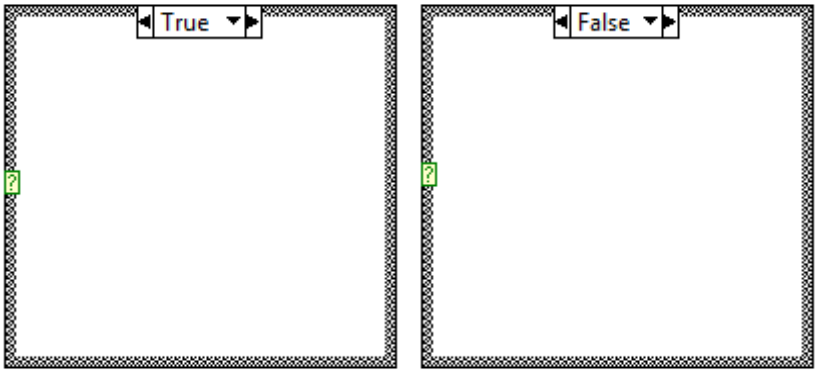


Obr. 25 Praktický příklad cyklu WHILE

Na obr. 25 vidíme příklad s použitím cyklu WHILE. Řídící terminál je podobu ikonky  , jenž znamená, že se cyklus bude provádět dokud nezmáčkneme tlačítko STOP a nepřivedeme na terminál logickou hodnotu TRUE. Dále zle spatřit *Wait Until Next ms Multiple* terminál, do kterého vstupuje hodnota 1000, tzn. zpoždění každé iterace o 1000 ms. Posledním prvkem ve smyčce se nachází sčítací člen. Sčítá hodnotu každé iterace počínaje nulou a hodnotu přivedené z terminálu X. Průběžné sčítání lze vidět na terminálu NUMERIC a poté co cyklus ukončíme tlačítkem STOP se konečná hodnota zapíše do terminálu Y. Příklad nalezneme na přiloženém CD pod názvem 02.vi.

Struktura CASE

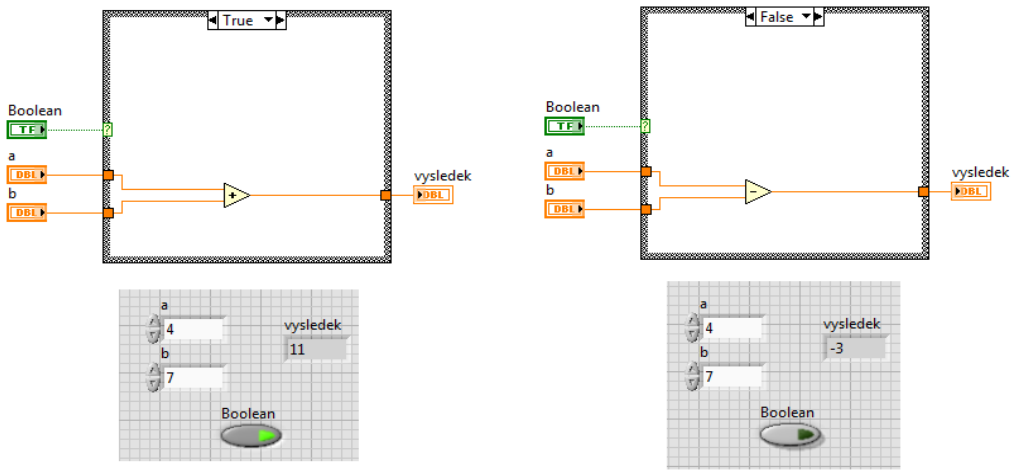
Strukturu CASE nalezneme v paletě funkcí Functions → Structures → CASE. Po vložení do blokového diagramu vidíme rámeček z obrázku 26.



Obr. 26 Reprezentace struktury CASE

Struktura CASE se skládá ze dvou nebo více rámečků. Přepínat mezi jednotlivými okny diagramu lze snadno za pomocí šipek. Takováto struktura je vhodná především v případě, chceme-li aby v jednom případě program provedl obsah diagramu 1 a v jiném případě obsah diagramu 2. A to který diagram se provede, určí hodnota, kterou přivedeme na rozhodovací termináš  .

Po vložení struktury CASE do blokového diagramu se nám nakonfigurují dvě prázdná okna s logickými hodnotami *TRUE* a *FALSE*. V tuto chvíli lze do řídícího terminálu přivést logické proměnné. Rozhodovací proměnné lze změnit na jiné typy jako např. Řetězec. Chceme-li přidat další okénko ve struktuře, klepneme pravým tlačítkem myši na rámeček a zvolíme možnost *Add Case After* nebo *Add Case Before*. Pro odebrání volíme možnost *Delete This Case.*

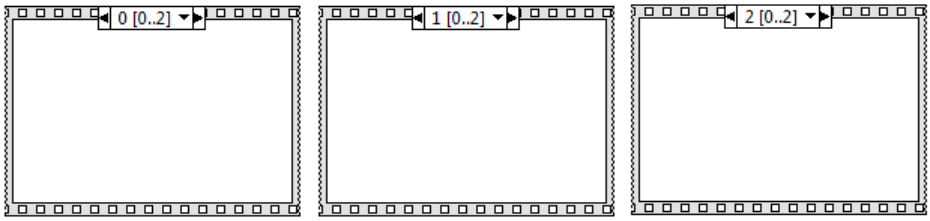


Obr. 27 Praktický příklad struktury CASE

Příklad z obrázku 27 pracuje velice jednoduše. Do rozhodovací proměnné nám vstupuje Boolean hodnota. Pokud je tlačítko v čelním panelu zapnuté, do rozhodovacího terminálu vstupuje hodnota TRUE, v opačném případě FALSE. V blokovém diagramu máme umístěnou strukturu CASE se dvěma okny nastavenými na hodnotu TRUE a FALSE. Podle toho jakou hodnotu přivedeme z tlačítka na rozhodovací terminál, vykoná se příslušné okno struktury. V případě sepnutého tlačítka (pravá strana obrázku) se vykoná okno, kde vstupující hodnoty z terminálu A a B se sečtou a výsledek se zapíše do terminálu VYSLEDEK. V případě vypnutého tlačítka (levá strana obrázku) se vykoná okno s hodnotou FALSE a vstupující hodnoty z terminálu se od sebe odečtou. Příklad nalezneme na přiloženém CD pod názvem 03.vi.

Struktura SEQUENCE

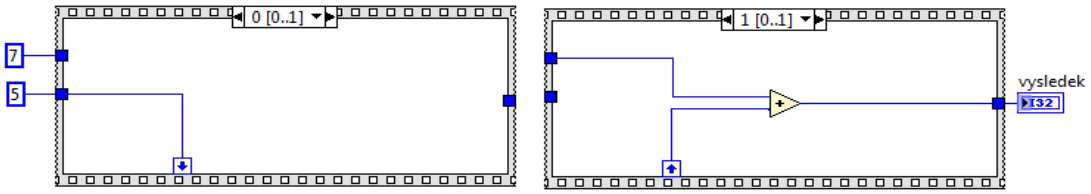
Velkou nevýhodou grafického prostředí pro tvorbu programu je problematické určení sledu jednotlivých kroků. U textově orientovaného jazyka je to vyřešeno posloupností příkazů napsaných ve zdrojovém kódu. K tomu abychom mohli provádět některé kroky v LabVIEW dříve než jiné, nám slouží struktura Sequence nacházející se v paletě blokového diagramu Functions → Structures → Sequence. Reprezentace této struktury připomíná jednotlivá políčka fotografického filmu jak je vidět na obrázku.



Obr. 28 Reprezentace struktury Sequence

Jednotlivá políčka můžeme přepínat stejně jako u struktury CASE šipkami v horní části rámečku nebo, jedná-li se jen o několik sekvencí, lze je řadit za sebe, tak jak mají být prováděny. Pravým tlačítkem si zvolíme *Add Frame After* resp. *Add Frame Before* pro přidání nového rámce za resp. před stávající. Smazání provedeme *Delete This Frame* po klepnutí pravým tlačítkem myši na rámeček, který chceme smazat.

