VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

MECHANICKÝ MANIPULÁTOR ŘÍZENÝ Z PLATFORMY CRIO

MECHANICAL MANIPULATOR CONTROLLED FROM THE CRIO PLATFORM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — David Karásek

AUTHOR

VEDOUCÍ PRÁCE — Ing. Zdeněk Havránek, Ph.D.

SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: David Karásek ***ID:*** 152698
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Mechanický manipulátor řízený z platformy cRIO

POKYNY PRO VYPRACOVÁNÍ:

1) Zpracujte přehled hardwarových možností řízení pohonů s krokovými motory s využitím platformy cRIO a softwarových nástrojů LabVIEW FPGA a SoftMotion firmy National Instruments pro vytváření pokročilých řídicích systémů pro tyto pohony.
2) Navrhněte hardwarovou i softwarovou strukturu řídicího systému s kontrolerem cRIO pro realizaci kombinovaného mechanického manipulátoru s dvěma translačními pohony a jedním rotačním pohonem.
3) Sestavte hardware mechanického manipulátoru s využitím pohonů firmy Standa, inteligentních krokových motorů ISM-7402 a I/O modulů NI 9501 a SISU-1004. Doplňte systém o automatickou detekci výchozích a koncových poloh pomocí spínačů a digitálního modulu NI 9401.
4) Realizujte řídicí aplikaci pro systém cRIO a ovládací/vizualizační aplikaci pro počítač, obojí v prostředí LabVIEW, které umožní pokročilé řízení manipulátoru.
5) Prakticky ověřte funkčnost realizovaného systému, zejména využití limitních spínačů, a dále možnost realizace složitějších trajektorií (synchronní řízení více pohonů).

DOPORUČENÁ LITERATURA:

[1] Vlach, J., Havlíček, J., Vlach, M. Začínáme s LabVIEW. 1. vyd. Praha: BEN - technická literatura, 2008. ISBN 978-80-7300-245-9.

[2] National Instruments: Getting Started with CompactRIO and LabVIEW. 31 s., 2009. Dostupné z http://www.ni.com/pdf/manuals/372596b.pdf.

Termín zadání: 6.2.2017 ***Termín odevzdání:*** 29.5.2017

Vedoucí práce: Ing. Zdeněk Havránek, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

Abstrakt

Tato bakalářská práce si klade za cíl vytvořit řídicí software pro mechanický 2 D manipulátor skládající se ze dvou translačních a jednoho rotačního členu. V teoretické části rozebírá možné hardwarové řešení a přibližuje základní pojmy použité v praktické části, praktická část pojednává o návrhu aplikace. Výsledný program je plně funkční, na konci práce jsou však předestřeny další možnosti, jak lze program dále vylepšit.

Klíčová slova

CompactRIO, LabVIEW, operační systém reálného času, FPGA, Front panel, Block diagram, krokový motor, manipulátor

Abstract

This bachelor thesis is about designing and creating control software for 2 D mechanical manipulator. This device consists two translational actuators and one rotational actuator. At theoretical part author discuss possible hardware solutions and talks about basic terms used in practical part. Practical part is about designing main control application. Final program is fully functional, at the end of this thesis the author suggests some improvements which could be done to the program.

Keywords

CompactRIO, LabVIEW, real-time operating system, FPGA, Front panel, Block diagram, stepper motor, manipulator

Bibliografická citace:

KARÁSEK, D. *Mechanický manipulátor řízený z platformy cRIO.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 52 s. Vedoucí bakalářské práce Ing. Zdeněk Havránek, Ph.D..

Prohlášení

Prohlašuji, že svou závěrečnou práci na téma Mechanický manipulátor řízený z platformy cRIO jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne 26. května 2017 ……

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Zdeňku Havránkovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích a vypracování mé bakalářské práce.

V Brně dne 26. května 2017 ……

podpis autora

Seznam Obrázků

Seznam tabulek

Obsah

1 ÚVOD

Cílem této práce je návrh a realizace software pro manipulátor řízený jednotkou CompactRIO. Tento manipulátor se má skládat ze dvou translačních a jednoho rotačního krokového motoru, které jsou řízeny zařízením cRIO se třemi vloženými I/O kartami.

Teoretická část práce se dělí na 3 hlavní oblastí, z nichž první stručně vysvětluje princip fungování krokových motorů a způsob jejich zapojení a řízení, Druhá oblast se zabývá analýzou hardwarových možností řízení pohonů s krokovými motory, které firma National Instruments k platformě cRIO nabízí. Ve třetí podkapitole jsou blíže vysvětleny základní pojmy týkající se prostředí LabVIEW, které se vyskytují v praktické části práce.

Praktická část se nejdříve zaměřuje na porovnání poskytnutého hardwaru s tím, který je v teoretické části vyhodnocen jako nejvhodnější. Dále rozebírá propojení I/O karet s krokovými motory a s koncovými snímači. Poslední oblast se zabývá realizací řídicí/vizualizační aplikace. Tato oblast se dále dělí na dvě hlavní části, a to na vytváření konfigurace FPGA a vysvětlení algoritmů hlavního ovládacího programu.

2 TEORETICKÁ ČÁST

Teoretická část práce se primárně zabývá analýzou hardwaru, který je v manipulátoru použit, a základními principy programovacího prostředí LabVIEW. První oblast teoretické části stručně vysvětluje principy fungování krokových motorů a způsob jejich zapojení a řízení. Druhá podkapitola analyzuje hardwarové možnosti, které jsou firmou NI nabízeny na jejích internetových stránkách, poslední oblast velmi stručně přibližuje prostředí LabVIEW, zabývá se zejména pojmy, které jsou použity v praktické části práce.

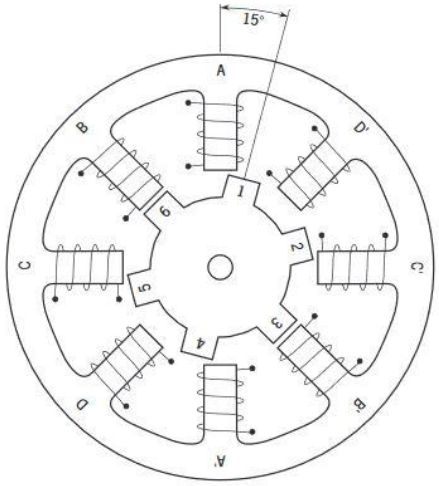
2.1 Krokové motory

Někdy je v praxi nutné, aby se motor posunul na definovanou polohu a tuto polohu držel, čehož se dá docílit dvěma způsoby. První možností je použití klasického motoru (např. servomotoru), od kterého je zavedena zpětná vazba, která sleduje aktuální polohu – nejčastěji resolver. U takového řešení se však objevují jisté komplikace. Zpětná vazba celé řešení prodražuje a zesložiťuje, v některých speciálních případech dokonce není možná. Největší nevýhoda spočívá v neschopnosti motoru po najetí na pozici polohu „držet silou“, toho se musí docílit opět dalším zařízením. Druhou možností je použití krokových motorů. Krokové motory oproti tomu už z principu dosaženou pozici dokáží udržet a nepotřebují ani speciální zpětnou vazbu. Díky těmto vlastnostem jsou pro zadaný manipulátor ideálními akčními členy.

2.1.1 Základní princip

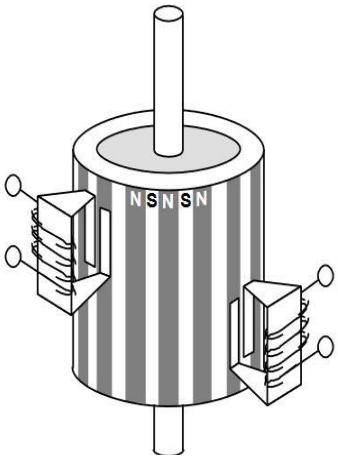
Jako každý motor se i krokové motory skládají z rotoru a statoru. Stator se skládá z několika vhodně zapojených cívek, které vytvářejí magnetické pole. Podle typu rotoru se krokové motory rozdělují na 3 hlavní typy.

Rotory motorů s proměnnou reluktancí (Obr. 1) jsou nejméně používané. Rotor neobsahuje žádné vinutí, skládá se pouze z několika zubů. Připojením stejnosměrného napájení se póly magnetizují a rotor se pootočí tak, aby jeho „zuby“ byly co nejblíže k napájeným částem statoru. Tyto motory nejsou příliš přesné a dokáží vyvinout pouze malý točivý moment.



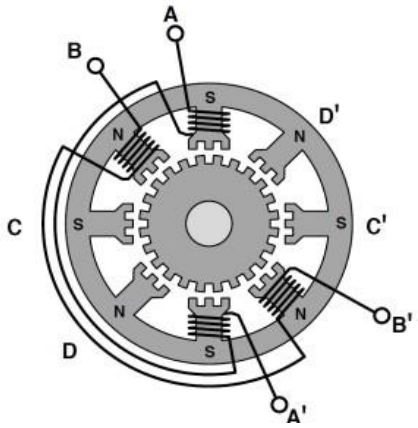
Obr. 1 Náčrt principu krokového motoru s proměnnou reluktancí [7]

Dalším typem motorů jsou motory s permanentním magnetem (Obr. 2). Zde se rotor skládá z radiálně pólovaného permanentního magnetu, kde je počet pólů poloviční oproti počtu pólů statoru. Rotory tohoto typu nemají vůbec žádné zuby. Oproti motorům s proměnnou reluktancí dokáží motory s permanentním magnetem vyvinout větší moment, jeho výkon však i tak není příliš závratný. Zároveň tyto motory nedokáží dosáhnout vysokého počtu kroků na otáčku, minimální krok se většinou pohybuje mezi 15° - 7,5°. Díky své jednoduché konstrukci jsou tyto motory levné a poměrně hojně používané.



Obr. 2 Náčrt principu krokového motoru s permanentním magnetem [7]

Posledním typem motorů jsou tzv. hybridní (Obr. 3), které jsou spolu s motory s permanentním magnetem nejčastěji používané. Jak už název napovídá, hybridní motory mají rotor obsahující permanentní magnet. Rotor má tvar ozubeného kola. Tento motor kombinuje výhody obou předešlých typů, tj. má vysoký moment, je rychlý a zároveň je přesný – jeho krok se většinou pohybuje v rozmezí 3,6° - 0,9°.



Obr. 3 Náčrt hybridního krokového motoru [8]

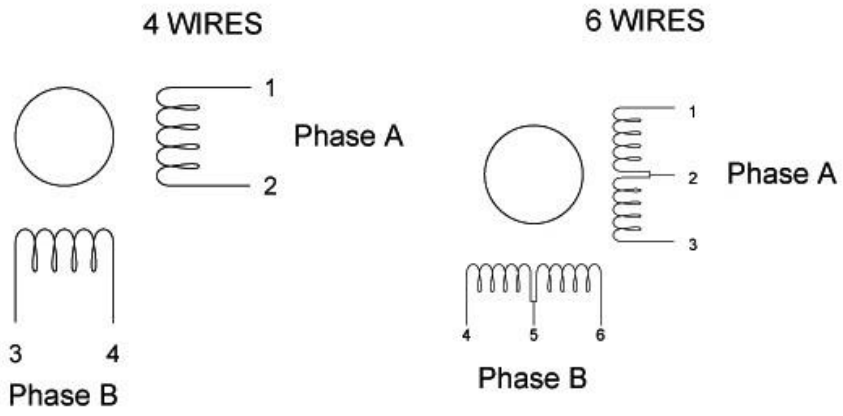
2.1.2 Metody řízení

U krokových motorů se používají dva základní typy zapojení – unipolární a bipolární. Při unipolárním zapojení je buzena pouze jedna cívka. Díky tomu má motor malý odběr energie a k jeho řízení stačí jednoduchý obvod, ale zároveň má malý moment.

Bipolární zapojení naopak znamená, že jsou buzeny vždy dvě protilehlé cívky tak, aby měly vždy opačné magnetické pole. Motor má tím pádem vyšší spotřebu energie a vyžaduje již složitější řídicí obvody, jeho moment je oproti unipolárnímu zapojení větší.

Dále se motory dají dělit na jednofázové, dvoufázové a vícefázové. U jednofázového řízení je buzena vždy jedna cívka (v případě bipolárního zapojení cívky dvě). Nevýhodou tohoto řešení je, že při jednofázovém řízení má motor menší kroutící moment. Dvoufázové řízení budí vždy dvě sousední cívky, čímž se zvyšuje moment motoru. Jestliže motor střídá jednofázové a dvoufázové řízení, dokáže se motor pohybovat po půl krocích. Vícefázové řízení už potřebuje složitější řídicí elektroniku, budí se zaráz ještě více cívek. Motory se díky němu dokáží pohybovat plynuleji a s ještě menším krokem – tzv. microstepping.

Dále se krokové motory liší v počtu vývodů. Nejčastěji se vyskytují krokové motory se čtyřmi a se šesti vodiči, uživatel se může setkat i s vodiči osmi. Čtyřvodičové motory jsou bipolární, šestivodičové jsou unipolární – dají se však zapojit i jako bipolární.



Obr. 4 Čtyřvodičové a šestivodičové zapojení krokových motorů [9]

2.1.3 Výhody krokových motorů

- Úhel pootočení je přímo úměrný počtu impulzů z řídicí elektroniky
- Možnost řízení bez zavedení zpětné vazby
- V případě zapnutého napájení dokáže motor vyvinout moment i v klidovém stavu
- Chyby natočení se pohybují okolo 3 % - 5 %, chyba se však nesčítá
- Jediná mechanicky namáhaná část jsou ložiska rotoru, motor je tedy spolehlivý a velmi odolný, bez jakýchkoliv následků přečká dokonce přetížení či násilné zastavení
- Velké rozpětí možných rychlostí – od několika stovek kroků za sekundu až po velmi malé rychlosti, nedojde přitom ke snížení momentu
- Možnosti otáčet motor oběma směry

2.1.4 Nevýhody krokových motorů

- Nutné složitější řídicí obvody
- V případě nekvalitního řízení se mohou objevit silné rezonance
- Vzhledem ke své velikosti mají oproti ostatním typům motorů nižší moment
- Oproti ostatním motorům mají menší maximální rychlost otáčení, při jejím překročení začíná docházet ke ztrátě kroků a v případě absence zpětné vazby i ke ztrátě informace o poloze

2.1.5 Shrnutí

Existují celkem tři typy krokových motorů. Nejméně používané jsou krokové motory s proměnnou reluktancí. Krokové motory s rotorem z permanentního magnetu jsou levné a mají větší moment a zároveň vetší délku kroku. Hybridní motory kombinují výhody obou předchozích typů, mají větší moment, jsou rychlé a mají malý krok, nevýhodou je vyšší cena. Spolu s motory z permanentního magnetu jsou často používané.

