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

ANALÝZA SLABÝCH STRÁNEK LASEROVÉHO 2D VIBROMETRU A JEHO VYLEPŠENÍ
WEAKNESSES ANALYSIS OF A 2D LASER VIBROMETER AND ITS IMPROVEMENT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. Ondřej Vybíral

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Stanislav Pikula, Ph.D.

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Ondřej Vybíral ***ID:*** 195466

Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Analýza slabých stránek laserového 2D vibrometru a jeho vylepšení

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je odhalení neoptimálně fungujících částí stávajícího laserového 2D vibrometru, který využívá software v LabVIEW. Student musí prostudovat existující HW a SW řešení, zorientovat se v něm, navrhnout úpravy a ty realizovat.

Zadání diplomové práce lze shrnout do následujících bodů:

1. Prostudujte stávající HW a SW řešení v dokumentaci zařízení (viz literatura) a při práci se samotným zařízením. Zdokumentujte neoptimálně fungující části systému. Ať už z funkčního či uživatelského hlediska.
2. Navrhněte úpravy SW v LabVIEW, které povedou k optimalizaci programu, jeho lepší funkčnosti a vyššímu uživatelskému komfortu.
3. Realizujte nejpodstatnější z navržených úprav, otestujte plnohodnotnou funkčnost výsledného řešení a porovnejte původní a vaše řešení pro jednoznačné zhodnocení přínosu.

DOPORUČENÁ LITERATURA:

[1] TOMEK, Tomáš Laserový 2D skener: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2016. 89 s. Vedoucí práce byl doc. Ing. Petr Beneš, Ph.D.

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 17.5.2021

Vedoucí práce: Ing. Stanislav Pikula, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Diplomová práce navazuje na práci Ing. Tomka, kterou stručně popisuje, a analyzuje slabé stránky tohoto původního řešení. Na základě rozboru jsou navrženy opravy slabých míst a návrhy nových funkcionalit. Dále je popsána jejich realizace a oprava dalších nalezených nedostatků, včetně implementace do původního programu. Výsledkem je lepší uživatelská přívětivost programu a jeho lepší funkčnost. Ověření funkčnosti nových vylepšení je otestováno s využitím kontrolního měření harmonicky buzeného nosníku. Poslední kapitola obsahuje náměty a doporučení pro další rozvoj softwaru.

KLÍČOVÁ SLOVA

Polytec OFV-505, Polytec OFV-5000, LabVIEW, laserový 2D vibrometr, Thorlabs GVS012

ABSTRACT

The master's thesis follows the work of Ing. Tomek, who briefly describes and analyzes the weaknesses of this original solution. Based on the analysis, fixes for vulnerabilities and proposals for new functionalities are proposed. It also describes their implementation and correction of other shortcomings found, including implementation into the original program. The result is better user friendliness of the program and its better functionality. Verification of the functionality of the new improvements is tested using a control measurement of a harmonically excited beam. The last chapter contains suggestions and recommendations for further software development.

KEYWORDS

Polytec OFV-505, Polytec OFV-5000, LabVIEW, laser 2D vibrometer, Thorlabs GVS012

VYBÍRAL, Ondřej. *Analýza slabých stránek laserového 2D vibrometru a jeho vylepšení*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2021, 76 s. Diplomová práce. Vedoucí práce: Ing. Stanislav Pikula, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Ondřej Vybíral
VUT ID autora:	195466
Typ práce:	Diplomová práce
Akademický rok:	2020/21
Téma závěrečné práce:	Analýza slabých stránek laserového 2D vibrometru a jeho vylepšení

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné 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/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Stanislavovi Pikulovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V roce 2016 Ing. Tomek publikoval svou práci „Laserový 2D skener“ [1]. V této i ve své bakalářské práci [2] se věnoval návrhu a realizaci 2D laserového vibrometru. Tato práce byla velmi zdařilá a tak ji začali používat a rozvíjet akademičtí pracovníci Ústavu automatizace a měřicí techniky.

Bylo proto nutné tuto práci dále rozvíjet. Původní práce měla jisté nedostatky jak na straně uživatelského rozhraní, tak i určité roztříštěnosti vzniklé nesystémovým vývojem. Bylo proto nutné provést hloubkovou a systémovou analýzu současného stavu vibrometru, zejména jeho programu.

Hlavním úkolem této diplomové práce je nastudovat původní práci a nalézt slabé místa stávajícího řešení laserového 2D vibrometru, popsat je a hlavní z nich realizovat. Podle zadání je také strukturována osnova práce. Kapitola 1 se věnuje popisu systému, obsahuje základní popis hardwarového a softwarového vybavení, včetně výčtu potřebných komponent pro spuštění systému. Na konci této kapitoly je pak popsáno jádro stavového automatu, na základě kterého celý program funguje.