Poslední věcí, kterou je dobré u struktury Sequence vědět je, jak předáme hodnotu z jednoho rámce do druhého. K tomu nelze použít posunovací registr jako u cyklů, ale je potřeba použít lokálních proměnných. Pravým tlačítkem klepneme na rámeček sekvence a zvolíme *Add Sequence Local*. V dané sekvenci se nám vytvoří čtvereček do kterého nyní můžeme přivést hranu. Jakmile tak učiníme, čtvereček se změní na šipku vycházející z rámečku ven nebo dovnitř, podle toho jestli se jedná o vstupní nebo výstupní terminál.



Obr. 29 Praktický příklad struktury Sequence

Na příkladu z obrázku 29, lze vidět postupné sčítání dvou čísel za pomocí struktury SEQUENCE. Do prvního rámečku přivedeme číslo pět, ke kterému v dalším kroku rámeček číslo 2) přičteme hodnotu z jiného vstupního terminálu. Přenos hodnoty je zajištěno pomocí lokální proměnné. Výsledek se zobrazí na výstupním terminálu VYSLEDEK. Příklad lze nalézt na přiloženém CD pod názvem 04.vi.

6.2.5 Pořizování dat v LabVIEW

Pořizování dat (Data AcQuisition) zkráceně DAQ, zahrnuje získávání dat a generování fyzikálních signálů. Řešení DAQ systémů umožňuje velmi rychlé měření a zpracování dat za pomocí programového vybavení. Takto vzniklá virtuální přístrojová technika kombinuje technické a programové vybavení s klasickými počítači. Program LabVIEW nám umožňuje číst, zpracovávat a zobrazovat získaná dat stejně pohodlně jako z programu generovat signál pro ovládání zařízení.

Získávání dat v LabVIEW lze vyhodnocovat podle několika informací: stavu (analogový nebo digitální signál), frekvenčního obsahu (frekvenční analýza) a úrovně (průběžná hodnota). K přenosu fyzikálních signálů (napětí, proud apod.) slouží měřící karta, která je spjata s programovým vybavením. Firma National Instruments neustále vyvíjí ovladače pro karty DAQ. Nyní používaný má název NI-DAQmx a nahradil předešlou verzi NI-DAQ.

MAX v LabVIEW

MAX neboli Measurement & Automation Explorer je aplikace v prostředí LabVIEW pro konfiguraci zařízení DAQ. MAX čte informace o ovladači zařízení ve Windows a přiřazuje zařízení logické jméno s nímž následně LabVIEW pracuje.

Program Measurement & Automation Explorer spustíme v hlavním menu LabVIEW zvolením *Tools → Measurement & Automation Explorer*.

Nainstalovaný ovladač NI-DAQmx rovněž umožňuje provádět simulaci bez nutnosti skutečné přítomnosti zařízení DAQ. Klikneme pravým tlačítkem myši na položku *Device and Interfaces → Create New.* Poté vybereme *NI-DAQmx Simulated Device* a ukončíme stisknutím *Finish.* Nyní lze zvolit v nabídce menu *Choose Device* zařízení DAQ, na kterém si můžeme simulovat chování bez nutnosti připojeného fyzického zařízení.

Úlohy v MAX

Vytvoření nové úlohy pomocí NI-DAQmx je jednoduché. Úlohy se v terminologii LabVIEW nazývají Task. Klikneme pravým tlačítkem na *Data Neighborhood* a volbou *Create New* zvolíme úlohu *NI-DAQmx Task.* Stiskem *Next* přejdeme na obrazovku konfigurace virtuálního kanálu. Zde kanál zvolíme konkrétní vstup nebo výstup karty DAQ. Posledním krokem před vytvořením nové úlohy je zadání názvu úlohy. Po zadání názvu a stisknutí *Finish* máme novou úlohu plně nakonfigurovanou.

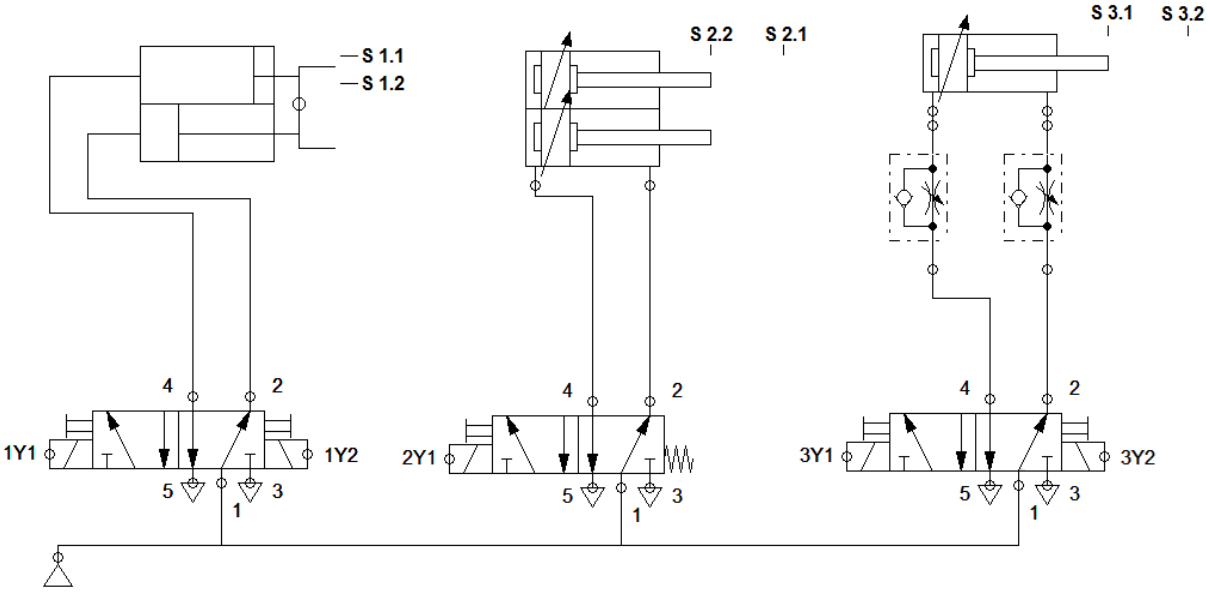
Jinou možností jak vytvořit úlohu je pomocí DAQ asistentu. Ten se nachází na paletě blokového diagramu Functions → Measurement I/O → DAQmx - Data Acquisiton → DAQ Assistant. V grafickém rozhraní nás systém navádí jak nakonfigurovat NI-DAQmx úlohu nebo NI-DAQmx kanál podobně jako tomu bylo v MAX. Po vytvoření úlohy se nám DAQ asistent zobrazí v blokovém diagramu a my s ním můžeme pracovat jak jsme zvyklí, připojováním vstupní a výstupních terminálů a vytvářet programy.

7 REALIZACE

Laboratorní stanoviště lze rozdělit na dva úseky. Prvním úsek se skládá z elektropneumatického manipulátoru a druhý úsek z pracovního PC. Obě pracoviště jsou propojeny pomocí DAQ karty PCIe-6251 firmy National Instruments a programu LabVIEW od téže firmy. V předešlé kapitole jsme se seznámili se základy programování v LabVIEW a v této kapitole si podrobněji přiblížíme a popíšeme jednotlivé prvky pracoviště. Veškeré datasheety a návody k použití nalezneme na přiloženém CD k této práci pod kódovým označením jednotlivých komponentů. Všechny komponenty lze také nalézt pod kódovým označením na stránkách firmy Festo http://www.festo.com/net/cs_cz/SupportPortal/, kde kromě manuálů a datasheetů, lze stáhnout 2D a 3D CAD soubory.