Existuje několik základních typů zapojení – čtyřvodičové bipolární a šestivodičové unipolární, které lze zapojit i jako bipolární. Obecně je výhodnější bipolární, má sice větší spotřebu energie a je náročnější na řídicí obvody, avšak má větší moment.

Motory se nejčastěji řídí jednofázově či dvoufázově, jednofázové řízení má větší krok a má menší moment. Dvoufázové má oproti jednofázovému dvakrát menší krok a zároveň větší moment, kombinací jednofázového a dvoufázového řízení lze docílit posouvání o půl kroku.

Jak už bylo zmíněno na začátku kapitoly 2.1 a jak zároveň vyplývá z kapitol 2.1.3 a 2.1.4, krokové motory jsou nejvhodnější pro aplikace typu „otoč se na pozici a tu drž“. Z toho vyplývá, že jsou tyto motory pro manipulátor vzhledem k jeho použití a k poskytnutému řídicímu hardwaru nejvýhodnější.

2.2 Řídicí hardware

Hlavní řídicí jednotka celého manipulátoru se bude skládat z počítače s řídicím softwarem a zařízením CompactRIO (dále již jen cRIO), které bude dostávat příkazy z PC a bude ovládat jednotlivé motory. V následujících kapitolách se zaměřím na jednotku cRIO.

Toto zařízení je modulární. Samotné tělo obsahuje programovatelné hradlové pole FPGA a procesor podporující real-time aplikace. Do tohoto těla se následně vkládají vstupně-výstupní karty (neboli I/O karty), které se pořizují zvlášť podle požadavků na aplikaci.

2.2.1 I/O karty

Jedna z největších výhod cRIO spočívá ve vyměnitelných vstupně-výstupních kartách. Tyto karty mohou podporovat logické vstupy a výstupy, sériovou komunikaci, bezdrátovou komunikaci, měření napětí, proudu, teploty, odporu, řízení motorů… Díky tomu může být zařízení extrémně univerzální.

Vezmeme-li v potaz obrovské množství typů karet, budou přiblíženy pouze ty, které jsou pro manipulátor použitelné. Tím jsou myšleny karty pro řízení motorů a karty s logickými vstupy.

Na internetových stránkách výrobce (ni.com) je uvedeno celkem šest typů I/O karet, které jsou určeny pro řízení motorů. Hlavní rozdíl těchto karet je v typech motorů, pro které jsou karty určeny (tj. krokové motory nebo servomotory), v podpoře koncových snímačů a také samozřejmě v ceně. Všechny karty nabízené na stránkách ni.com podporují pouze jeden připojený motor.

Typ karty	Typ motoru	Počet podporovaných enkodérů	Cena[1] [Kč]
NI 9502	Bezkartáčový servomotor	0	16 230
NI 9503	Krokový	0	14 170
NI 9505	Kartáčový servomotor	0	13 340
NI 9512	Krokový motor	1	15 540
NI 9514	Kartáčový i bezkartáčový servomotor	1	16 640
NI 9516	Kartáčový i bezkartáčový servomotor	2	17 740

Tab. 1 Typy I/O karet určených pro řízení motorů [1]

Základní výběr tedy závisí na přesné aplikaci, ke které chceme kartu využít. Jak již bylo uvedeno, manipulátor obsahuje pouze krokové motory, na výběr tedy zbývají pouze karty NI 9503 a 9512.

Karta NI 9503 podporuje konstantní proud 3 A RMS (ve špičce až 4,24 A) na jednu fázi, softwarově programovatelné poloviční kroky nebo mikrokrokování až do hodnoty 256 (viz kapitola 2.1.2), dvoufázové bipolární nebo unipolární zapojení a programovatelné snížení proudu z důvodu šetření energie. Model NI 9512 má navíc ještě možné připojení koncových snímačů, snímače polohy „home“ a inkrementálního enkodéru.

Obr. 5 Karty NI 9503 (vlevo) a NI 9512 (vpravo) [1]

[1] Uvedené ceny jsou platné ke dni 4.5.2017

Následný výběr závisí tedy na potřebě snímat polohu pomocí zpětné vazby. Jak bylo uvedeno v kapitole 2.1, u krokových motorů tato funkce není nutnost. I tak je ale třeba polohu nějak inicializovat, k tomu všemu je zavedení koncových snímačů požadováno v zadání. Z tohoto požadavku vyplývá, že podpora zpětné vazby je vhodná, jinak by bylo třeba využít další karty, což by celé řešení prodražilo. Tím pádem jako nejvhodnější karta pro potřeby manipulátoru vychází typ NI 9512.

V případě, že by karta diskutovaná v předchozím odstavci nepodporovala zavedení zpětné vazby (s čímž nyní budeme počítat), bylo by kvůli požadavku na podporu koncových snímačů nutné použít další, která by dokázala snímat logické úrovně vstupního signálu. Pro tyto aplikace firma National Instruments prodává celkem 13 karet, z toho 7 karet podporuje přepnutí na logické výstupy.

Typ karty	Logika [V]	Počet kanálů	Výstup	Cena[2] [Kč]
NI 9401	5	8	Ano	9 080
NI 9402	5	4	Ano	6 880
NI 9403	5	32	Ano	12 380
NI 9411	±5 až ±24	6 diff	-	7 020
NI 9421	24	8	Ne	2 970
NI 9422	24	8 diff	Ano	5 500
NI 9423	24	8	Ne	9 080
NI 9425	24	32	Ne	10 730
NI 9435	±5 až ±250 VDC 10 až 250 VAC	4 diff	Ano	5 500
NI 9381	5	4	Ano	11 830
NI 9375	12, 24	16	Ne	12 650
NI 9436	±100 až ±250 VDC ±100 až ±250 VAC	8 diff	Ano	8 250
NI 9437	24 až 250 VDC	4 diff	Ne	7840

Tab. 2 Typy I/O karet určených pro detekci logických úrovní signálu [1]

Z nabízených karet tedy vyplývají základní parametry, podle kterých kartu vybírat. Jako první parametr je třeba rozhodnout, zda je potřeba u karty nastavovat výstup signálu. Manipulátor žádné logické úrovně nepřijímá, teoreticky tedy výstupy potřeba nejsou. Obvody pro koncové snímače je potřeba napájet, nabízí se tedy otázka, zda by logický výstup obvody napájet dokázal neboli zda je zdroj dostatečně „tvrdý“ a nebylo by vhodnější obvody napájet zdrojem externím. Zvážíme-li všechny tyto parametry, vychází nám, že toto kritérium nerozhoduje. Další parametr, který opomeneme, je konektor – obvod pro koncové snímače navrhujeme sami, konektor tedy může být libovolný. Taktéž v úvahu vůbec

[2] Uvedené ceny jsou platné ke dni 4.5.2017

nebereme případnou rychlost přepínání mezi konfigurací vstup-výstup, tato konfigurace je nastavena na začátku běhu programu a po celou dobu se nemění.

Dalším parametrem je počet potřebných kanálů. V manipulátoru je celkově 5 koncových snímačů, z toho jeden v rotačním členu, translační členy mají každý dva. Potřebný počet kanálů tedy závisí na schopnosti zkombinovat dva koncové snímače v translačním členu do jednoho logického výstupu. Jestliže by toto bylo možné, stačila by karta se čtyřmi vstupy, v opačném případě je nutné použít kartu se vstupy osmi. Karty s více vstupy jsou zbytečné, z výběru je tedy odstraníme. Pro jednodušší realizaci taktéž odstraníme karty se vstupy diferenčními.

Poslední otázka se týká použité logické úrovně signálu – zda je nutné použít 5 V či 24 V. Tato otázka závisí na použitém zdroji napětí, berme tedy zatím v potaz obě možnosti.

Z toho nám vychází několik možností. Jestliže bude možné spínače zapojit tak, aby využívaly pouze 3 logické vstupy, pro logiku 5 V bude nejvýhodnější karta NI 9402, pro logiku 24 V model NI 9421. V opačném případě pro logiku 5 V vychází nejvýhodněji model 9401, pro úroveň 24 V opět nejvýhodněji vychází karta NI 9421.

Obr. 6 Karty NI 9401 (vlevo), NI 9402 (uprostřed) a NI 9421 (vpravo) [1]

2.2.2 CompactRIO

Jak již bylo zmíněno, pod pojmem CompactRIO (nebo cRIO) je myšleno šasi řídicího zařízení, které obsahuje real-time procesor, programovatelné hradlové pole FPGA a sloty pro I/O karty. Nabízená zařízení se liší mimo jiné typem konektoru pro komunikaci, programovatelným hradlovým polem a zejména počtem slotů pro I/O karty.

Typ[3]	Procesor [MHz]	Pevný disk [GB]	RAM [MB]	Sloty	Cena[4] [Kč]
9063	667	0,512	256	4	27 400
9066	667	0,512	256	8	48 200

[3] Kvůli úspoře místa není v názvu modelu uvedeno slovo cRIO

[4] Uvedené ceny jsou platné ke dni 4.5.2017

Typ[3]	Procesor [MHz]	Pevný disk [GB]	RAM [MB]	Sloty	Cena[4] [Kč]
9064	667	1	512	4	48 200
9065	667	1	512	4	71 300
9030	1 330	4	1 000	4	71 300
9067	667	1	512	8	71 300
9035	1 330	4	1 000	8	83 900
9031	1 330	4	1 000	4	96 300
9065	667	1	512	4	99 600
9035	1 330	4	1 000	8	99 600
9068	667	1	512	8	112 800
9032	1 330	8	2 000	4	120 800
9033	1 330	8	2 000	4	132 900
9037	1 330	8	2 000	8	132 900
9038	1 330	8	2 000	8	145 000
9034	1 910	16	2 000	4	169 200
9039	1 910	16	2 000	8	181 300

Tab. 3 Typy zařízení cRIO [1]

Jak je z Tab. 3 zřetelné, cRIO se prodává v mnoha variantách a s velkými cenovými rozdíly. Velké množství rozdílných znaků není kvůli úspoře místa v tabulce uvedeno – patří mezi ně typ a počet komunikačních konektorů (ethernet se vyskytuje vždy), přítomnost Wi-Fi, typ FPGA, odolnost proti okolním vlivům jako je vlhkost atd. Vzhledem k nenáročné aplikaci a nepředpokládaným zhoršeným podmínkám provozu, absence těchto prvků nehraje žádnou roli.

Aby bylo možné vybrat vhodné zařízení, musíme nejprve přesně definovat požadavky, které na něj máme. Z kapitoly 2.2.1 vyplývá, že budou do zařízení zasunuty maximálně čtyři I/O karty – karty pro řízení motorů podporují pouze jeden motor, tím pádem připadají 3 karty na řízení a 1 na koncové snímače. Dále je nutné si rozvrhnout náročnost programu, který cRIO musí zvládnout. Manipulátor bude ovládán hlavním řídicím programem spuštěným na PC. Na zařízení samotném bude na FPGA nahrán pouze program pro komunikaci s procesorem. Zároveň na zařízení nebudou probíhat žádné výpočty či složitější analýzy signálů, ani se zde nebudou ukládat žádná naměřená data.

Z předcházejícího odstavce tedy vyplývají následující požadavky: 4 sloty pro karty, stačí nejméně výkonný procesor, nejmenší velikost paměti pevného disku a nejmenší velikost paměti RAM. Za těchto podmínek spolu s tím, že se vezme v potaz i otázka ceny, jako nejlepší a plně vyhovující požadavkům vychází zařízení cRIO 9063.

Obr. 7 Zařízení cRIO 9063 [1]

2.3 Vývojový software

Jak jsem se již zmínil v úvodu, řídicí program bude vytvářen v prostředí LabVIEW, použita je verze z roku 2014. V této kapitole jsou uvedeny základy tohoto prostředí, vzhledem k obrovské rozsáhlosti a komplexnosti softwaru je toto téma probráno pouze velmi zběžně, spíše než popis prostředí vysvětluje tato kapitola pojmy dále používané v této práci. Ještě předtím je ale nutné položit jednu základní otázku: co to LabVIEW vlastně je? *„LabVIEW je integrované vývojářské prostředí, speciálně navržené pro inženýry a vědce, určené pro vývoj měřicích a kontrolních systémů.“* [2]

2.3.1 Historie

Vývoj programovacího prostředí LabVIEW (zkratka z Laboratory Virtual Instrumentation Engineering Workbench) se hluboce pojí s vývojem firmy National Instruments. Firma vznikla v roce 1976, kdy se tři podnikatelé z Austinu (James Truchard, Jeffrey Kodosky a Bill Nowlin) rozhodli založit společnost orientovanou na vývoj zařízení pro vědeckou instrumentaci. První produkt začali vyvíjet v garáži Jamese Trucharda, později prezidenta této firmy. Výrobek byl uveden na trh o rok později, tj. v roce 1977. Firma se poměrně rychle rozrůstala. V roce 1983 oznamuje uvedení prvního zařízení určeného pro osobní počítače, v tomto roce také začal

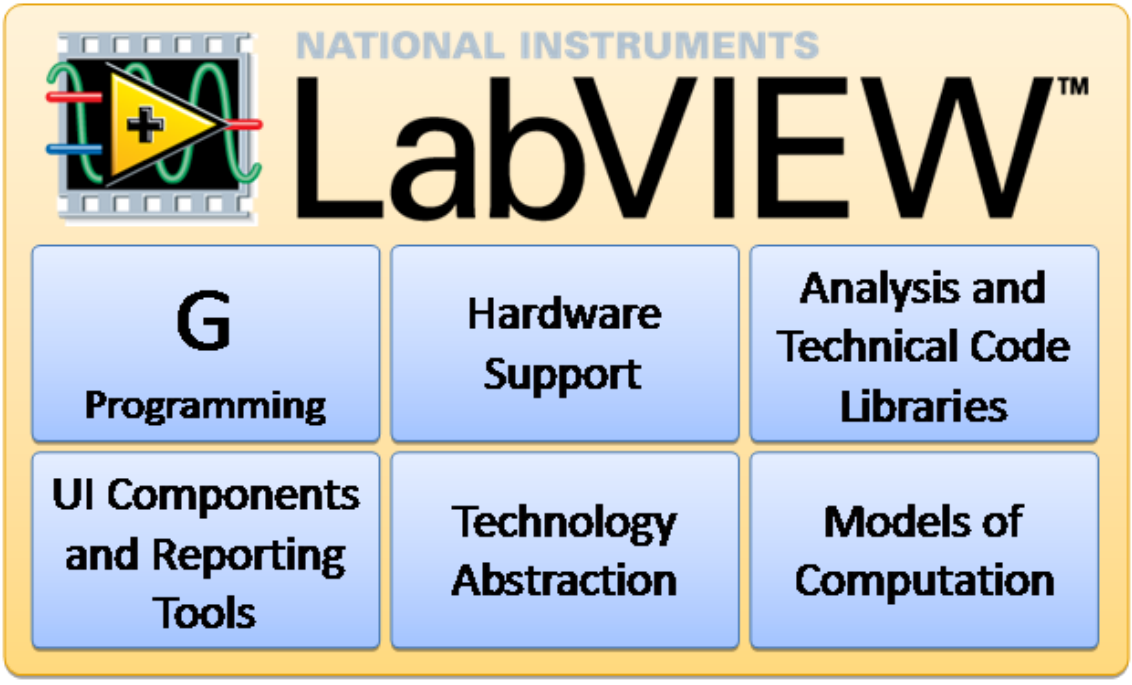
Jeffrey Kodosky pracovat na první verzi LabVIEW. Později začíná být označován jako „otec“ tohoto prostředí.