7.1 Manipulátor firmy Festo

Elektropneumatický manipulátor je poskládán z komponentů od firmy Festo. Sloužil jako demonstrace využití pneumatiky a elektroniky při přemisťování předmětů. Hnací sílu tvoří stlačený vzduch, přiváděný z centrálního rozvodu vzduchu v laboratoři a řídící elektroniky pro ovládání. Kompletní elektronika a logické signály jsou napájeny 24 V zdrojem přivedeným do svorkovnice a z ní dále rozvedeny do programovatelného automatu FC34 od firmy BECK a polohovacího kontroléru SPC200. Právě tyto dva řídící prvky mají být nahrazeny adekvátním řídícím systémem, proto byly z manipulátoru odstraněny a nahrazeny řízením přes DAQ kartu programem LabVIEW.



Obr. 30 Schéma zapojení pneumatických prvků

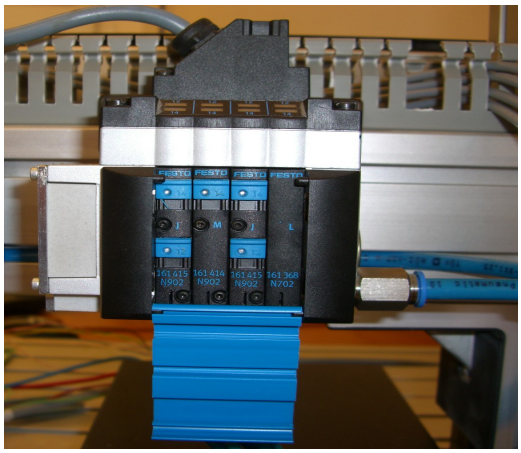
Před připojením jiného řídícího systému je nutné řádně prostudovat jednotlivé komponenty aby nedošlo při připojování k jejich poškození. Proto si uvedeme všechny prvky a přiblížíme si jejich funkci na manipulátoru.

Vzduch je přiváděn do manuálního spínacího ventilu *HE-D-MINI* následně pak projde redukčním ventilem s filtrem na odstranění nečistot *LFR-14-D-5M-MINI*, na kterém je umístěn elektronický tlakový senzor pro snímání tlaku *SDE1-D10-G2-R18-C-P1-M8*. Posledním prvkem při vstupu je opět spínací ventil, tentokrát ovládaný elektronicky *HEE-D-MINI-24* kabelem s elektronikou a signalizační led diodou *KMEB-1-24-2,5-LED*. Dále je vzduch rozveden do elektropneumatického rozvaděče a proporcionálního ventilu.

Obr. 31 Ovládací prvky na vstupu stlačeného vzduchu, přívod vzduchu vlevo

7.1.1 Ventilový terminál

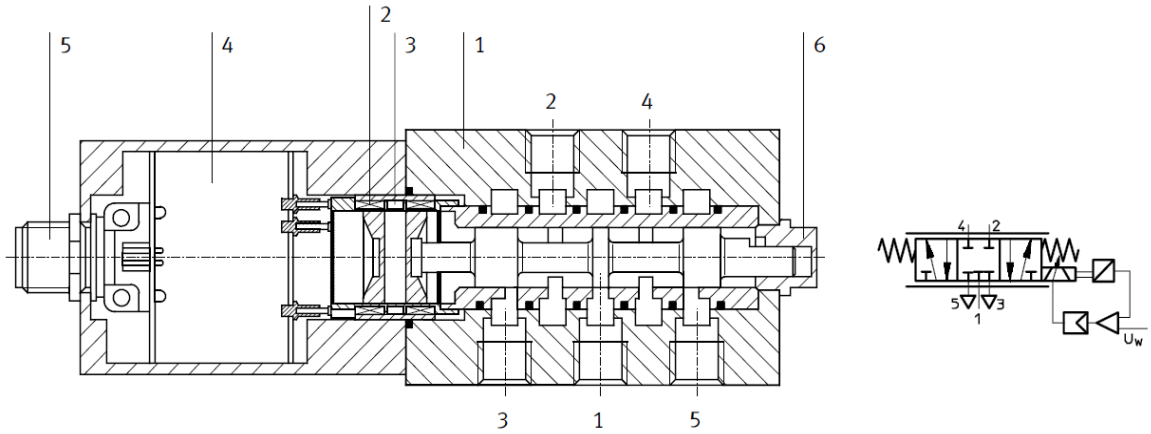
Katalogové označení terminálu *CPV10-GE-MP-4*. Je výhradně navržen pro sjednocení pneumatických ventilů ovládaných elektronicky. Propojení zajišťuje multipinový konektor a přivádí k jednotlivým ventilům elektronické signály pro jejich ovládání. Terminál je variabilní tzn., že je do něj možné připojil libovolné množství pneumatických ventilů. Tento konkrétní terminál nainstalovaný na manipulátoru je však konstruován pouze na 4 elektropneumatické ventily. Jak lze vidět na obrázku jsou obsazeny jen tři pozice a to dvěma dvojčinnými a jedním jednočinným ventilem.



Obr. 32 Terminál pro umístění pneumatických rozvaděčů

Dvojčinné ventily *CPV10-M1H-5JS-M7* jsou určeny k ovládání příčného pneumotoru a k sevření resp. rozevření úchopové hlavy (dále označované jako chapadlo). Jednočinný ventil *CPV10-M1H-5LS-M7* (v terminálu na obrázku 32 poznáme snadno podle jednoho modrého přepínače) je určen pro ovládání pneumatických saní.

7.1.2 Proporcionální ventil

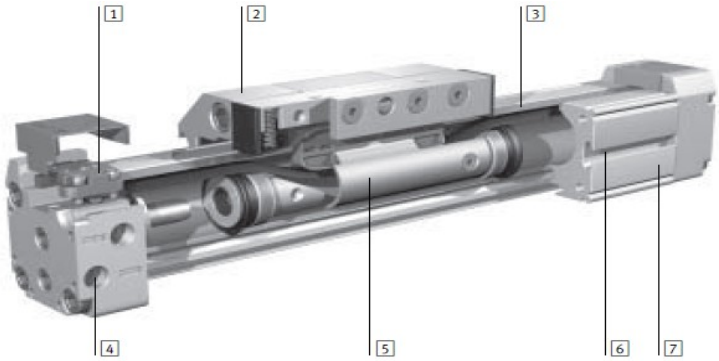


Obr. 33 Funkční řez proporcionálního ventilu a jeho schématická značka [14]

Součásti ventilu	Vývody pro vzduch
1 – těleso ventilu 2 – kotva s pohonem 3 – snímač 4 – elektronika 5 – konektor 6 – krycí víčko	1 – přívod stlačeného vzduchu 2 – přívod do pracovního prostoru A 3 – odvzdušnění pracovního prostoru A 4 – přívod do pracovního prostoru B 5 – odvzdušnění pracovního prostoru B

Proporcionální ventil se hodí vždy v případě, chceme-li spojitě řídit pneumatický píst. Pro potřeby našeho manipulátoru byl vybrán proporcionální průtokový ventil *MPYE-5-1/8-HF-010-B*, který ovládáme přímo a převádí analogové vstupní napětí v rozmezí 0 až 10 V na vhodný průřez průtoku vzduchu.