První verze programu je vydána v roce 1986, avšak zatím pouze pro Apple Macintosh, verze pro MS DOS vyšla o rok později pod názvem LabWindows. Firma se začala vyvíjet téměř raketovou rychlostí, spolu s tím samozřejmě i LabVIEW – kvůli stále rostoucímu počtu zákazníků a požadavkům na použití musely být do programu přidávány další a další moduly, které vylepšovaly výkon, a čím dál více rozšiřovaly možnosti použití.

Postupem času se program vyvinul do velmi rozsáhlého a komplexního systému s podporou hardwaru vyráběného nejen firmou NI. V současné době je software ve velké míře využíván například firmami Airbus a Subaru, dále ho používá společnost National Grid UK pro návrhy a realizaci měření v elektrické rozvodné soustavě. Software byl dokonce aplikován při analýze a zobrazování dat v NASA Jet Propulsion Laboratory.

2.3.2 Základy

Prostředí LabVIEW se skládá z několika hlavních modulů, které jsou potřeba k jakémukoliv měření nebo ovládání zařízení. Tyto moduly jsou zobrazeny na Obr. 8.



Obr. 8 Hlavní moduly LabVIEW [3]

Jedno z hlavních hesel programátorů NI je: *„Píšeme nízkoúrovňový kód, abyste to nemuseli dělat vy.“* [3]. Tím se toto prostředí liší od jiných programovacích jazyků. V praxi to znamená, že uživatel vytvoří v grafickém prostředí program, který je následně zkompilován tak, aby byl co nejefektivněji využit výkon počítače.

Například jsou výpočty rozděleny mezi jádra procesoru, což v jiných jazycích musí dělat sám programátor.

Nejvíce přijde uživatel do styku s modulem G Programming – grafické programování. Dá se říci, že ostatní moduly G Programming spíše rozšiřují, než že existují samostatně. Jazyk G Programming je pro prostředí tak typický, že se občas označuje LabVIEW Programming. Při vytváření jazyka měl „otec prostředí" Jeffrey Kodosky takovou vizi, aby *„technik, který je schopen zapsat své poznatky a požadavky do blokového diagramu, mohl intuitivně zapsat podobně i program"* [4].

2.3.3 Popis prostředí

Po vytvoření nového projektu se uživateli otevře tzv. Project Explorer. V něm se zobrazují všechny soubory projektu a případně připojená zařízení. Projekt obsahuje soubory s vytvořenými programy, může však obsahovat i soubory s typovými definicemi nebo s globálními sdílenými proměnnými.

Samotné programy jsou vytvořeny v souborech VI (virtuální instrumenty). Ty mohou obsahovat jak hlavní program, tak je lze použít jako podpůrné programy v nadřazených souborech označovaných jako subVI.

Při otevření nového VI se zobrazí prázdná obrazovka, tzv. Front panel. Ten zobrazuje vzhled uživatelského prostředí aplikace. Kliknutím pravého tlačítka myši na volnou plochu se zobrazí nabídka dostupných ovládacích a zobrazovacích prvků – přes tlačítka, přepínače a posuvníky až k různým pokročilým prvkům, jako jsou například grafy. Na Front panelu následně programátor může prvky posouvat, měnit jejich velikost, vzhled nebo chování.

Každý Front panel má k sobě přidružený tzv. Block diagram. Mezi Front panelem a Block diagramem může uživatel libovolně přepínat pomocí klávesové zkratky <ctrl+e>. Na obrazovce blokového diagramu definuje programátor vlastní algoritmus programu. Při kliknutí pravým tlačítkem na volnou plochu zobrazí Block diagram dostupné bloky - např. aritmetické, pro práci s řetězci, se soubory, různé struktury, jako jsou for nebo while cykly, event struktury atd.

Velký rozdíl v chování oproti jiným programovacím jazykům spočívá v pořadí vykonávání algoritmu, v LabVIEW označovaný jako dataflow (doslova přeloženo jako tok dat). Jazyky jako Java nebo C++ vykonávají příkazy v předem daném pořadí shora dolů. LabVIEW oproti tomu nemá předem danou posloupnost, jediné pravidlo, které dodržuje je to, že se postupně vykonávají příkazy, které jsou zapojené za sebe. Tím pádem může při využití proměnných docházet k chybám způsobeným zapsáním hodnot v jiném pořadí, než předpokládá programátor. K tomu nemusí ještě ke všemu docházet při každém spuštění programu, ale pouze někdy – časy zápisu a čtení se totiž mohou lišit v závislosti na vytížení zařízení. Tím pádem se použití jak lokálních, tak globálních sdílených proměnných v programu

příliš nedoporučuje, což může být pro programátory zvyklé na jiné programovací jazyky ze začátku nezvyklé. Jestliže je použití proměnných opravdu nutné, musí být programátor velmi obezřetný – zejména v částech kódu, ve kterých se do proměnných zapisuje.

Zdrojový kód programu se skládá z funkčních bloků, které jsou propojeny pomocí spojů (tzv. wire). Jak jsem již psal výše, funkční bloky mohou mít různé funkce – provádějí matematické operace, zapisují do souborů, jako funkční blok lze dokonce využít i jiné VI s vlastním algoritmem. Jako funkční bloky se v Block diagramu zobrazují i ovládací a zobrazovací prvky vytvořené na Front panelu. V opačném pořadí program funguje stejně, jestliže v Block diagramu vytvoříme např. tlačítko nebo graf, zobrazí se i na Front panelu. Jak vyplývá z předchozího textu, funkční bloky mají mnoho rozličných funkcí a tím pádem mohou mít mnoho různých vstupních a výstupních datových typů (např. float, string, boolean, cluster nebo referenci). Z toho je zřejmé, že i spoje připojené na vstupy a výstupy mohou mít různé datové typy. Aby byl program přehlednější, jednotlivé datové typy jsou u spojů rozlišeny barvou a typem čáry (např. boolean je zelený, string fialový atd.).

2.3.4 Moduly

Do programu LabVIEW lze přidávat dodatečné moduly, které pomocí dalších funkčních bloků a podpory hardwaru velmi rozšiřují jeho funkcionalitu. V následujícím textu se zaměřím na moduly FPGA a SoftMotion, který jsou nezbytné při návrhu softwaru pro manipulátor.

2.3.4.1 FPGA

Jak jsem již psal výše, jedna ze tří hlavních částí zařízení cRIO je programovatelné hradlové pole FPGA, které slouží ke komunikaci mezi procesorem a vstupně-výstupními moduly a může sloužit např. k rychlým aritmetickým výpočtům nebo zpracování signálů. FPGA lze brát jako velké množství tranzistorů, mezi kterými uživatel naprogramuje pomocí jazyka HDL spoje tak, aby hradlové pole provádělo operace přesně podle jeho přání.

Jazyk HDL slouží jako strukturovaný popis chování součástek a jejich spojení. Tento způsob zápisu je mnohem efektivnější než pouhé vytvoření grafického schématu. Přesto občas není dostačující. Základní myšlenka prostředí LabVIEW spočívá v tom, že technik bez hlubších znalostí programovacích jazyků dokáže vytvořit aplikaci, stačí mu pouze dokázat vytvořit vývojový diagram. Zároveň je prostředí vytvořené tak, aby programátor navrhl algoritmus a samotný program už se postará o to, aby byl co nejefektivněji využit výpočetní výkon přístroje.

Nyní si představme, že technik má už na hradlové pole FPGA složitější požadavek a nestačí mu např. pouze sčítačka nebo podobně jednoduchý obvod.

Když by muselo být hradlové pole nakonfigurováno pomocí HDL, byly by porušeny všechny principy zmíněné v předchozím odstavci. Jednak by technik musel strávit velké množství času návrhem obvodu a jednak by už nezbylo příliš prostoru pro program, aby optimalizoval kód. A v neposlední řadě, ne každý technik pracující s LabVIEW zná jazyk HDL.

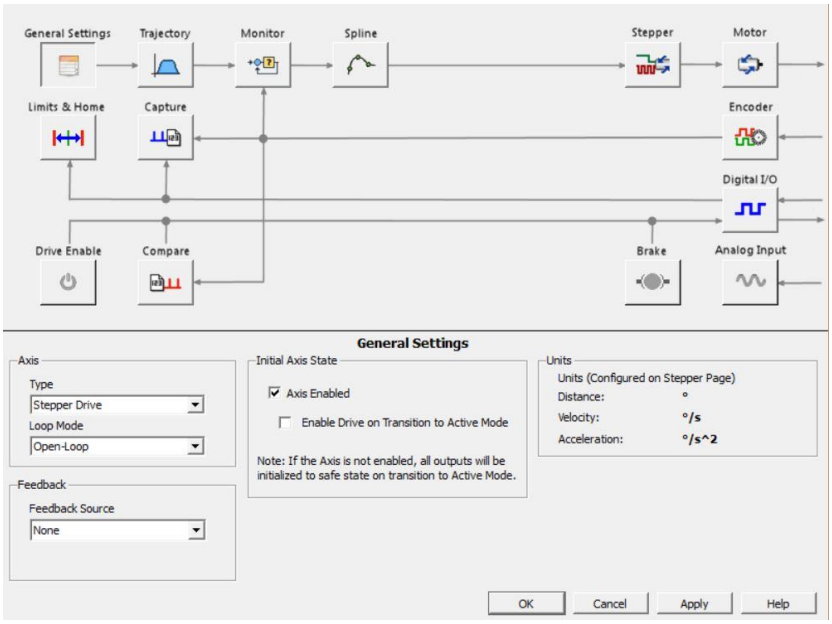
Kvůli tomu vznikl modul LabVIEW FPGA. Technik pomocí Project Exploreru vytvoří na hradlovém poli FPGA nové VI. V tom následně navrhne požadovaný algoritmus stejným způsobem, jako v klasickém VI. Jediný rozdíl spočívá v tom, že jsou k dispozici navíc některé funkce vztahující se ke konkrétnímu zařízení cRIO. Po dokončení návrhu programátor spustí VI, následně se kód pomocí nástroje od firmy Xilinx (dodavatel FPGA hradlových polí do zařízení) zkompiluje a hradlové pole se nakonfiguruje. Kód je možné spustit buď přímo z VI, další způsob je pomocí tzv. bitových souborů (v angličtině označovaných jako bitfile). Kompilátor totiž na předem určeném místě vytvoří bitový soubor, který obsahuje zkompilovaný kód. Jediný další rozdíl oproti klasickému VI je, že po zkompilování a spuštění VI s FPGA kódem není možné kód jakýmkoliv způsobem debugovat. To je ale poměrně logický důvod vyplývající z charakteristiky FPGA kódu – kód není program, který by byl na hradlu spuštěn, kód je architektura, na kterou se hradlové pole nakonfiguruje (velmi zjednodušeně řečeno „způsob propojení tranzistorů“).

V předchozím textu bylo zmíněno, že FPGA slouží v zařízení ke komunikaci mezi RT procesorem a I/O kartami, přičemž karty mohou být připojeny dvěma způsoby. Tzv. Scan Interface znamená, že si po připojení karty zařízení svoje FPGA samo nakonfiguruje. Tato funkce však není dostupná pro všechny karty a je doporučena pouze pro základní aplikace, kdy nemá uživatel požadavky na vysokou rychlost a velký objem přenášených dat a nevyžaduje využití pokročilejších funkcí přímo na hradlovém poli (např. analýza signálů, složitější výpočty atd.). Mód FPGA Interface naopak uživateli umožní FPGA nakonfigurovat ručně a tím využívat i velmi pokročilé funkce hradlového pole. Jestliže má cRIO více karet, mohou být některé připojeny přes Scan Interface a některé přes FPGA interface. V tomto případě se uvádí, že hradlové pole pracuje v hybridním módu.

2.3.4.2 SoftMotion

Modul LabVIEW SoftMotion je určený k pokročilému řízení pohybu. Tento modul je speciálně navržen pro co největší efektivitu při použití se zařízením cRIO, přímo totiž podporuje některé typy vstupně-výstupních karet. Mimo ovládání pro motory modul obsahuje funkce pro generování pohybových trajektorií, funkce enkodéru či funkce PID regulátoru.

Základní myšlenka modulu SoftMotion spočívá v ovládání os. Programátor v Project Exploreru nadefinuje osy pohybu, jako osu lze použít např. jeden motor. Nastavení os je poměrně komplexní záležitostí, kdy lze zadat jednak základní technologické údaje motoru (např. počet kroků na jednu jednotku nebo napájecí proud na fázi), jednak modul počítá i s pokročilými funkcemi, jako zpětná vazba nebo maximální limity osy, příklad obrazovky konfigurace je na Obr. 9.



Obr. 9 Příklad konfigurace osy krokového motoru

Po správné konfiguraci už programátor pouze ovládá osy pomocí funkčních bloků uvnitř VI. LabVIEW v tomto modulu poskytuje velké množství rozličných bloků, např. pro povolování a zakazování pohybu os, pro zjišťování aktuálního stavu motoru, pro lineární, kruhový nebo zakřivený pohyb, v případě nakonfigurovaného snímače inicializační polohy modul zajišťující návrat do této polohy, existují i funkce, které pracují s více osami zároveň (např. pro pohyb po kartézské rovině s osami X a Y).

2.4 Shrnutí teoretické části

Teoretická část se rozděluje na tři hlavní oblasti. První oblast stručně vysvětluje principy fungování krokových motorů a způsob jejich zapojení a řízení. Druhá oblast provádí analýzu hardwarových možností, které firma NI poskytuje na svých internetových stránkách.

S ohledem na požadavky, které jsou na hardware kladené, jako nejvhodnější bylo určeno zařízení cRIO 9063 se třemi I/O kartami NI 9512. V případě, že by na řízení motorů byl poskytnut jiný typ karet, k detekci stavu koncových snímačů by byla nutná ještě čtvrtá karta. Její přesný typ závisí zejména na použité logické úrovni

signálu a na počtu vstupů, který závisí na navrhnutém obvodu. Při zvážení těchto kritérií byly k výběru zvoleny tři karty. Jestliže by byla použita logická úroveň 24 V, nejlépe vychází karta Ni 9421. V případě úrovně 5 V je v případě 4 vstupů nejvhodnější model 9402, jestliže by čtyři vstupy nebyly dostačující, jako nejlepší vychází karta NI 9401.

V poslední podkapitole je přiblíženo prostředí LabVIEW. Vzhledem k obrovskému rozsahu tohoto tématu se však kapitola spíše zabývá vysvětlením pojmů, které se objevují v praktické části.

3 PRAKTICKÁ ČÁST

Praktická část se nejdříve zaměřuje na porovnání poskytnutého hardwaru s tím, který je v teoretické části vyhodnocen jako nejvhodnější a následně se zabývá diskuzí možných rozdílů. Dále projednává propojení I/O karet s krokovými motory a s koncovými snímači.

Další dílčí část praktické části se zabývala prvotní konfigurací hardwaru, tzn. připojením cRIO k PC a jeho nastavením a následným vytvořením projektu v LabVIEW a nakonfigurováním pohybových os.

Předposlední oblast se týká vytvoření konfigurace FPGA, poslední se zabývá návrhem řídicí/vizualizační aplikace. Zde je popisována architektura programu a hlavní algoritmy, které program využívá.

3.1 Hardware

3.1.1 Použité přístroje

V kapitole 2.2 byly probrány nejvhodnější vstupně-výstupní karty a zařízení cRIO s ohledem na použití v manipulátoru. V tomto případě je řízením manipulátoru myšleno řízení tří krokových motorů a detekce stavu pěti koncových snímačů. Výsledkem tohoto rozboru byl nejjednodušší prodávaný model cRIO-9063 se třemi kartami NI 9512 pro ovládání krokových motorů. Hardware diskutovaný v teoretické části se však od poskytnutého liší.

První rozdíl spočívá v poskytnutém zařízení cRIO, manipulátor řídí model cRIO 9068. Zařízení obsahuje dvoujádrový procesor s taktem 667 MHz, 512 MB RAM a 1 GB paměti na pevném disku. Pro komunikaci jsou přítomny dva ethernetové porty, jeden slot USB a tři sériové porty. Zařízení má celkem 8 slotů pro I/O karty, tedy o 5 více, než je potřeba. Model má taktéž zvýšenou teplotní odolnost.

Obr. 10 CompactRIO 9068 [1]

Hlavní rozdíl spočívá v poskytnutých krokových motorech. V kapitole 2.2 byl proveden rozbor pro klasické řízení krokových motorů, kdy musí uživatel řídit buzení cívek. V translačních členech manipulátoru (typ Standa 8MT195) byly vyměněny krokové motory za inteligentní krokové motory NI ISM-7402 vyráběné

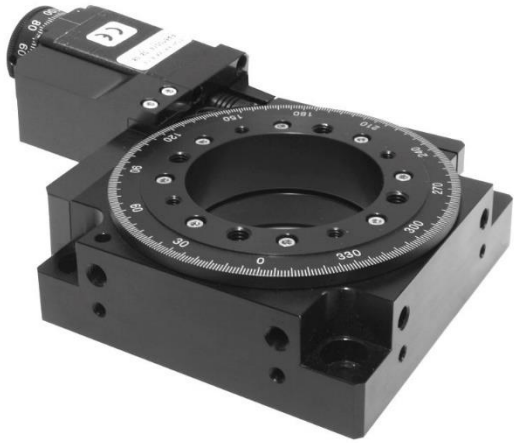
firmou National Instruments. Tyto motory obsahují jak vlastní akční člen, tak řídicí obvody. Na binární vstupy jsou jim předávány signály určující parametry pohybu, vnitřní obvody pak podle nich budí cívky akčního členu. Motory podporují několik typů zapojení, tj. jak pro řízení se signály STEP+, STEP-, DIR+, DIR-, EN+ a EN-, tak pouze pro signály STEP, DIR a EN (případ manipulátoru). Tyto vstupy podporují logickou úroveň jak 5 V, tak 24 V. Dále obsahuje logický výstup OUT signalizující chybu. Poslední konektory mají označení V+ a V-, slouží k napájení celého zařízení. Napájecí napětí je 12 V–60 V. Jako volitelné příslušenství se k motoru dá zakoupit i enkodér. Pro signalizaci stavu motoru je vedle konektorů zabudována LED dioda, která může svítit buď zeleně, nebo červeně. Jestliže dioda svítí nepřerušovanou zelenou barvou, nevyskytla se žádná chyba a pohyb motoru je zakázán. Jestliže dioda zeleně bliká, je pohyb povolen. V případě jakékoliv kombinace zelené a červené barvy nastala chyba – např. tři bliknutí červeně a jedno zeleně znamená přehřátí.

Jak již bylo zmíněno, původní translační členy Standa 8MT195 mají místo původních akčních členů přidělány inteligentní krokové motory od firmy National Instruments. Translační členy 8MT195 jsou prodávány v mnoha variantách. První rozdíl spočívá ve směru pohybu. Členy totiž mohou být konstrukčně uzpůsobeny pro horizontální nebo vertikální pohyb, při zkombinování tří členů se dá dosáhnout pohybu v kartézské soustavě po souřadnicích X, Y i Z. Další rozdíl spočívá v maximálním rozsahu, ten může být až 2840 mm. Poslední parametr, ve kterém se mohou translační členy lišit, je průměr vodícího šroubu, který se uvnitř jednotky nachází. Přítomny jsou dva koncové snímače polohy. V případě jejich aktivace se obvod rozpojí. Maximální horizontální zatížení je 40 kg.

Obr. 11 Translační motor Standa 8MT195 (vlevo) a inteligentní krokový motor NI ISM-7402 (vpravo), [13; 1]

Typ rotačního členu je Standa 8ME190-2-28. Na rozdíl od motorů použitých v translačních členech pohon rotačního členu neobsahuje žádnou řídicí elektroniku, cívky tedy musí budit sám uživatel. K řídicí elektronice se motor připojuje pomocí čtyřvodičového bipolárního zapojení. Jeho rozsah je 360 °, rozlišení na jeden krok je 0,01 °. Maximální rychlost udávaná výrobcem je až 8 otáček za minutu. Maximální horizontální zátěž je 10 kg, vertikální 2 kg. Motor obsahuje koncový snímač polohy „home“. Jestliže se motor na této poloze nachází, je koncový snímač sepnutý.

Poskytnutý model nemá žádné úpravy, na přání zákazníka však může výrobce motor dodat v několika variantách – může zabudovat jiný typ krokového motoru, dodat úpravu pro použití ve vakuu atd.



Obr. 12 Rotační krokový motor Standa 8ME190-2 [12]

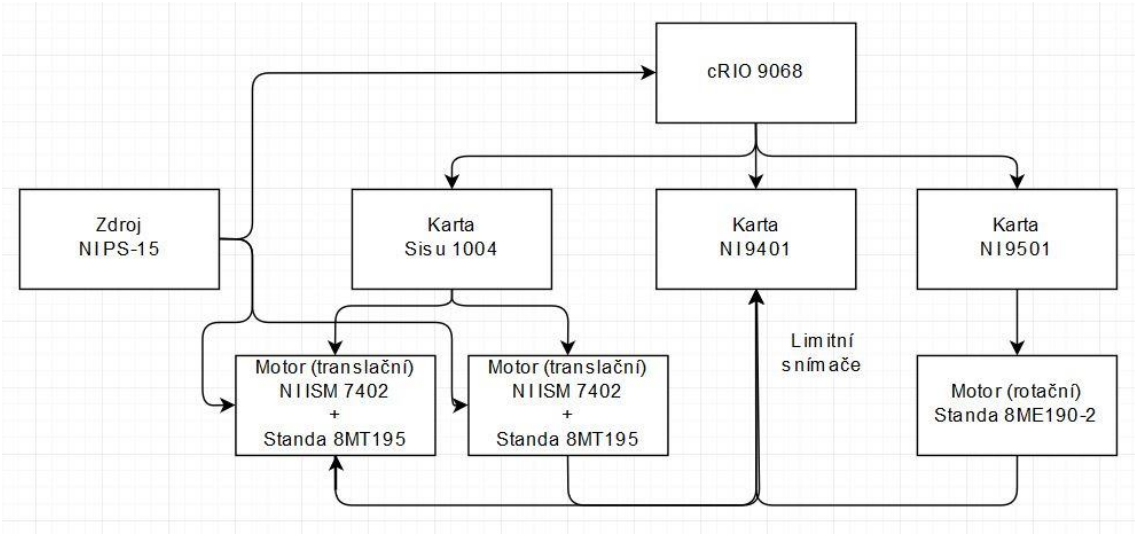
Pro řízení motorů ISM 7402 není třeba žádná složitá řídicí elektronika, dostačující je jakákoliv karta s dostatečným počtem binárních výstupů. Pro řízení motorů byla poskytnuta karta od externího výrobce, její typ je SISU 1004. Ta podporuje ovládání až čtyř krokových motorů najednou spolu s připojením koncových snímačů. Díky této kartě byl počet potřebných I/O karet snížen na 2 až 3, což je finančně velmi výhodné. Komunikace s motory probíhá přes konektor RJ-45, tedy přes ethernetové kabely. Karta obsahuje čtyři ethernetové sloty, pro každý motor jednu. Každý slot obsahuje tři piny pro řízení motorů Step, Dir a Enable. Dále karta obsahuje dva vstupy pro zapojení koncových snímačů. Logická hodnota 0 je pro tyto vstupy reprezentována napětím menším, než je 5 V, napětí mezi 11 V až 30 V značí logickou hodnotu 1. Tyto vstupy mají vstupní impedanci 26,7 kΩ, lze je tedy s koncovými snímači spojit přímo a není třeba používat rezistory. Dále karta obsahuje binární vstup pro chybu (tzn. při logické hodnotě 1 karta přestává dávat pokyn k pohybu motoru) a výstup se stabilním napětím 5 V.

Pro řízení rotačního krokového motoru byla poskytnuta karta NI 9501. Tato karta se v teoretickém rozboru vůbec nevyskytla, výrobce ji totiž přestal uvádět ve svém internetovém obchodě. Její současnou obdobou je karta NI 9503, od poskytnutého modelu se její technické specifikace vůbec neliší. Karta zvládá proud ve fázi 3 A (ve špičce až 4,24 A). Modul je určen pro připojení krokových motorů jak pomocí dvoufázového bipolárního, tak unipolárního zapojení. Karta podporuje mikrokrokování až do hodnoty 256. Samozřejmostí je programovatelná funkce režimu snížené spotřeby. Tato karta nepodporuje připojení koncových snímačů, musí tedy být použita další karta určená k detekci logických úrovní signálu.

V teoretickém rozboru vycházelo pro logické signály velmi výhodně několik I/O karet, hlavní kritérium představovala úroveň logického signálu a počet nutných logických vstupů. Poskytnutý model karty je nakonec NI 9401, který byl jako vhodný v rozboru uveden. Tento model obsahuje 8 vstupů a pracuje s logickou úrovní signálu 5 V. Karta podporuje přepínání konfigurace vstup – výstup, a to s velmi malou periodou 100 ns. Karta však podporuje přepínání vstupů pouze po čtveřicích, jsou tedy možné pouze konfigurace 8 vstupů, 8 výstupů nebo 4 vstupy a 4 výstupy.

Obr. 13 Použité I/O karty – NI 9401 (vlevo), NI 9501 (uprostřed), SISU 1004 (vpravo) [1]

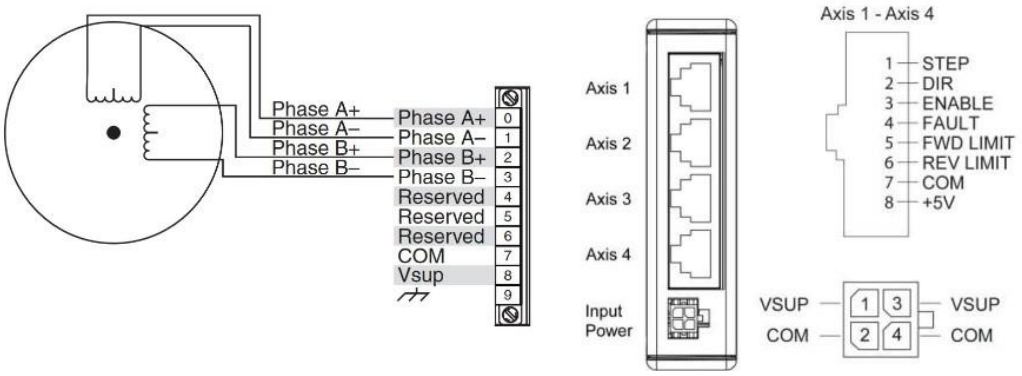
Celý manipulátor se tedy nakonec skládá ze zařízení cRIO-9068, které je napájeno ze zdroje NI PS-15. Toto zařízení je připojeno pomocí ethernetového kabelu k PC. V cRIO jsou zasunuty celkem 3 I/O karty. Karta SISU 1004 slouží k řízení dvou translačních motorů, NI 9501 je určena k řízení rotačního členu a nakonec NI 9401 detekuje stav koncových snímačů. Jako translační motory jsou použity modely Standa 8MT195, u kterých byla pohonná jednotka nahrazena motorem NI ISM 7402, jako rotační motor slouží motor Standa 8ME190-2-28.



Obr. 14 Zapojení celého zařízení

3.1.2 Zapojení

Karty NI 9501 i SISU 1004 jsou přímo určeny pro řízení motorů, čemuž odpovídají i jejich výstupy – motory se připojují přímo na příslušné piny v konektorech, není potřeba vytvářet žádné řídicí obvody. Karta SISU 1004 podporuje připojení koncových snímačů polohy. Tyto vstupy ale nebyly použity ze dvou důvodů. První spočívá v kartě použité pro ovládání rotačního členu – karta nepodporuje připojení koncových snímačů, je tedy nutné pro tento účel použít další kartu. Tím pádem je pro celkovou implementaci jednodušší použít stejný způsob detekce pro všechny snímače v manipulátoru. Druhý důvod je složitost kódu pro hradlové pole FPGA. Aby modul SoftMotion se snímači správně komunikoval, bylo potřeba upravit konfiguraci FPGA. Tento úkol byl však tak složitý, že se mi ho nepodařilo splnit. Při použití další karty nebylo třeba kód upravovat, karta totiž byla připojena v režimu Scan Interface.