7.1.3 Lineární pohon



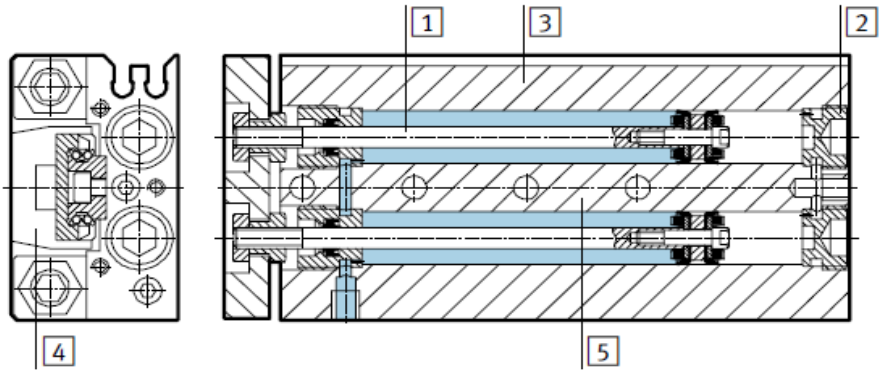
Obr. 34 Řez přímočarého pohonu DGPIL [15]

1 – nastavitelné tlumení v koncových polohách
2 – saně spojené s unašečem
3 – krycí pás, který chrání před nečistotami
4 – přívod stlačeného vzduchu
5 – píst
6 – upevňovací drážka (možnost využití pro upevnění snímačů)
7 – tělo pohonu

Katalogové označení *DGPIL-25-750-PPV-B-KF-AIF-GK-SV-AH.* Tento přímočarý pohon se vyznačuje nepřímím spojením unašeče a pístu jak lze vidět na obrázku 34. Mezi další kladné vlastnosti patří variabilní montáž, kompaktní rozměry při zachování velkého zdvihu a bezúdržbový provoz. V koncových polohách jsou vzduchové tlumiče.

7.1.4 Pneumatické saně

Jedná se druh pneumatického pohonu vhodný díky svým malým rozměrům a velké adaptaci pro nejrůznější aplikace. Katalogové označení SLT-25-100-P-A. Nejčastěji se však vyskytuje ve spojení s chapadlem na manipulačních linkách. Dvojice pneumatických posuvů zaručuje i přes svoje malé rozměry dostatečnou tuhost.

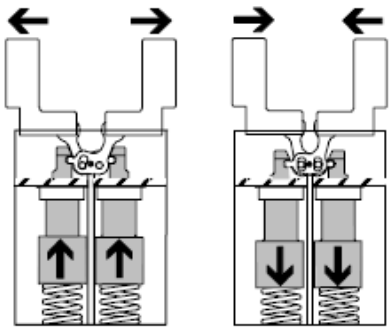


Obr. 35 Funkční řez [16]

1 – pístnice
2 – víko
3 – těleso
4 – saně
5 – vedení

7.1.5 Pneumatické chapadlo

Katalogové označení *HGP-10-A*. Na obrázku 36 můžeme vidět princip pohybu. Po přivedení vzduchu se pístnice pohybuje dopředu a dozadu. Pomocí mechanismu je převeden axiální pohyb pístu na čelisti. Chapadlo pracuje s dvojčinným pneumotorem.



Obr. 36 Princip pohybu čelistí [17]

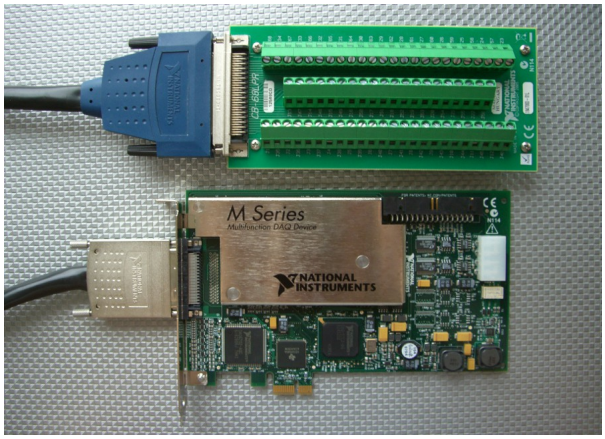
7.1.6 Senzory pneumatických motorů

Katalogové označení senzorů *SME-8-K-LED-24*. Bezkontaktní magnetický senzor vhodný pro drážku typu T snímající polohu pístu. Senzor se umisťuje podél drážky a je připojen pomocí kabelu do vícenásobného rozdělovače *MPV-E/A08-M8.* Propojení s DAQ kartou obstarává kabel *KMPV-SUB-D-15-5.*

7.2 Periferie PC

Komunikaci s počítačem obstarávají tři prvky:

- Svorkovnice CB-68LPR od firmy NI
- Propojovací kabel
- DAQ karta PCIe-6251

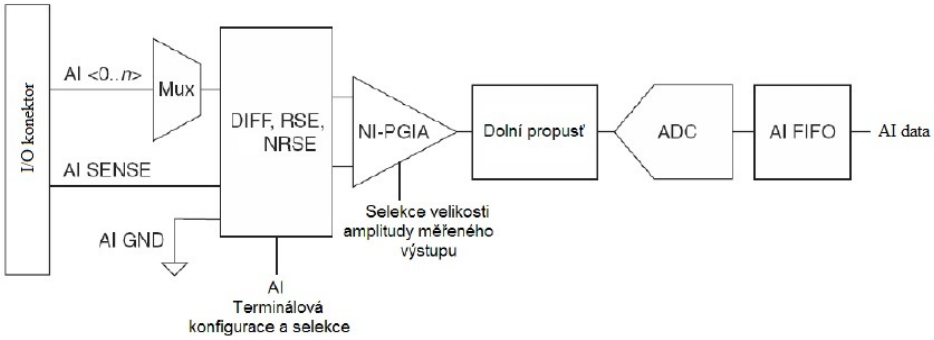


Obr. 37 Svorkovnice, propojovací kabel a DAQ karta

Číselné označení jednotlivých kontaktů na svorkovnici se shoduje s čísly pinů na kartě PCIe-6251 (obr. 37). Příklad: využijeme-li k propojení externího zařízení s kartou kontaktu, označeném na svorkovnici jako *J52,* pak podle obrázku 39 si můžeme najít, že se jedná o digitální port pojmenovaný P0.0, což značí v LabVIEW Port0/Line0. Více o vytvoření nové úlohy se dozvíme v kapitole 8.1.

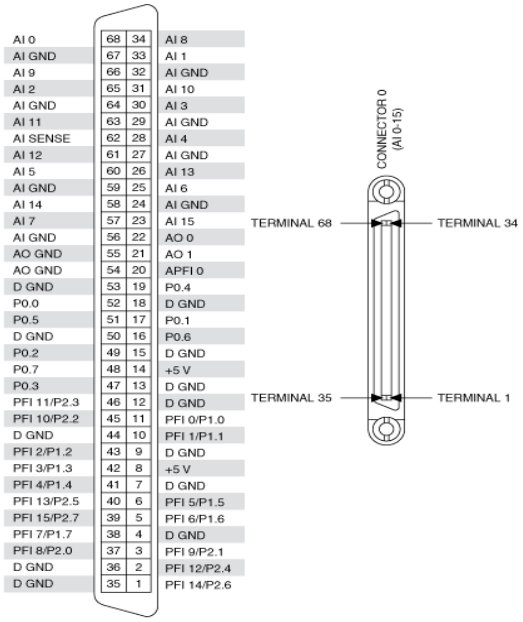
7.2.1 DAQ karta PCIe-6251

Pro získávání a generování signálů byla použita karta DAQ (Data AcQuisition) PCIe-6251 od firmy National Instruments (obr. 37). Karta disponuje 16 analogovými vstupy, 2 analogovými výstupy, 24 digitálními výstupy/vstupy a dvěma čítači. Analogové vstupy a výstupy jsou realizovány 16-bitovými A/D a D/A převodníky. Karta je zapojena v PC do PCI express.



Obr. 38 Blokové schéma karty PCIe-6251[18]

Pro určení správného zapojení pinů na kartě, můžeme vyhledat označení karty na stránkách firmy National Instruments. Nebo jednodušeji a spolehlivěji v kartě Measurement & Automation Explorer na pravé straně v nabídce klikneme *Devices and Interfaces → "Označení karty"*, kterou máme v počítači nainstalovanou. V horním liště hlavního okna klikneme na Device Pinouts a tím otevřeme okno, kde program sám najde instalovanou kartu a zobrazí její vstupní a výstupní porty. Konkrétní propojení pinů použité karty vidíme na obrázku 39.