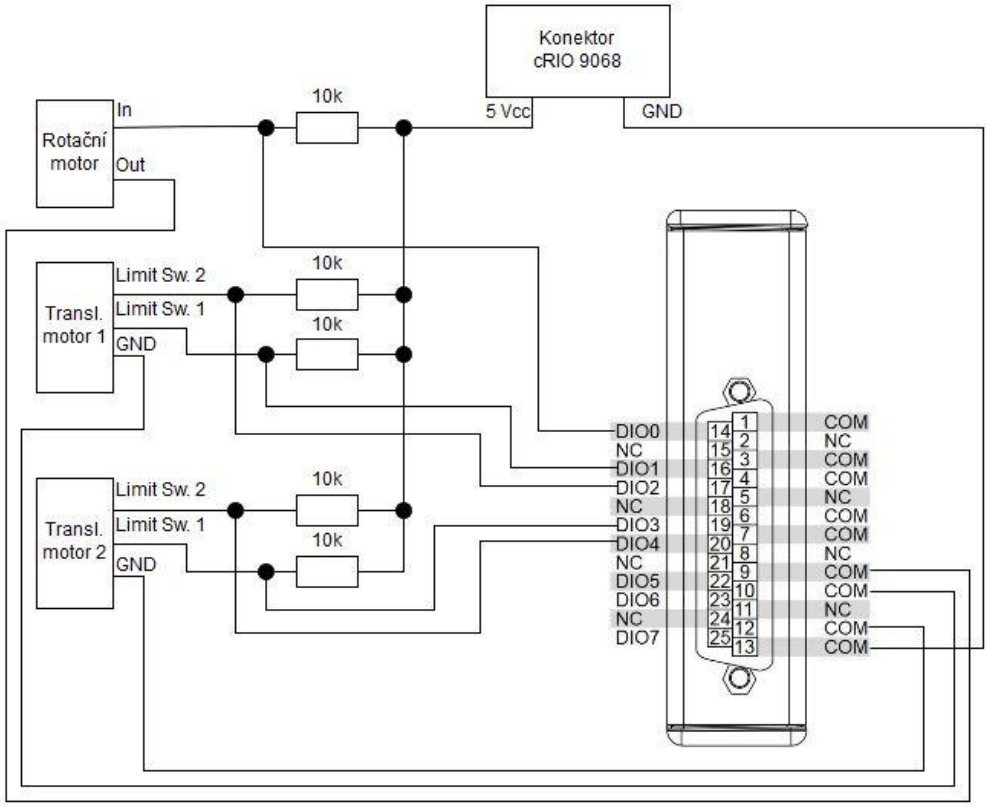


Obr. 15 Propojení I/O karet s motory – karta NI 9501 vlevo, SISU 1004 vpravo [11; 10]

Karta NI 9401 pro detekování logických úrovní signálu je velmi univerzálního zaměření, bylo tedy třeba vytvořit obvod. Nyní je na místě přiblížit koncové snímače v motorech. Rotační motor (Standa 8ME190-2) obsahuje jeden snímač polohy „home“. Tento snímač je v poloze „home“ sepnutý, přesněji řečeno je obvod sepnutý v rozmezí natočení přibližně 3° až 5° (poloha stupnice vzhledem k rysce nacházející se na motoru). Bohužel při běhu motoru občas dochází k šumu, pravděpodobně dochází k vibracím, které koncový snímač sepnou a obvod detekuje polohu i v případě, že se v ní rotační člen nenachází. Dráty pro zapojení snímače jsou součástí připojovacího kabelu. Translační motory obsahují každý dva koncové snímače, jeden na každé straně motoru, detekují koncové polohy. Obvody jsou v případě sepnutí snímačů rozpojené. Než byly v translačních motorech vyměněny akční členy, celý motor byl ovládán přes konektor DB9(M) nacházející se na boku těla. Při výměně byla část vodičů vedoucích k akčnímu členu odletována, NI ISM 7402 se ke kartě připojuje přímo. Na konektoru tedy aktivní zůstaly pouze piny

určené pro připojení koncových snímačů. Motory jsou k ovládací kartě připojeny přímo, koncové snímače jsou vyvedeny do konektoru nacházejícího se na boku těla.

Díky tomu, že jsou obvody koncových snímačů v sepnuté poloze rozpojené, bez složitějších obvodů (např. hradlo nand) nelze spínače spojit tak, aby spínače pro jeden translační člen zabíraly pouze jeden logický vstup. Na kartu tedy musí být přivedeno celkem pět vstupů, z důvodů uvedených v kapitole 3.1.1 tak musí být karta NI 9401 nakonfigurována na 8 vstupů a zdroj pro obvod musí být zvolen externí. Napájení celého obvodu (5 V) je vyveden z volného slotu zařízení cRIO, stejně tak připojení GND.



Obr. 16 Schéma zapojení koncových snímačů

Jak vyplývá z předchozího odstavce a z Obr. 16, poloha „home“ na rotačním motoru je detekována logickou nulou na vstupu DIO0. Přední koncový snímač (tj. snímač blíže akčnímu členu) je pro translační motor 1 detekován logickou jedničkou na vstupu DIO1, zadní na DIO2. Přední snímač motoru 2 je detekován logickou jedničkou na DIO3, zadní jedničkou na DIO4. Jednotlivé úrovně se na vstupech detekují pomocí globálních proměnných, které se v projektu po přidání karty objevily. Jinak řečeno se do programu jednoduše přidá globální sdílená proměnná typu boolean, jejíž hodnota je řízena zařízením.

3.1.3 Shrnutí kapitoly

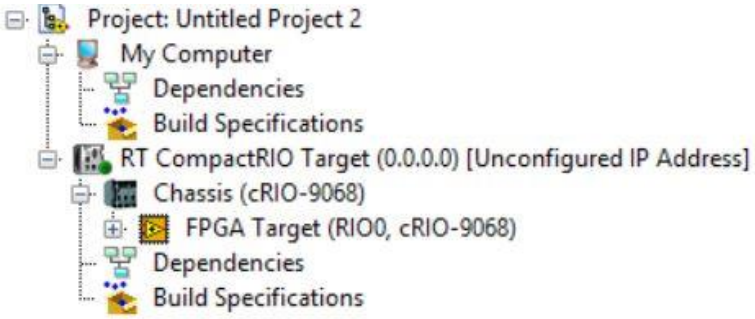
Kapitola se zabývá srovnáním poskytnutého hardwaru s tím, který v teoretickém rozboru vyšel jako nejvhodnější. Samotné poskytnuté zařízení cRIO 9068 svými možnostmi a funkcemi vysoce převyšuje požadavky, které jsou manipulátorem kladeny. V případě nákupu zařízení by rozhodně bylo vhodnější díky několikanásobně nižší ceně použít zařízení cRIO 9063 doporučené v teoretické části. Rovněž karta určená pro řízení motoru NI 9512 je výhodnější než poskytnutá, obsahuje totiž konektory pro zapojení snímače „home“ pozice a tím pádem by nebylo nutné použít kartu NI 9401. Naopak motory ISM 7402 umožnily použití karty SISU 1004, čímž je v zařízení možno použít o jednu kartu méně. Koncové spínače translačních členů jsou místo na kartu SISU 1004 propojeny (stejně jako translační člen) na kartu NI 9401. Toto řešení totiž nevyžaduje úpravu kódu hradlového pole FPGA.

I/O karty se s krokovými motory propojovaly přímo, pro koncové snímače bylo nutné navrhnout a vytvořit obvod. Napájení 5 V pro tento obvod bylo vyvedeno přímo z volného konektoru zařízení cRIO.

3.2 Prvotní nastavení

Jako první úkon nutný ke splnění zadaných cílů bylo nutné připojit zařízení cRIO. Toto připojení je realizováno přes ethernetový kabel, kdy je cRIO dostupné z PC pomocí IP adresy. Prvotní konfigurace připojení byla provedena přes součást LabVIEW nazvanou NI MAX (National Instruments Measurement & Automation Explorer), která dokáže najít všechna připojená podporovaná zařízení a provézt jejich nastavení. Nejdříve bylo nutné u zařízení kompletně zformátovat disk a následně provést jeho reinstalaci se zvolenými součástmi (např. podpora SoftMotion nebo Scan engine). Dále bylo provedeno nastavení pevné IP adresy na hodnotu 192.168.1.2, přes ní mohl program LabVIEW k zařízení přistupovat.

Se správně připojeným a nakonfigurovaným zařízením už bylo možné začít vytvářet samotný program. Po založení nového projektu bylo do něj nutné cRIO přidat. Tato činnost je v LabVIEW značně automatizovaná, uživatel pouze řekne, že chce připojit nové zařízení a nastaví jeho přesný typ. Následně pouze nakonfiguruje IP adresu. Nově objevené zařízení by mělo vypadat stejně, jako je ukázáno na Obr. 17.



Obr. 17 Ukázka obsahu nově přidaného zařízení v Project Exploreru

Jak je z Obr. 17 vidět, zařízení v Project Exploreru obsahuje tři položky – Chassis, Dependencies a Build Specifications. Nás nejvíce zajímá položka Chassis, která představuje vlastní tělo zařízení. V ní se také nachází položka FPGA Target, která představuje programovatelné hradlové pole FPGA, o kterém se zmiňuji v předchozím textu. Do něj se vkládá VI soubor, ze kterého se vytvoří konfigurace hradla.

Dále je nutné vytvořit osy pohybu. Tento úkon je, jak je u LabVIEW zvykem, opět velmi jednoduchý – uživatel pouze na zařízení přidá tři nové položky typu SoftMotion Axis. Pro rotační motor byla vytvořena osa s názvem Rotace, pro translační motor 1 osa Axis 1 a pro motor 2 osa Axis 2

Nyní je třeba se zmínit ještě o jedné vlastnosti modulu SoftMotion. Tento modul má rozšířenou podporu některých vstupně-výstupních karet. Modul Ni 9501 je vyroben přímo firmou NI a je právě jeden z modulů s rozšířenou podporou. Modul SISU 1004 je vyroben partnerem firmy NI a jeho karta je sice podporována, avšak ne v takové míře, jako karta NI. V praxi to znamená, že při vytváření os je pro osu Rotace možné zadat přímo typ karty, pro osu Axis 1 a Axis 2 se jako typ udává položka Generic. U os byly následně nastaveny základní údaje, které byly vyčteny z katalogových listů. Názvy nastavených údajů a jejich hodnoty jsou v Tab. 4.

Rotace	Spline	Linear Interpolation
	Rated Phase Current	0,67 A/ϕ
	Units	°
	Steps per Unit	1 600
Axis 1	Spline	Linear Interpolation
	Units	mm
	Steps Per Unit	10 000
Axis 2	Spline	Linear Interpolation
	Units	mm
	Steps Per Unit	10 000

Tab. 4 Hodnoty nastavené pro jednotlivé osy

3.3 Konfigurace hradlového pole FPGA

Po prvotním nastavení je již možné začít se samotným vytvářením programu. Jak jsem se zmínil v teoretickém úvodu, cRIO podporuje několik módů – Scan Interface, FPGA Interface a hybridní mód. V této aplikaci nejsou na hradle nutné žádné aritmetické výpočty ani zpracování signálů a zároveň neprobíhá žádná komunikace, při které by byly nutné velké datové toky – z toho vyplývá, že by měl stačit nastavit mód Scan Interface. Bohužel není splněna základní podmínka – karty NI 9501 a SISU 1004 režim Scan Interface nepodporují, je tedy nutné hradlové pole nakonfigurovat. Ve výsledku byl tedy v programu použit hybridní mód, kdy byl vytvořen FPGA kód pro karty NI 9501 a SISU 1004, následně se kód zkompiloval, přičemž se hradlové pole automaticky nakonfigurovalo i na práci s kartou NI 9401.

Ze všeho nejdříve je ještě nutné nastavit připojené vstupně-výstupní karty. Tento úkon je opět poměrně jednoduchý – v případě, že byly na cRIO nainstalovány všechny ovladače správně, stačí v Project Exploreru přidat nový prvek typu C-Series Module a LabVIEW už samo připojené moduly vyhledá. Moduly jsou v projektu defaultně pojmenovány názvem Mod a číslem slotu, ve kterém jsou zasunuty. V našem případě je modul NI 9401 pojmenován Mod6, NI 9501 Mod7 a SISU 1004 Mod8.

Spolu s ovladači modulů se do prostředí LabVIEW nainstalovaly i tzv. examply neboli již vytvořené příklady použití. V nich byly vytvořeny VI soubory s kódem pro hradlové pole FPGA. Konfiguraci hradlového pole tedy nebylo nutné tvořit kompletně od začátku, ale kód stačilo zkopírovat a poupravit tak, aby fungoval i v případě manipulátoru. Celý kód se skládá z několika paralelně běžících smyček while, které plní různé úlohy.

Největší překážka, na kterou jsem v tomto bodě narazil, se týkala jedné ze základních vlastností FPGA. Na hradlovém poli zařízení cRIO totiž může být v jednu chvíli spuštěn pouze jeden kód FPGA. Každá I/O karta měla vlastní předpřipravený

příklad. V té chvíli existují dvě možnosti – program na FPGA může být nahraný, avšak ne spuštěný. Z čehož vyplývá, že by mohl být jeden kód vždy spuštěný, obsluhovat zvolenou osu, poté se vypnout a na hradle spustit kód pro osu druhou. To se nejeví jako vhodná volba z několika důvodů – hradlové pole by bylo na tak jednoduchou aplikaci velmi vytížené, hlavní nevýhoda ale spočívá v čase přepínání. Přepnutí konfigurace trvá na hradle zhruba 2 sekundy, což je pro ovládání os neakceptovatelný časový interval. Dále je nutné, aby osy byly ovladatelné zároveň, což toto řešení neumožňuje.

Bylo tedy potřeba oba dva kódy sjednotit. Tento úkol ulehčoval příklad dodaný pro modul SISU 1004. Modul umožňuje ovládat až 4 osy současně, což je právě v příkladu ošetřeno. Kvůli správné funkci je nutné v celém kódu dodržet pořadí os. Neboli je třeba dbát zvýšenou pozornost při používání indexů os. V kódu má osa Rotace index 0, osa Axis 1 má index 1, Axis 2 používá index 2.