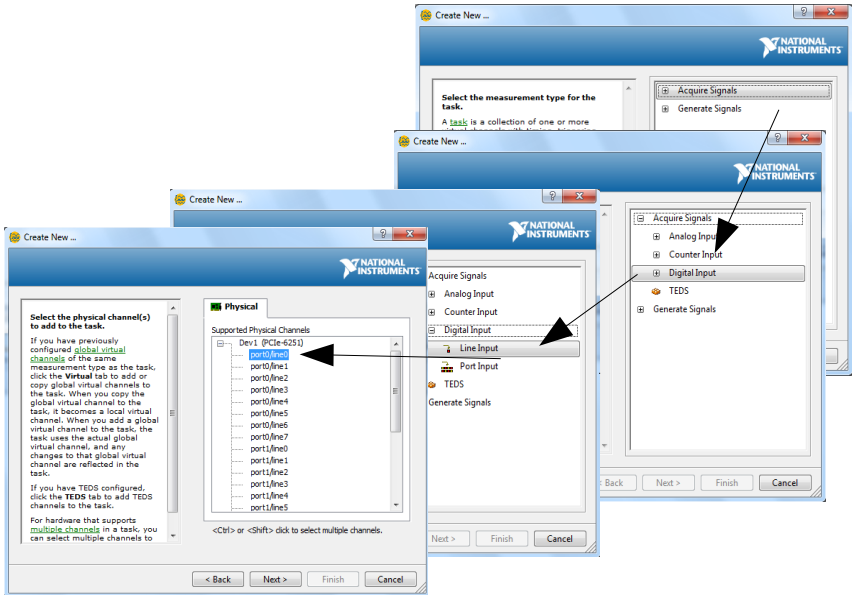
Obr. 39 Zapojení terminálů karty PCIe-6251 [19]

8 HARDWAROVÉ API

8.1 Vytvoření nové úlohy

Ovládání manipulátoru vyžaduje zapojení a nastavení vstupních a výstupních portů. Zapojení vidíme v tabulce 5. Vytváření nových úloh (v terminologii LabVIEW se setkáme s označením Task) jsme si teoreticky popsali v kapitole 6.2.5. V této kapitole se ukážeme vytvoření nové úlohy s DAQ PCIe-6251 na konkrétním případě.

V úvodní nabídce prostředí LabVIEW zvolíme *Create Project → Blank Project* a po klepnutí na *Finish,* vytvoříme svůj první projekt. Na obrazovce se nám zobrazí okno s názvem *Untitled Project.* K vytvoření nové úlohy klikneme pravým tlačítkem myši na *My Computer* a v roletové nabídce zvolíme *New → NI-DAQmx Task.* Po inicializaci se nám otevře okno pro konfiguraci nového fyzického kanálu. V průvodci nastavíme digitální vstup *Acquire Signals → Digital Input → Line Input → Port0/line0* (obr. 40). Posledním krokem k vytvoření úlohy je zvolení jména. Kliknutím na *Finish* máme vytvořenou novou úlohu, která komunikuje po portu P0.0. Lze označit i více vstupů a sjednotit tak všechny pod jednu úlohu. Tímto zpřehledníme náš projekt a usnadníme orientaci v programu.



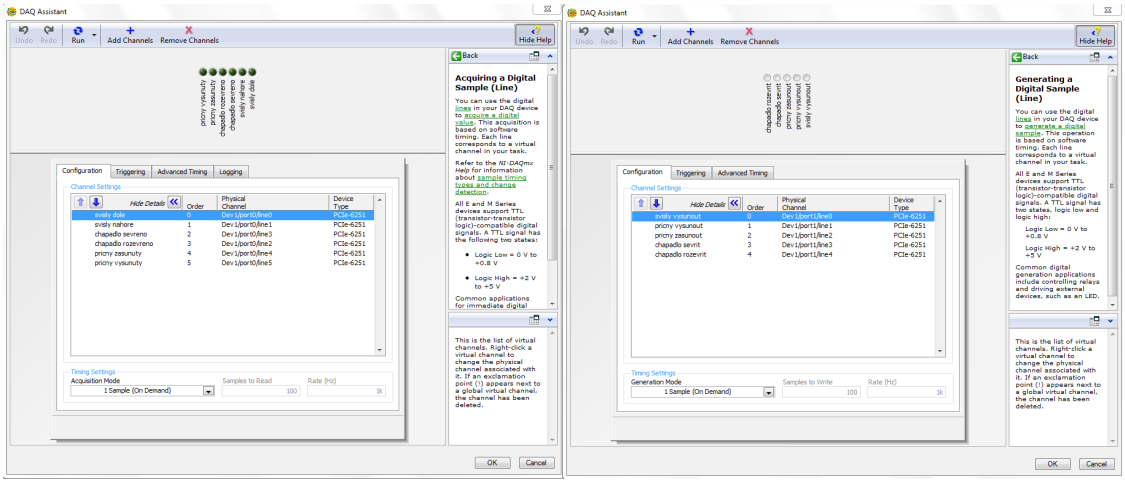
Obr. 40 Postup vytvoření úlohy

Takto nakonfigurovanou úlohu nyní vidíme na pravé straně okna našeho projektu. Kliknutím pravým tlačítkem na naši právě vytvořenou úlohu můžeme zvolit zda ji chceme odstranit, přejmenovat a nebo upravit. Analogicky lze vytvořil úlohy pro digitální nebo analogové vstupy a výstupy. Ve volbě *Acquire Signals* nebo *Generate Signals* zvolíme jaký typ signálu chceme číst nebo zapisovat na kartě a pak stačí jen správným způsobem zapojit.

Zapojení manipulátoru je prováděno přes svorkovnici CB-68LPR popsanou v kapitole 7.2. Zapojení jednotlivých elektronických komponentů na svorkovnici lze vidět v tabulce 5. Po zapojení svorkovnice můžeme začít vytvářet úlohy. V program si nejprve vytvoříme dvě úlohy pro čtení a zápis dat. *Digital in* je úloha, která sbírá data ze senzorů. Oproti tomu úloha *Digital out* zapisuje na výstup a přes svorkovnici ovládáme rozvaděče. Nastavení DAQ asistentu ukazuje obrázek 41.

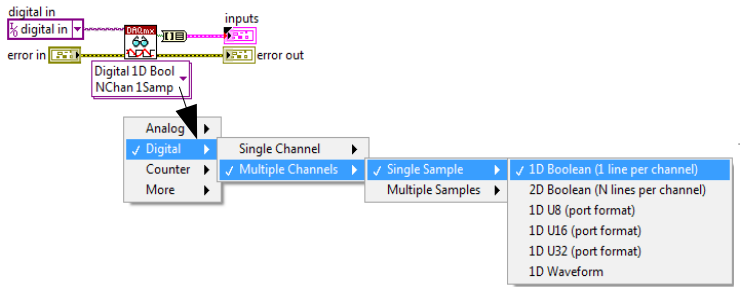
Označení	Zapojení do svorkovnice	Pin na DAQ kartě
1Y1	J42	P1.3
1Y2	J41	P1.4
2Y1	J11	P1.0
3Y1	J10	P1.1
3Y2	J43	P1.2
S1.1	J49	P0.3
S1.2	J47	P0.2
S2.1	J52	P0.0
S2.2	J17	P0.1
S3.1	J19	P0.5
S3.2	J51	P0.4

Tab. 5 Zapojení do svorkovnice



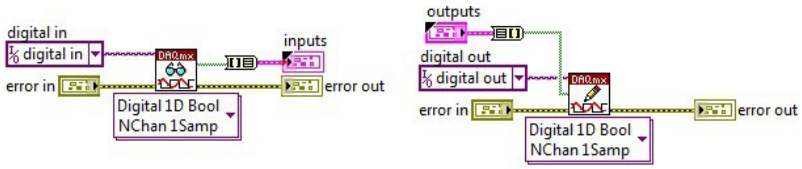
Obr. 41 Nastavení DAQ asistentu

Pro čtení z karty jsme si vytvořili samostatné VI a pojmenovali *Digital inputs Read.vi.* VI vytvoříme vložením z palety blokového diagramu *Function → Measurement I/O → DAQmx – Data Acquisition → DAQmx Read.* Ve funkci DAQmx Read nastavíme čtení jednoho vzorku ve více kanálech v datovém formátu 1D Boolean (obr. 42).