Nyní je vhodné vysvětlit, jak FPGA kód pro SoftMotion funguje. Modul FPGA detekuje na zařízení připojené karty, které mají vnitřní proměnné (např. Status, Chyba atd.). Tyto bloky proměnných budou dále v textu pro větší přehlednost nazývány jako struktury. Každá karta má ještě vlastní soubor s globálními proměnnými, ty jsou stejné, jako vnitřní proměnné struktur. Globální proměnné modulu NI jsou skaláry, proměnné modulu SISU vektory o délce čtyř prvků – pro každou osu jeden. V jednotlivých smyčkách se porovnávají jednotlivé hodnoty vnitřních proměnných struktur a globálních proměnných, tyto výsledky se následně posílají jako vstupní parametry do funkčních bloků modulu SoftMotion. Celý kód FPGA funguje jako poměrně komplexní stavový automat, kdy globální proměnné představují předchozí stav, proměnné struktur stav současný. Samotný automat vykonává kód ne v závislosti na nastaveném stavu, ale na přechodu mezi dvěma stavy.

V následujících podkapitolách projdu jednotlivé cykly, které se nacházejí v kódu pro FPGA.

3.3.1 Fault Monitoring Loop

Tento cyklus zajišťuje zjišťování chybových stavů. Zjišťuje se status vnitřních proměnných struktur a zároveň z globálních proměnných a v případě chyby je do modulu SoftMotion poslána informace o chybě. Cyklus while obsahuje sekvenční strukturu, v jejímž prvním kroku čeká přístroj 250 mikrosekund (v tomto časovém intervalu se provádějí ostatní operace jiných smyček), následuje kontrola chybných stavů pro modul SISU 1004, v posledním kroku se kontrolují stavy modulu NI 9501.

3.3.2 Control Status Loop

Tento cyklus provádí 3 hlavní operace – kontroluje registry a rozpoznává změny stavů, provádí operace při změně stavů, nakonec generuje stavy do registrů. Celý kód se skládá z cyklu while, který zajišťuje neustálé opakování algoritmu, a sequence struktury, jejíž první krok obsahuje čekání po dobu 250 mikrosekund. Druhý krok obsluhuje všechny osy. Kontrolní registry z os jsou sjednoceny do vektoru, který je následně zaslán do cyklu for. V tomto cyklu je vytvořen stavový automat, který obsluhuje změny statusů. For cyklus způsobí, že stavový automat funguje vždy pouze pro jednu osu, po obsloužení stavu se přepne na osu následující. V samotném cyklu for se ještě nachází struktura case, která kontroluje, jaká osa je zrovna obsluhována, podle toho je vhodně zvolen správný automat.

3.3.3 Synchronization Loop

Cyklus obsluhuje synchronizace hodin FPGA s hodinami RT procesoru. Tyto cykly jsou vytvořeny dva, pro každou vstupně-výstupní kartu jeden. Každá smyčka totiž využívá jiných globálních proměnných a jejich spojením by mohly vzniknout výkonnostní problémy a chyby spojené s dataflow.

3.3.4 Mailbox Loop

Tento cyklus zprostředkovává odesílání a přijímání dat z a do modulu SoftMotion. Zároveň obsluhuje čtení a zápis do paměti hradlového pole FPGA.

3.3.5 DIO Port A

Cyklus je pouze pro modul SISU 1004. Obsahuje algoritmus, který obsluhuje digitální vstupy předního a zadního limitu os. Koncové snímače (Limit switch) jsou sice obsluhovány jinou kartou, cyklus však byl z důvodů možné nekompatibility kódu s modulem SISU ponechán. Cyklus také provádí digitální filtrování.

3.3.6 Position Loop

Tento while obsahuje algoritmus, který z modulu SoftMotion získává polohu. Díky ní poté dokáže SoftMotion „zjemnit pohyb“ motorů (tzv. spline).

3.3.7 Step Generation Loop

Tento cyklus generuje signály určující počet vygenerovaných kroků a směr pohybu. Ve smyčce jsou obsluhovány obě dvě vstupně-výstupní karty.

3.3.8 Další součásti

Kromě VI souboru, který konfiguruje hradlové pole FPGA, je v zařízení nahráno ještě dalších několik pomocných souborů.

První dva jsou soubory s definicí globálních proměnných. Jak už bylo zmíněno, tyto proměnné jsou jeden z prostředků, pomocí kterých pracuje kód pro konfigurace FPGA. Každá vstupně-výstupní karta má svůj vlastní soubor. Další pomocný soubor je subVI pro kartu SISU, který obstarává detekci chyb.

Po vyladění souboru pro konfiguraci FPGA se spustí kompilace, která vytvoří bitový soubor. Poměrně velká nevýhoda kompilace je doba jejího trvání, prvotní kompilace FPGA kódu trvala zhruba 50 minut, po částečné změně kódu se doba snížila na průměrnou hodnotu 35 minut. Na hradlovém poli lze poté spouštět přímo VI, stačí však pouze spouštět bitový soubor. Ten lze následně pomocí speciálních funkčních bloků spouštět přímo v uživatelských aplikacích, v případě manipulátoru se bitový soubor otevírá v hlavním ovládacím programu.

Z předchozího textu vyplývá, že VI, které obsahuje kód pro konfiguraci FPGA, se spolu se všemi jeho pomocnými soubory v projektu nemusí vůbec nacházet, tyto soubory jsou potřeba pouze pro vytvoření bitového souboru. Tyto soubory však byly z důvodu větší názornosti v projektu ponechány. Zároveň byly soubory v projektu ponechány pro případ, že by byly v budoucnosti zjištěny chyby v kódu.

Pro případ, kdy uživatel nebude chtít pro kontrolu manipulátoru používat dodaný ovládací program se v projektu nachází taktéž soubor RT.vi. Tento velmi jednoduchý program při svém startu otevře bitový soubor, následně se dostane do nekonečné smyčky, která je opuštěna při zmáčknutí tlačítka STOP. Nakonec je bitový soubor zavřen.

3.3.9 Shrnutí vytvoření konfigurace FPGA

Pro funkčnost modulu SoftMotion bylo potřeba vytvořit správnou konfiguraci hradlového pole FPGA. Kód nebylo potřeba vymýšlet od začátku, stačilo pouze provést úpravy v příkladu dodaném k modulu SISU 1004.

Vytváření funkčního kódu velmi znesnadňovala základní vlastnost FPGA modulu, a to nemožnost jakéhokoliv debugování. Jestliže kód nefungoval, bylo kvůli rozsáhlosti kódu obtížné detekovat přesné místo chyby, LabVIEW FPGA totiž dokáže detekovat pouze syntaktické chyby např. typu spoj ve vzduchu atd. Opravy algoritmu byly dále znesnadňovány velmi dlouhým časem kompilace, díky kterému mohlo odstranění několika chyb trvat řádově hodiny.

Výsledný zkompilovaný kód je v tzv. bitovém souboru. Ten je otevírán přímo v hlavní aplikaci, zároveň však může být spuštěn aplikací RT.vi. Zbylé soubory se už tedy nemusí na zařízení cRIO ani v projektu vůbec nacházet, kvůli možným

pozdějším úpravám či opravám chyb a kvůli větší názornosti jsem se rozhodl pro jejich ponechání.

3.4 Ovládací software

Ještě, než bude vysvětlen chod programu ovládajícího manipulátor, je nutné uvést požadavky, jaké jsou na tento program kladeny.

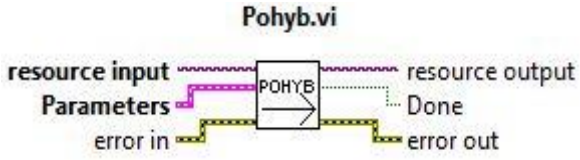
- Ovládání dvou nebo tří nezávislých pohybových os – jedna translace + rotace nebo dvě translace + rotace, volba závisí na uživateli
- Pohyb by měl být absolutní – není zadáváno, o kolik se má manipulátor posunout, ale na jakou pozici
- Možnost nastavit směr otáčení rotačního motoru – kladný, záporný nebo po nejkratší dráze
- Neustálá kontrola stavu koncových snímačů
- Umožnění nastavit manipulátor do inicializační polohy
- Možnost nastavit novou referenční pozici, na které se nastaví nulová hodnota absolutních souřadnic
- Možnost nastavit jemnost pohybu (tj. zrychlení při začátku a konci pohybu)
- Vizualizace, při pohybu vizuální zvýraznění právě se pohybujících os

Původní vize programové architektury byla, že celý program bude fungovat jako jednoduchý stavový automat, kdy funkce, které musí být kontrolovány po celou dobu běhu programu (např. kontrola spínačů), budou mít své vlastní cykly. S přibývajícími funkcemi však program začal být značně rozsáhlý, velmi nepřehledný a práce s prostředím přestala být plynulá – sebemenší změny kódu program dlouho zpracovával. Program tedy musel být přepracován na architekturu front. V tomto případě tedy program obsahuje celkem dvě smyčky while. Jedna obstarává zachytávání příkazů od uživatele a kontroluje požadované činnosti. Následně posílá přes frontu příkazy druhé smyčce, která tyto příkazy vykonává. Jeden cyklus funguje jako komunikační, druhý jako realizační (dále budou tyto cykly pro jednoduchost takto označovány).

3.4.1 SubVI

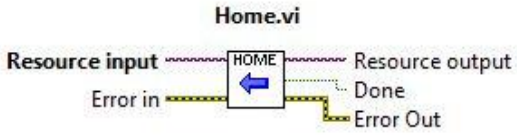
Aby byl program jednodušší a lépe čitelný, byly některé funkce dány do podprogramů. V této kapitole budou přítomné subVI jednotlivě popsány.

Nejzákladnějším subVI, který se v programu nachází, je podprogram s názvem Pohyb. Jak už název napovídá, zajišťuje pohyb osy o zadané hodnoty. Jako vstup přijme referenci osy, se kterou má pohybovat a cluster se zadanými parametry pohybu (tj. rychlost, vzdálenost, zjemnění a směr). Na výstupu má podprogram opět referenci osy (aby se jednodušeji řetězil) a proměnnou typu boolean informující o dokončení pohybu. Toto VI může fungovat i samo o sobě. Jestliže je spuštěno (se správně zadanými parametry) a jestliže je zároveň na hradlovém poli otevřen bitový soubor, probíhá pohyb os – samozřejmě pouze s relativním zadáváním polohy. Podprogram tedy může být používán i v jiných uživatelských aplikacích.



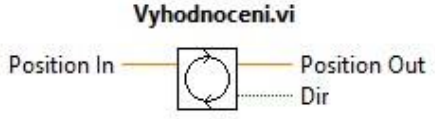
Obr. 18 Podprogram Pohyb.vi

SubVI s názvem Home se v hlavním programu vůbec nevyskytuje, je vytvořeno pouze z důvodů použití v jiných uživatelských aplikacích. Slouží k inicializování motorů. Uživatel na vstupu nastaví referenci motoru, který chce inicializovat, ten tuto akci provede a na výstupu s názvem Done program nastaví logickou jedničku.



Obr. 19 Podprogram Home.vi

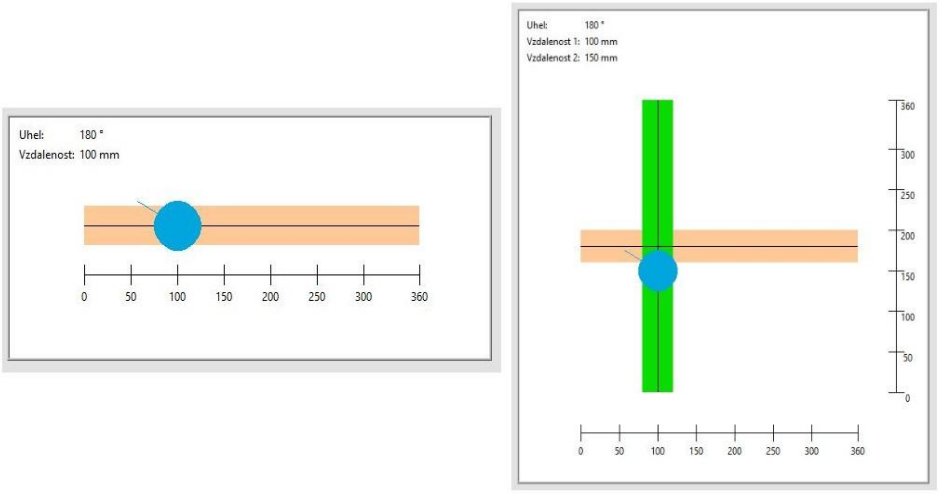
Další použité subVI má název Vyhodnoceni. Tento podprogram slouží k výpočtu dráhy nejkratší k zadanému úhlu – zajišťuje, že rotační motor na daný úhel pojede vždy nejkratší dráhou. Algoritmus podprogramu je velmi jednoduchý, v hlavním programu však kód zabíral příliš mnoho místa. SubVI dostane na vstupu vzdálenost, o kterou se má motor pootočit a tu zkontroluje (a případně upraví), aby nebyla větší než 180°.



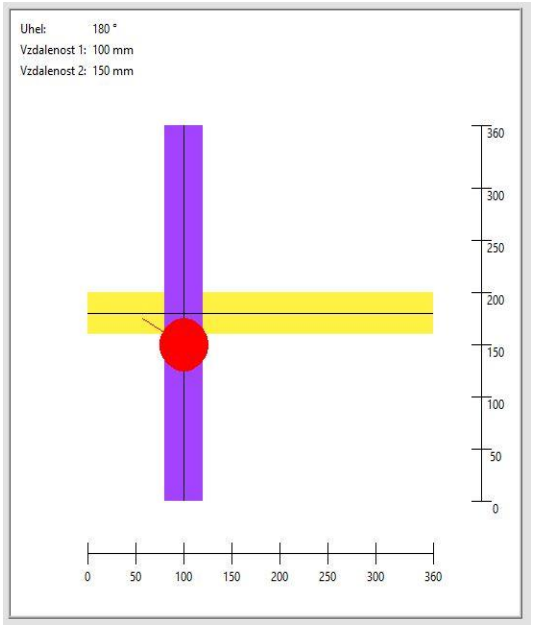
Obr. 20 Podprogram Vyhodnoceni.vi

SubVI Vizualizace má, jak už název napovídá, za úkol zobrazovat aktuální pozici motorů. SubVI má jako vstupní parametry vzdálenost motorů od počáteční pozice, vstup Color slouží k přepínání barev při běhu motoru, T. motors quantity určuje, jestli se bude zobrazovat vizualizace pro jeden nebo dva připojené translační

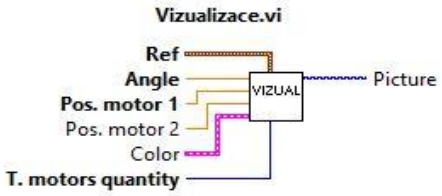
motory. Vstup Ref slouží k vizualizaci referenční pozice. Vizualizace pomocí tohoto vstupu upravuje číselné zobrazení aktuální pozice (viz např. Obr. 22 vlevo nahoře). Výstup z tohoto podprogramu je proměnná typu Picture. Kód tohoto podprogramu je poměrně rozsáhlý, avšak algoritmus je jednoduchý, pouze se generuje počátek nebo střed základních geometrických útvarů (kruh, obdélník a úsečka) v závislosti na pozici motorů a podle vstupu Color se u těchto obrazců při generování mění barva. Vizualizace.vi obsahuje ještě dvě další subVI (Legenda-X a Legenda-Y), ty slouží ke generování měřítka na osách.