Obr. 42 Funkce DAQmx Read

Dále vložíme *DAQmx Task Name*, ve kterém zvolíme *Digital in* a spojíme jej se vstupem *task in*. *Task out* propojíme s terminálem typu Cluster, který bude zobrazovat hodnoty vstupních proměnných. Protože proměnná vystupující z *Data* je jiného typu je potřeba na hranu vložit převodník *Array To Cluster*. Obdobným způsobem vytvoříme VI pro zápis *Digital outputs Write.vi* s funkcí *DAQmx Read.* Obě kompletní VI ukazuje obrázek 43.

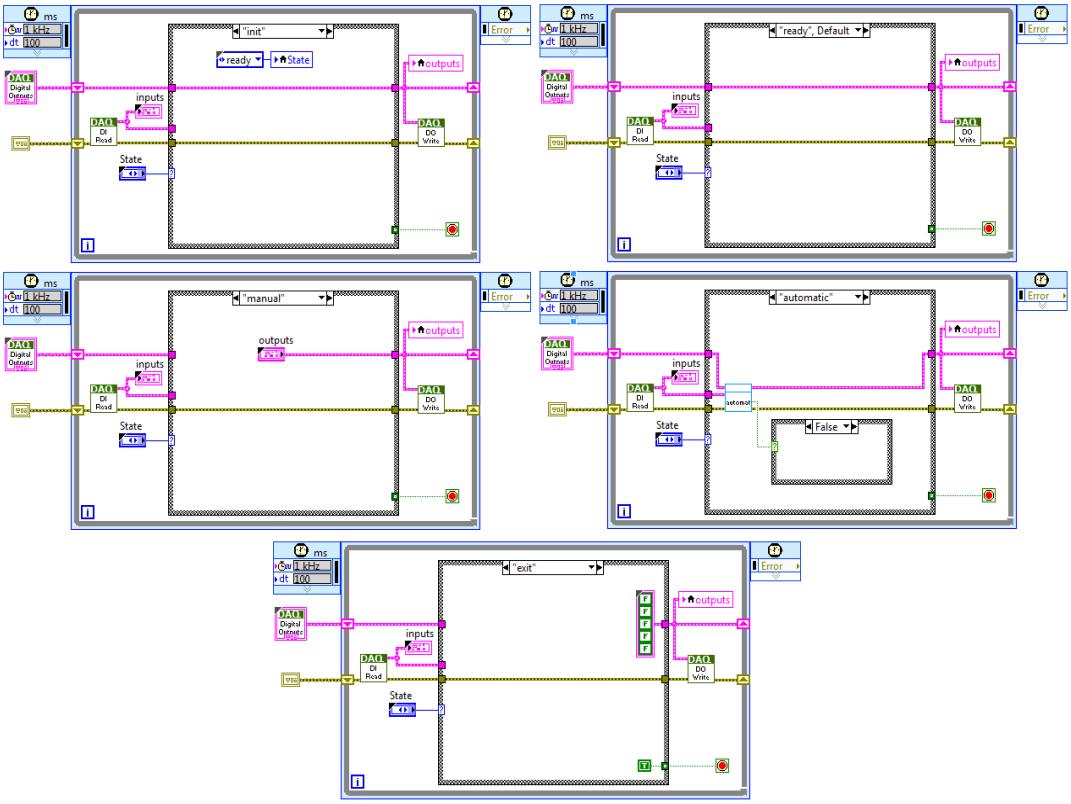


Obr. 43 VI čtení a zápisu digitální vstupů a výstupů

Přejděme k hlavní části vytvořeného projektu *main.vi*. Jak můžeme vidět na obrázku 44, *main.vi* se skládá z časové smyčky a struktury CASE. Struktura je definovaná pěti stavy. Init, Ready, Manual, Automatic a Exit. Volba stavů se provádí terminálem *State* v čelním panelu *main.vi* nebo *main States.vi.* Každý terminál řeší jiný stav.

Do blokového diagramu nového VI, vložíme časovou smyčku *Function → Structures → Timed Structures → Timed Loop.* Otevřeme konfigurační okno časové smyčky kliknutím na hodiny vlevo a z roletového menu zvolíme *Configure Input Node*. Periodu nastavíme na hodnotu 100. To je vše co budeme měnit a oknu můžeme zavřít. Uvnitř časové smyčky vytvoříme strukturu CASE z *Function → Structures → Case Structures.* Ve které vytvoříme pět stavů popsaných níže a každý blok struktury vyplníme podle obrázku 44. Jako výchozí okno zvolíme *ready.* Dále do VI vložíme *Cluster* obsahující výstupy, vytvořené VI *Digital outputs Write* a *Digital intputs Write.* Pro určování stavů jsme si vytvořili datovou typ Main States. Následně vše propojujeme tak jak je tomu na obrázku 44.

- *Init* je stav struktury, kde dochází k inicializování výstupních hodnot. Zavádí se pro nastavení výchozí polohy pneumatických členů. Základní poloha příčného pohonu je 'zasunuto', chapadla 'rozevřeno' a pneumatické saně se nachází v horní poloze (píst zasunutý). Základní poloha pneumatických prvků je zároveň výchozím bodem pro start programu, proto by se pro tento příklad neměla přenastavovat. Po inicializaci se přechází do stavu *Ready.*
- *Ready* neprovádí žádné speciální úkoly. Jedná se o *Default* (výchozí) stav a pouze přemosťuje nastavení výstupů.
- *Manual* slouží k manuálnímu ovládání manipulátoru. Ovládání probíhá přes terminál *outputs.*
- *Automatic* je stav struktury CASE pro uvedení manipulátoru do automatického režimu. Manipulátor pracuje v automatickém režimu samostatně podle vytvořeného programu v *Automatic.vi*, který si přiblížíme podrobněji v kapitole 9. Stav *automatic* se provádí tak dlouho, dokud se na rozhodovací proměnou vnitřní struktury nepřivede logická hodnota TRUE. Stane-li se tak, program se opět přepne do stavu *ready.*
- *Exit* podobně jako *init,* nastavuje pneumatické prvky manipulátoru do polohy jakou si navolíme. Nastavení není doporučené měnit a mělo by být shodné s nastavení výchozí polohy aby nedocházelo ke kolizím.



Obr. 44 Hardwarového API - jednotlivé stavy

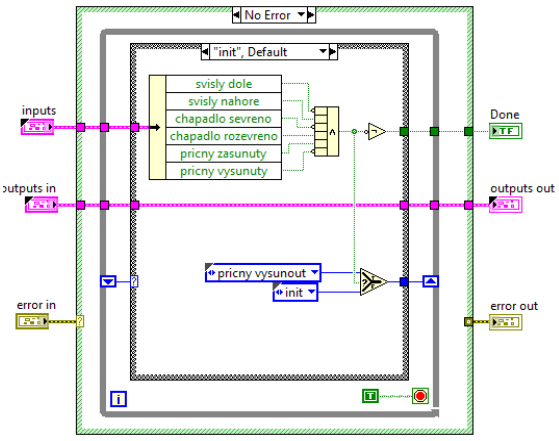
9 REALIZACE ÚLOHY

Závěrem zhodnotíme získané znalosti z předchozích kapitol na jednoduché úloze. Uživatel si může podle tohoto návodu lehce zkonstruovat svoji vlastní praktickou úlohu, popřípadě navázat na již vytvořený řídící systém a dále s ním pracovat a upravovat jej. Tato úloha je ve formě projektu i s vytvořeným API v předešlé kapitole, uložena na přiloženém CD s názvem Manipulator_DP.lvproj. Aby vše pracovalo správně je nutné jej otevírat jako projekt, nikoliv jako samostatné soubory VI. Projekt byl vytvořen v NI LabVIEW verze 12.0f5 (32-bit).

Posledním VI v našem projektu je Automatic.vi. Zde si uživatel může nadefinovat program pro manipulátor. Pro ukázku jak lze takový program vytvořit je již v tomto projektu jeden program přednastaven. Jedná se o jednoduchou úlohu, kdy nejprve zkontrolujeme, zda se všechny písty nachází ve výchozí pozici. Následuje v postupných krocích vysunutí příčného pístu, vysunutí pneumatických saní, sevření chapadla, zasunutí pneumatických saní a zasunutí příčného pístu. Každý krok není uskutečněn, dokud není předchozí krok zcela dokončen, tzn. dokud se píst nenachází ve správné poloze, následující krok se nevykoná. Určení správné polohy signalizují senzory, které jsou jako vstupní proměnné *Inputs* přiváděny do smyčky.

VI je vytvořené ze třech rámečků. Vnější rámeček tvoří struktura CASE a slouží pro detekci chyby. Jakmile se v běhu programu vyskytne chyba, program se ihned zastaví. Smyčka *While* nacházející se uprostřed mezi dvěma strukturami CASE nám jen zajišťuje kontinuální běh. Vnitřní rámeček je opět struktura CASE, která obsahuje jednotlivé kroky uživatelsky navoleného programu. Jednotlivé kroky si nyní popíšeme.

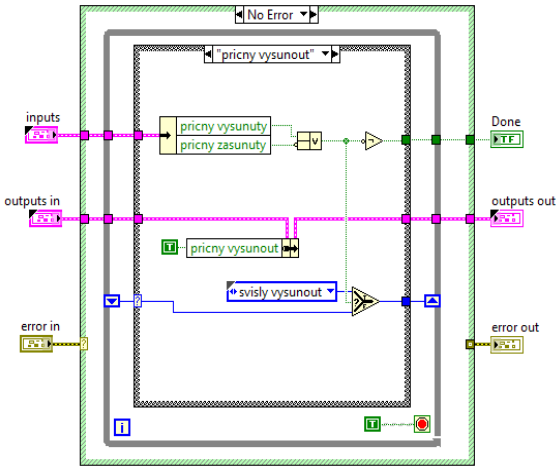
- Krok 1. Z terminálu *inputs* nám vstupují hodnoty senzorů. Pokud jsou písty správně postaveny ve výchozí poloze, tedy příčný píst a pneumatické saně jsou zasunuty a chapadlo je rozevřeno, pak člen *Compound Arithmetic,* nastavený na logickou funkci AND, vrátí logickou hodnotu TRUE, jinak FALSE. Tato hodnota dále vstupuje do výběrového členu *Select* a zároveň negovaná do terminálu *Done.* Je-li hodnota terminálu *Done* True, program se ukončí. V opačném případě se hodnota porovná v *Select.* Hodnota TRUE určí, že se má pokračovat dalším krokem, hodnota FALSE nastavuje strukturu CASE na default.



Obr. 45 Krok 1

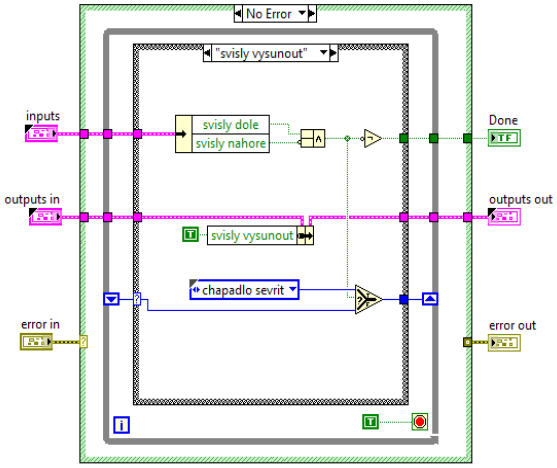
- Krok 2. Je-li úspěšně splněna podmínka prvního kroku, přejde se k vykonání kroku druhého. V tomto kroku se nejprve vykoná příkaz v objektu *Bundle By Name,* který způsobí vysunutí příčného pístu. Následuje ověření zda je příčný píst vysunutý. Do *Compound Arithmetic* vstupují proměnné ze senzorů příčného pneumatického

pohonu. Další postup je identický s postupem v kroku 1.



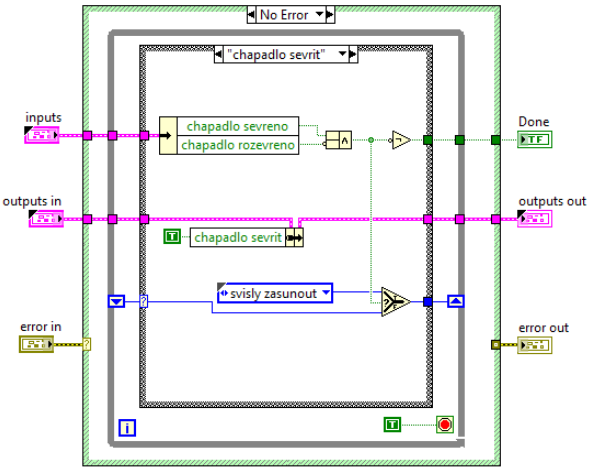
Obr. 46 Krok 2

- *Krok 3.* Neukončí-li se program, pokračuje krokem 3. Svislé pneumatické saně se vysunou. Další postup je analogický z předchozím s tím rozdílem, že zde ověřujeme podmínku zda jsou pneumatické saně vysunuté. Pokud ano, program stále pokračuje.



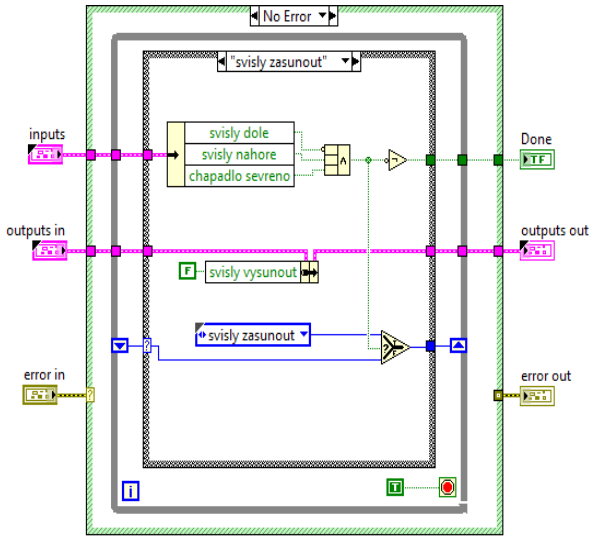
Obr. 47 Krok 3

- *Krok 4.* V tomto kroku program sevře chapadlo a ověří zda se tak skutečně stalo.



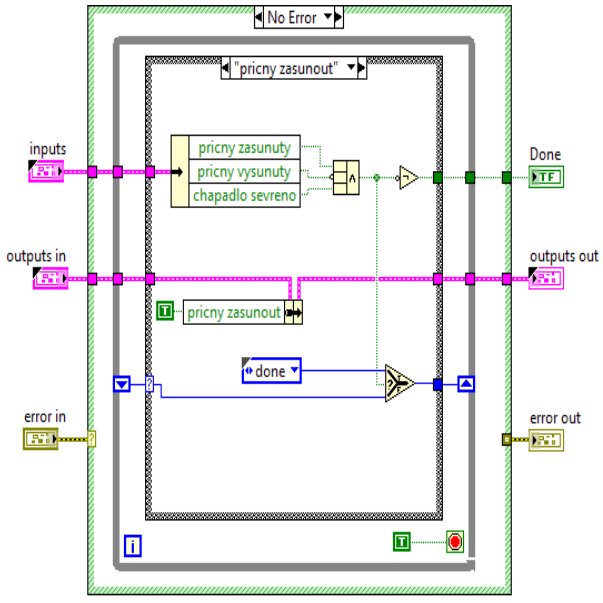
Obr. 48 Krok 4

- *Krok 5.* V kroku pět se vracíme zpět do základní polohy s chapadlem sevřeným. Pneumatické saně se zasunou. Nyní kontrolujeme i stálé sevření chapadla.