Obr. 22 Vizualizace pro jeden (vlevo) a pro dva (vpravo) translační motory



Obr. 21 Vizualizace se změněnými barvami, značí pohyb všech motorů



Obr. 23 Podprogram Vizualizace.vi

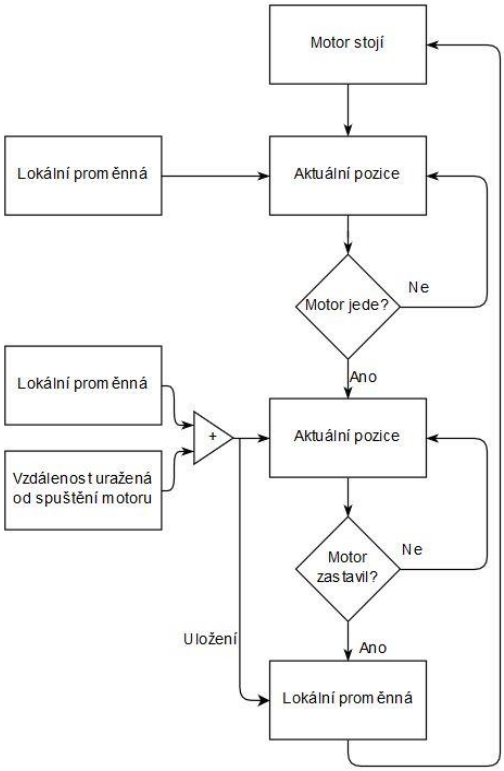
Poslední subVI, které se v projektu nachází, se jmenuje ActPosition. Tento podprogram umožňuje pro každou osu pohybu zaznamenávat pozici, na které se právě nachází. Podprogram vznikl hlavně kvůli úspoře místa a větší přehlednosti v hlavním programu, sám o sobě by jeho algoritmus neměl žádnou funkci. Algoritmus pro absolutní pohyb, na který je podprogram přímo napojen, bude vysvětlen později v kapitole 3.4.2.

Jestliže chce uživatel používat vlastní aplikaci, musí napřed spustit program RT.vi zmíněný v kapitole 3.3.8. Následně už může podle potřeby používat subVI Home k inicializaci motorů a Pohyb k jejich relativnímu pohybu. Při užívání subVI Pohyb je však doporučena obezřetnost při zadávání vzdálenosti posuvu, pohyb je totiž relativní a podprogram vůbec nepoužívá zavedené koncové snímače. Uživatel tak velmi lehce může překročit rozsah translačních motorů.

3.4.2 Algoritmus absolutního pohybu

Ještě, než budou vysvětleny funkce komunikačního a realizačního cyklu, je nutné přiblížit algoritmus, pomocí kterého program provádí absolutní pohyb. Jak bylo uvedeno v kapitole 2.3.4.2, modul SoftMotion umožňuje uživateli nakonfigurovat koncové snímače a následně se všemi funkčními bloky určenými pro pohyb pracovat v režimu absolutního pohybu, využívat funkční bloky určené pro inicializaci pozice, v případě aktivace spínače automaticky zastavovat pohyb atd. Toto vyžaduje jak správnou úpravu kódu FPGA, tak konfiguraci os. Tento úkol dále komplikuje skutečnost, že koncové snímače nejsou připojeny přímo k I/O kartám ovládajícím motory, ale ke kartě externí. Tento úkol se tedy pro mě stal tak složitým a komplexním, že se mi ho nepovedlo splnit a absolutní pohyb a další funkce související s koncovými snímači musely být dosaženy jiným způsobem.

V hlavním programu jsou vytvořeny tři lokální proměnné (pro každý motor jedna, ActStablePositionR, ActStablePositionT1 a ActStablePositionT2), které zaznamenávají poslední pozici, na které se motory nacházely, než bylo stisknuto tlačítko Start. V průběhu pohybu tato proměnná ukazuje stále stejnou hodnotu, subVI ActPosition ukazuje na výstupu hodnotu lokální proměnné sečtenou se vzdáleností, kterou motor za tento pohyb prozatím urazil. Ve chvíli, kdy se motor zastaví, se lokální proměnná upraví tak, aby ukazovala reálnou pozici motoru – ovšem pouze do chvíle, než bude spuštěn další pohyb. Celý tento algoritmus je zobrazen i na Obr. 24.



Obr. 24 Algoritmus zapisování do lokální proměnné pro jeden motor

Na vývojovém diagramu zobrazeném na Obr. 24 je mimo jiné abstraktní proměnná Aktuální pozice. Tato proměnná se v programu nikde fyzicky nevyskytuje, je jí myšlena pozice motoru, na které se právě nachází. Tato hodnota je předávána do subVI Vizualizace jako jeden ze vstupů.

Program si tak pamatuje pozici, na které se motor nachází, když stojí. Když uživatel zadá pozici, na kterou se má motor posunout, program ji porovná s aktuálním umístěním a podle toho vypočítá vzdálenost, o kterou se má motor posunout.

3.4.3 Komunikační cyklus

Jak již bylo zmíněno v předchozím textu, tento cyklus realizuje komunikaci s uživatelem, kontroluje požadované funkce v programu, které vyhodnocuje a následně odesílá příkazy do realizačního cyklu.

Tento cyklus je vytvořen jako stavový automat se dvěma stavy. Při spuštění programu se automat přepne do stavu Init, ve kterém inicializuje na správnou hodnotu proměnné kriticky nezbytné pro správný běh programu a nastaví minimální a maximální hodnoty, které může uživatel zadat do prvků pro nastavování pozice motorů. Následně se automat přepne do stavu Cekej, ve kterém se pohybuje až do ukončení celého programu.

Tento stav obsahuje event strukturu, která kontroluje a následně reaguje na akce, které by uživatel nebo program mohli vyvolat. Tato struktura obsahuje celkem 8 událostí, na které reaguje.

První událost je tzv. Timeout. Ten nastane, jestliže neproběhla žádná událost po předem definovanou dobu – ta je v programu nastavena na hodnotu 10 ms. V programu jsou v timeoutu takové události, které se musí provádět při celém běhu programu – kontroluje se zde stav koncových snímačů, probíhá zde čtení a zápis lokálních proměnných ActStablePosition a provádění vizualizace, nakonec se zde po každých pěti provedeních timeoutu provede změna lokální proměnné Blik (bude vysvětleno dále).

Další událostí je stisknutí tlačítka Start. Po stisknutí se do realizačního cyklu přepošle příkaz k přepnutí na stav Jedu (viz kapitola 3.4.4) a znepřístupní se k editování ta tlačítka uživatelského prostředí, která slouží k nastavení parametrů pohybu nebo k jeho spuštění, a povolí se tlačítko Stop.

Stisknutím tlačítka Konec se vyvolá událost, která nastaví celé uživatelské prostředí do defaultního stavu, a zakáží a povolí se stejná tlačítka jako v případě stisknutí tlačítka Start. Tyto úkony jsou provedeny proto, aby byl celý program při příštím spuštění VI v požadovaném stavu. Zároveň se provede zastavení pohybu všech motorů. Eventu lze dosáhnout i bez použití tlačítka Konec, nastane i v případě, že se v programu kdekoliv vyskytne chyba. Jako poslední tento event způsobí, že je do realizační smyčky zaslán příkaz k přepnutí do stavu Konec (viz kapitola 3.4.4) a zároveň je komunikační smyčka ukončena. To způsobí vypnutí celého programu.

Další událost se zabývá stisknutím tlačítka Domů. Při něm se opět zakáží všechny prvky, jako při spuštění tlačítka Start, zároveň se inicializují pomocné proměnné zajišťující nastavení nové nulové pozice na osách a reinicializují se limity pro nastavení pozic motorů. Nakonec se do realizačního cyklu odešle příkaz ke spuštění stavu Domů (viz kapitola 3.4.4).

Další ošetřená událost se týká změny ovládacího prvku Pocet t. motorů. Při ní se zakáží nebo povolí (podle zvolené možnosti) prvky uživatelského prostředí, které nastavují parametry pro pohyb druhého translačního motoru.

Stiskem tlačítka Stop, případně sepnutím některého koncového snímače, se vyvolá event, který podle stavu snímačů (nebo tlačítka Stop) zastaví motory. Tento event je využíván při zadání příkazu k nalezení inicializační pozice – motory jedou tak dlouho, dokud se nesepne koncový snímač, čímž se tato událost vyvolá a motor se zastaví.

Event Blik nemůže být vyvolán uživatelem. Nastává každý pátý průchod eventem Timeout. Při něm program kontroluje, zda se pohybují motory, a podle toho přepíná hodnotu další pomocné lokální proměnné. Její hodnota zajišťuje, že se mění barva jednotlivých motorů při vizualizaci. Jednoduše řečeno tedy tato událost určuje, zda má vizualizace ukazovat, že se motor pohybuje, či nikoliv.

Poslední event obsluhuje stisk tlačítka Nová referenční poloha. Při ní se nastaví nulová pozice na aktuální pozici všech motorů a zároveň se změní maxima a minima v uživatelských prvcích nastavující požadovanou pozici – samozřejmě pouze u translačních motorů. Je-li tedy translační motor na pozici 50 mm, rozsah motoru se po stisku tlačítka změní z 0 mm – 360 mm na hodnotu -50 mm – 310 mm.

3.4.4 Realizační cyklus

Jak už bylo zmíněno dříve, realizační cyklus přijímá příkazy z komunikačního a vykonává je. Konkrétněji řečeno je cyklus navrhnut jako stavový automat o šesti stavech, který buď vykonává příkaz nebo čeká příkaz další. V následujících odstavcích bude pouze popsána funkce jednotlivých stavů, přechody mezi nimi jsou zobrazeny na Obr. 25.

První stav je stav Inicializace. Ten slouží pouze pro detekci, že celý manipulátor byl inicializován do základního stavu a proměnné byly upraveny, díky čemuž může být spuštěna správná detekce pozice atd.

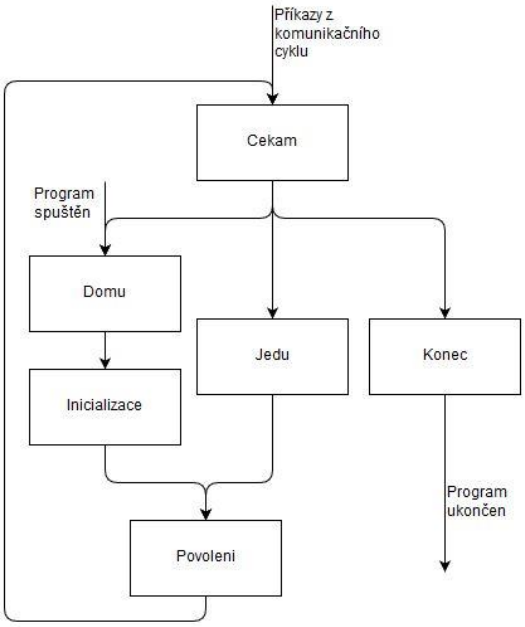
Další stav je nazván Domu. Při něm se v programu nastaví, že manipulátor není inicializován a tím pádem nemají probíhat zápisy do lokálních proměnných a vizualizace má být pozastavena. Následně se spustí pohyb motorů. Ten probíhá do té doby, než realizační cyklus detekuje sepnutí koncových snímačů, v té chvíli je pohyb zastaven. Algoritmus je nastaven tak, aby se translační motory inicializovaly do polohy 0 mm, rotační motor na úhel 2°. Tento stav je první, který se spustí při spuštění cyklu neboli při spuštění programu. To znamená, že první, co manipulátor při spuštění programu provede, je inicializace na domovskou pozici.

Stav Cekam má pouze jediný úkol. Jako jediný kontroluje, zda nebyl do fronty poslán příkaz z komunikačního cyklu.

Ve stavu Jedu probíhá samotný pohyb motorů. Stav polohu zadanou uživatelem porovná s polohou nastavených referenčních bodů (pozic nulového bodu), následně s aktuální pozicí motorů a následně do subVI Pohyb pošle hodnoty, o které se mají motory posunout.

Stav Povoleni, jak už jeho název napovídá, povoluje prvky uživatelského prostředí. Jak bylo zmíněno v kapitole 3.4.3, při spuštění jakéhokoliv pohybu (tj. po stisku tlačítka Start nebo při hledání pozice „home“) se v programu zakáží prvky prostředí, kterými uživatel nastavuje parametry pohybu nebo pohyb spouští a povoluje se tlačítko Stop. Stav povolení naopak prvky pro nastavení nebo spuštění pohybu povolí a tlačítko Stop zakáže.

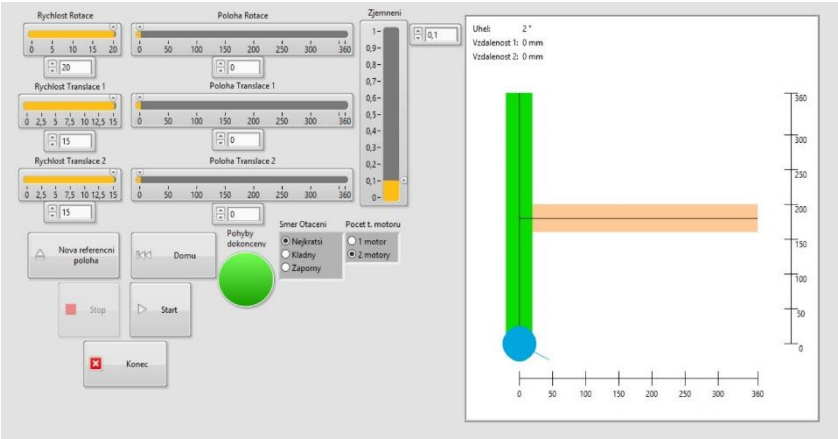
Poslední přítomný stav se nazývá Konec. Ten žádnou činnost nevykonává, pouze ukončí smyčku. Tento stav může nastat pouze v případě, že byl ukončen i komunikační cyklus, po jeho vykonání se tedy vypne celý program.