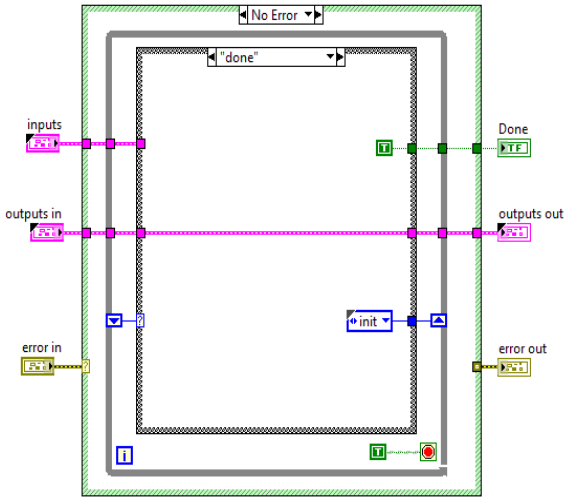
Obr. 49 Krok 5

- *Krok 6* Příčný píst se zasune a ověří se poslední podmínka zda jsou všechny písty zasunuty a zda je chapadlo stále sevřeno. Pokud je tato podmínka splněna přejde se na poslední blok programu.



Obr. 50 Krok 6

- *Krok 7. Posledním krokem je ukončení programu nastavením terminálu Done* na hodnotu TRUE, což způsobí návrat struktury CASE v *Automatic.vi* na default a v *main.vi* se nastaví terminál *State* na hodnotu *Ready.*



Obr. 51 Krok 7

Tímto jsme si ukázali příklad, jak lze tvořit vlastní program pro ovládání pneumatického manipulátoru. Přidáním dalších rámců struktury CASE lze vytvářet složitější programy se stejným postupem jako je tomu v tomto příkladě.

10 ZÁVĚR

Cílem této diplomové práce byl návrh univerzálního řídícího systému pneumatického manipulátoru, určeného pro praktické využití v předmětu tekutinové prostředky automatického řízení. Postup práce spočíval ve zmapování zapojení původních řídících prvků, kterými byl programovatelný automat FC34 a polohovací kontrolér SPC200. Řízení obou prvků nebylo možné žádným dostupným způsobem upravovat a tak bylo rozhodnuto je zaměnit za systém variabilnější a otevřenější. Následovalo odpojení a demontování z manipulátoru. Z možných variant, jak ovládat pneumatické a elektro-pneumatické prvky, bylo vybráno prostředí NI LabVIEW od firmy National Instruments. Prostředí je dostatečně otevřené a snadné na editaci. Zapojení prvků ve svorkovnici je možné jednoduše měnit a upravovat dle potřeb uživatele.

Pro lepší orientaci při návrhu řídícího systému jsou v kapitole 6 uvedeny základní postupy pro tvorbu programu v prostředí NI LabVIEW. Kapitola obsahuje též jednoduché příklady, které je možné nalézt na přiloženém CD. Kapitola 7 se věnuje pořizování a zápisu dat přes DAQ kartu PCIe-6251. Použití karty je ukázáno na praktických příkladech ovládání pneumatického manipulátoru. V poslední kapitole je ukázán jeden ze způsobů jak je možné sestavit úlohu pro manipulátor.

Hlavní cíl práce, kterým bylo navržení a otestování nového řídícího systému pro praktickou výuku byl splněn. Nyní je možné ve výuce přistoupit i k praktickému využití pneumatických pohonů a ovládat manipulátor pomocí navrženého prostředí. Dále je také možnost nahrazení nynějšího řízení jiným, např. propojení s některým z dostupných programovatelných automatů s využitím zmiňovaných programovacích jazyků.

SEZNAM POUŽITÉ LITERATURY

[1] SMC Industrial Automation CZ s.r.o., SMC_SKRIPTA_CZ_new.IDD – LG1_ Einfuehrung.pdf [online]. 2009 [cit. 24. května 2013]. Dostupné z WWW: http://2009.oc.smc-cee.com/sk/pdf/LG1_Einfuehrung.pdf

[2] Bureau International des Poids et Mesures, [online]. [cit. 24. května 2013]. Dostupné z WWW: http://www.bipm.org/en/CGPM/db/11/12/

[3] Wikipedia.cz, [online]. 2013 [cit. 24. května 2013]. Dostupné z WWW: http://cs.wikipedia.org/wiki/Atmosf%C3%A9rick%C3%BD_tlak

[4] SMC Industrial Automation CZ s.r.o., Vlastnosti stlačeného vzduchu [online]. 2009 [cit. 24. května 2013]. Dostupné z WWW: http://2009.oc.smc-cee.com/sk/pdf/LG2_Masseinheit.pdf

[5] Škorpík, Jiří. Škrcení plynů a par [online]. 2006 [cit. 24. května 2013]. Dostupné z WWW: http://www.transformacni-technologie.cz/skrceni-plynu-a-par.html

[6] SMC Industrial Automation CZ s.r.o., Pneumatické lineární pohony [online]. 2009 [cit. 24. května 2013]. Dostupné z WWW: http://2009.oc.smc-cee.com/sk/pdf/LG1_Antriebe.pdf

[7] Němec, Z. Prostředky automatického řízení: elektrické, 2. vyd. Brno 2008

[8] ČSN EN 61131-1: 2004. Programovatelné řídicí jednotky - Část 1: Všeobecné informace. Praha: Český normalizační institut, 2004. 28 s

[9] Vančura, Tomáš. Pneumatické akční členy a jejich řízení, [cit. 24. května 2013]

[10] PLC Fundamentals. [online]. 2013 [cit. 24. května 2013]. Dostupné z WWW: http://www.quia.com/pages/saimasaleem/ate326

[11] Katalog, Základy elektropneumatiky, Festo Didactic GmbH & Co., Denkendorf (EN). [CD-ROM] Festo, 2000

[12] SMC Industrial Automation CZ s.r.o., Pneumatické lineární pohony [online]. 2009 [cit. 24. května 2013]. Dostupné z WWW: http://2009.oc.smc-cee.com/sk/pdf/LG1_Antriebe.pdf

[13] Čermák, M. Moderní měřící systémy. Brno, 2003. 85 s. Diplomová práce na Přírodovědecké fakultě Masarykovy univerzity. Vedoucí diplomové práce RNDr. Zdeněk Bochníček, Dr.

[14] VOLF, M. Přesné polohování pneumatických pohonů. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010. 72s.

[15] Festo, s.r.o., Přímočaré pohony DGP/DGPL [online]. 2007 [cit. 24. května 2013]. Dostupné z WWW: http://xdki.festo.com/xdki/data/doc_CS/PDF/CZ/DGP_CZ.PDF

[16] Festo, s.r.o., Pneumatic sliding unit type SLT [online]. [cit. 24. května 2013]. Dostupné z WWW: http://www.festo.com/net/SupportPortal/Files/196862/704179d2.pdf

[17] Festo, s.r.o., Parallel Gripper [online]. [cit. 24. května 2013]. Dostupné z WWW: http://www.festo.com/net/SupportPortal/Files/51239/682514d6.pdf

[18] National Instruments (Czech Republic), s.r.o., Katalog produktů [online]. 2009 [cit. 24. května 2013]. Dostupné z WWW: https://www.ni.com

[19] Katalog NI LabVIEW, Measurement & Automation Explorer, NI-DAQmx Device Terminals Help