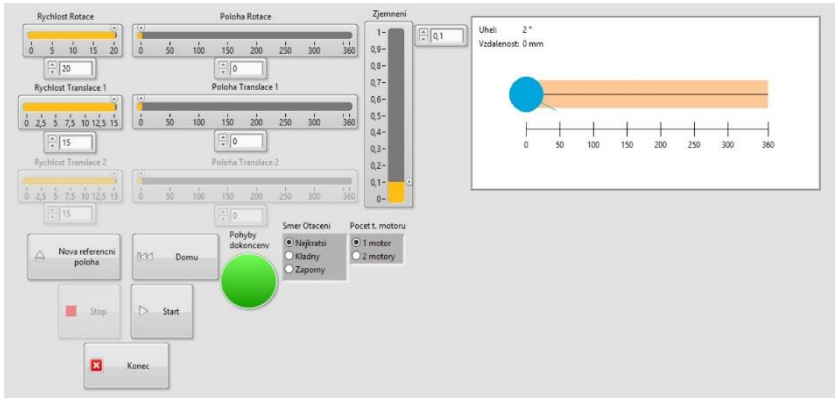
Obr. 25 Vývojový diagram stavového automatu realizačního cyklu

3.4.5 Uživatelské prostředí

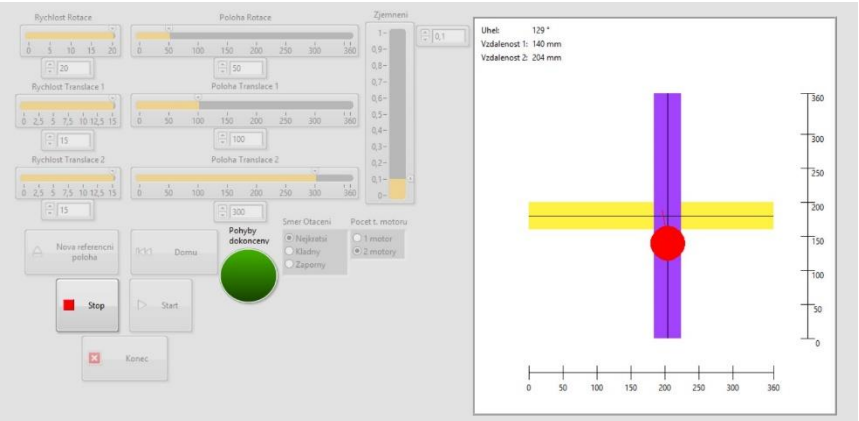
Uživatelské prostředí je zobrazeno na Obr. 27, Obr. 26 a Obr. 28. Po spuštění programu se motor nastaví na inicializační pozici, jediné, co uživatel může provést je stisknutí tlačítka STOP. Tím se pohyb zastaví, program se v té chvíli však chová, jako by byl manipulátor inicializován úspěšně. Po dokončení pohybu se povolí nastavení parametrů pohybu a počet použitých translačních motorů. Po přepnutí počtu motorů na dva jsou motory opět inicializovány, po přepnutí zpět na jeden však nikoliv. Po nastavení parametrů se pohyb spustí stiskem tlačítka Start, v tomto případě se opět zakáží všechna tlačítka s výjimkou tlačítka Stop. Program lze ukončit pouze v případě, že neprobíhá žádný pohyb.



Obr. 27 Uživatelské prostředí pro 2 translační motory, jestliže neprobíhá žádný pohyb



Obr. 26 Uživatelské prostředí pro 1 translační motor, jestliže neprobíhá žádný pohyb



Obr. 28 Uživatelské prostředí pro 2 translační motory, jestliže probíhá pohyb

3.4.6 Shrnutí návrhu ovládacího programu

Původně plánovaná architektura pro ovládací software byl jednoduchý stavový automat. S přibývajícími funkcemi však tato architektura začala být velmi nevhodná a program musel být předělán na architekturu dvou paralelních smyček, kdy jedna analyzuje nutné kroky a posílá pomocí fronty příkazy k jejich vykonání smyčce druhé. První smyčka analyzuje nutné kroky pomocí event struktury, kdy v eventu Timeout realizuje vizualizaci a úpravu proměnných pro absolutní pohyb, ostatní eventy mají za úkol provádět úkony a odesílat příkazy druhé smyčce na základě stisku tlačítka nebo změny pomocné proměnné. Druhá smyčka je stavový automat se šesti stavy, jeden stav přijímá úkoly od smyčky první. Podle toho se přepíná do příslušných stavů a po jejich vykonání opět čeká na příkazy.

Kvůli zlepšení výkonu a úspoře místa a tím pádem i zlepšení čitelnosti byly vytvořeny celkem čtyři subVI, které realizují a usnadňují pohyb, vizualizaci a výpočty v algoritmu pro pohyb v absolutních souřadnicích. Další dvě subVI jsou určeny pro použití v podprogramu pro vizualizaci, generují měřítko zobrazené na osách.

Jestliže je FPGA správně nakonfigurováno a modul SoftMotion správně nastaven, stav koncových snímačů si hlídá samotný modul. V tomto případě programátor nemusí koncové snímače nijak hlídat, pouze používá funkční bloky, např. pro hledání inicializační pozice, pro absolutní pohyb, zároveň lze nastavit, aby se v případě sepnutí motor automaticky zastavil. Nastavení FPGA a SoftMotion se však ukázalo být tak složitým a komplexním úkolem, že se mi ho nepovedlo splnit. Kvůli tomu muselo být velké množství funkcí, které SoftMotion zvládá, ošetřeno rozšířením hlavního ovládacího programu. Díky tomu je ztíženo i používání funkčních bloků modulu SoftMotion v případě, když by uživatel nechtěl využívat dodané aplikace, ale chtěl by vytvořit aplikaci vlastní.

Vytvořený program je díky své architektuře nevhodný k použití jako subVI (tj. k automatizování úloh), je určen pouze k manuální obsluze. Pro případ, že by chtěl uživatel pohyb automatizovat, jsou v projektu přítomné tři VI. Soubor RT.vi slouží k otevření bitfile. Následně je v projektu přítomno subVI Home, které inicializuje zvolený motor. Poslední soubor Pohyb.vi slouží k relativnímu pohybu zvoleného motoru o nastavené parametry. Při používání tohoto subVI je však doporučována obezřetnost, podprogram totiž nepodporuje snímání koncových snímačů, takže může velmi lehce dojít k překročení rozsahu translačních motorů. Zároveň není zavedeno tlačítko Stop.

3.5 Shrnutí praktické části

Praktická část se nejdříve zaměřuje na porovnání poskytnutého hardwaru s tím, který je v teoretické části vyhodnocen jako nejvhodnější. Poskytnuté zařízení cRIO 9068 svými možnostmi a funkcemi vysoce převyšuje požadavky, které jsou manipulátorem kladeny. V případě nákupu zařízení by rozhodně bylo vhodnější díky několikanásobně nižší ceně použít zařízení cRIO 9063 doporučené v teoretické části. Rovněž karta určená pro řízení motoru NI 9512 je výhodnější než poskytnutá, obsahuje totiž konektory pro zapojení snímače „home“ pozice, díky čemuž by nebylo nutné použít kartu Ni 9401. Naopak motory ISM 7402 umožnily použití karty SISU 1004, čímž je v zařízení možno použít o jednu kartu méně. Následně se kapitola zabývala propojením krokových motorů a I/O karet.

Dále se praktická část zabývá prvotní konfigurací hardwaru, tzn. připojením cRIO k PC a jeho nastavením a následným vytvořením projektu v LabVIEW a nakonfigurováním pohybových os.

Předposlední oblast se týká vytvoření konfigurace FPGA. Tuto činnost velmi ulehčovala přítomnost příkladů vytvořených výrobcem – stačilo je správně poupravit. Vytváření funkčního kódu naopak znesnadňovala nemožnost kód debugovat spolu s kombinací s kompilačními časy v řádech desítek minut. Díky komplexnosti a rozsáhlosti kódu taktéž nebyly do modulu SoftMotion implementovány koncové snímače, čímž mnoho funkcí modulu není využito a pro jejich zavedení musel být upraven hlavní ovládací program.

Poslední kapitola se zabývá návrhem řídicí/vizualizační aplikace, kde je popisována architektura programu a hlavní algoritmy, které program využívá. Výsledný program obsahuje všechny požadované funkce, při správném nastavení FPGA a SoftMotion by však mohl být značně zjednodušen. Program taktéž není vhodný k použití jako subVI, z tohoto důvodu se v projektu nacházejí 3 subVI spouštějící konfiguraci hradlového pole FPGA, zajišťující inicializaci zvoleného motoru a provádějící relativní pohyb zvoleného motoru.

4 ZÁVĚR

Cílem této práce bylo navrhnout ovládací software pro mechanický manipulátor řízený jednotkou cRIO. Manipulátor se skládá ze dvou translačních a jednoho rotačního motoru, tyto motory jsou krokové. Zařízení cRIO má vloženy dvě I/O karty určené k ovládání motorů a jednu kartu pro detekci stavu koncových snímačů.

V teoretické části práce byl stručně vysvětlen princip fungování krokových motorů a způsob jejich zapojení a řízení. Následně byl analyzován hardware nabízený na internetových stránkách výrobce National Instruments, který by byl k řízení manipulátoru nejvhodnější. Nakonec byly vysvětleny základní pojmy týkající se prostředí LabVIEW, které se vyskytovaly v praktické části práce.

Praktická část se nejprve zabývala porovnáním hardwaru, který byl v teoretické části vyhodnocen jako nejvhodnější, s hardwarem poskytnutým. Výsledky teoretického rozboru se s poskytnutým hardwarem shodovaly pouze v případě I/O karty určené k detekci stavu koncových snímačů. Rozdíl byl částečně způsoben translačními akčními členy, které jsou jiného typu, než bylo v teoretické části předpokládáno. Poskytnuté zařízení cRIO 9068 svými možnostmi a funkcemi vysoce převyšuje požadavky, které jsou manipulátorem kladeny. V případě nákupu zařízení by rozhodně bylo vhodnější díky několikanásobně nižší ceně použít zařízení cRIO 9063 doporučené v teoretické části.

Následný text praktické části se týká návrhu konfigurace FPGA a algoritmy použitými v řídícím programu.

Pro chod řídícího programu musela být vytvořena konfigurace FPGA, úkol velmi ulehčila přítomnost předpřipravených příkladů kódu. Konfigurace je spouštěna z bitového souboru, v projektu však z důvodu názornosti byly ponechány i nezkompilované soubory s kódem.

Samotný program pro ovládání manipulátoru byl vytvořen s pomocí dodatečného modulu SoftMotion. Díky veliké komplexnosti nastavení os v projektu a úpravě kódu FPGA nebyly v SoftMotion nakonfigurovány koncové snímače polohy, což znemožnilo používat všechny funkce, které modul nabízí a tyto funkce musely být zavedeny do algoritmů hlavní řídící aplikace. Všechny požadované funkce aplikace obsahuje, v případě nakonfigurování limitních snímačů by však mohla být mnohem jednodušší a méně náročná na výpočetní výkon. Program samotný není určen k použití jako subVI, tzn. je určen pouze k manuální obsluze. Jestliže by uživatel chtěl např. pohyb motorů automatizovat, nacházejí se v projektu soubory RT.vi, Home.vi a Pohyb.vi.

Všechny hlavní cíle práce tedy byly splněny. Největší příležitost k dalšímu zlepšení spočívá v úpravě konfigurace programovatelného hradlového pole a nastavení modulu SoftMotion, aby modul komunikoval přímo s koncovými snímači a chování motorů s nimi spojené nemuselo být ošetřeno v hlavní řídící aplikaci.

5 POUŽITÉ ZDROJE

[1] **National Instruments.** *National Instruments.* [Online] 2017. [Cited: 2 Květen 2017.] http://www.ni.com/.

[2] **National Instruments.** What Can You Do With LabVIEW? *National Instruments.* [Online] 2016. [Citace: 16. Prosinec 2016.] http://www.ni.com/labview/why/.

[3] **National Instruments.** What Is LabVIEW? *National Instruments.* [Online] 16. Srpen 2013. [Citace: 15. 12 2016.] http://www.ni.com/newsletter/51141/en/.

[4] **Vlach, Jaroslav, Havlíček, Josef a Vlach, Martin.** *Začínáme s LabVIEW.* Praha : BEN - technická literatura, 2008. 978-80-7300-245-9.

[5] **Zezulka, František.** *Prostředky průmyslové automatizace.* Brno : VUTIUM, 2004.

[6] **National Instruments.** *NI LabVIEW for CompactRIO Developer's Guide.* místo neznámé : Niational Instruments, 2016.

[7] **Ericsson.** Stepper Motor Basics. *solarbotics.net.* [Online] 18 Říjen 2016. http://solarbotics.net/library/pdflib/pdf/motorbas.pdf.

[8] **jrt.** Krokové motory 1 – typy motorů. *RoboDoupě.* [Online] 11. Září 2013. [Citace: 5. Květen 2017.] http://robodoupe.cz/2013/krokove-motory-1-typy-motoru/.

[9] **Lin Engineering.** Wiring Connections. *Lin Engineering - Your Stepper Motor Specialist.* [Online] 2016. [Cited: 5 Květen 2017.]

[10] **Sisu Devices.** SISU-1004. *Sisu Devices.* [Online] [Citace: 2. Květen 2017.] http://www.sisudevices.com/wp-content/uploads/2015/09/SISU-1004-OPERATING-INSTRUCTIONS-AND-SPECIFICATIONS_200006C-01.pdf.

[11] **National Instruments.** OPERATING INSTRUCTIONS AND SPECIFICATIONS NI 9501. *National Instruments* . [Online] 15 Srpen 2015. [Cited: 4 Květen 2017.] http://www.ni.com/pdf/manuals/375479f.pdf.

[12] **Standa.** 8MR190-2 - Motorized Rotation Stage. *Standa.* [Online] 2015. [Citace: 26. Prosinec 2016.] http://www.standa.lt/products/catalog/motorised_positioners?item=244.

[13] **Standa.** 8MT195 - Long-travel Motorized Linear Stage. *Standa.* [Online] 2015. [Citace: 10. Prosinec 2016.] http://www.standa.lt/products/catalog/motorised_positioners?item=259.

Vývojové diagramy a elektrické schéma (Obr. 14, Obr. 16, Obr. 24 a Obr. 25) byly vytvořeny na internetové stránce https://www.draw.io/