Kapitola 2 se věnuje rozboru slabých míst původního 2D vibrometru, zde se zaměřuji především na softwarové vylepšení. Jednotlivé podkapitoly obsahují popis toho, jak se nedostatek nebo nová vlastnost projevuje a návrh realizace.

Další kapitola 3 se věnuje realizaci některých částí, které jsou uvedeny v předchozí části. V zadání nebyly specifikované konkrétní úkoly pro vylepšení programu. Postupně během analýzy a realizace vznikaly různé nápady pro zlepšení stávajícího řešení. V rámci práce pak byly realizovány ty nejdůležitější. Jednotlivé podkapitoly obsahují základní seznámení s problematikou. Řešení je zde popsáno často stručně.

Předposlední kapitola 4 popisuje kontrolní měření. Jednalo se o měření vibrací nosníku. Cílem tohoto měření bylo ověřit funkčnost provedených změn a prokázat, že nebylo poškozeno původní řešení. V rámci tohoto úkonu došlo i na kontrolu dalších komponent, nikoliv pouze kontrola naměřených dat. Data pak byla vyhodnocena v nově vytvořeném Prohlížecím režimu.

Poslední kapitola 5 obsahuje body, které mohou být předmětem dalšího rozvoje. Jedná se spíše o podněty pro další bakalářské a diplomové práce. V této kapitole je navržena možnost, jak by se vylepšení měla chovat nebo vypadat. Dále obsahuje pár prvních kroků, které jsou potřebné pro realizaci. A naopak kde se mohou objevit potíže.

Práce navazovala na mou semestrální práci [3].

1 Popis systému

Tato práce navazuje na práci Ing. Tomka [1]. Jeho práce byla napsána v roce 2016, avšak od té doby na ní byl prováděn další vývoj. Chybí tak zdokumentování současného stavu přípravku, tedy doplnění o části, které původní práce neobsahuje.

Popis systému lze rozdělit na dvě hlavní části - na hardwarové a softwarové vybavení. Jednotlivé podkapitoly se zabývají výčtem částí a jejich principem. Zapojení vibrometru zobrazuje obr. 1.1.

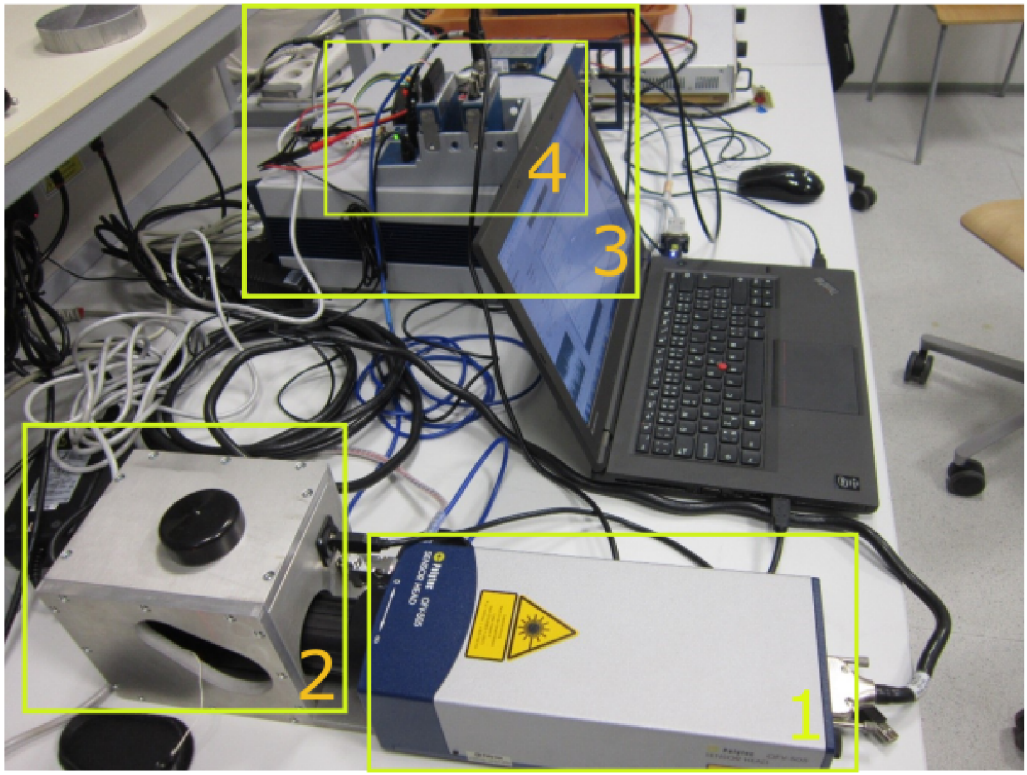
Na tomto obrázku jsou vyznačené jednotlivé hardwarové části. Software běží na zobrazeném notebooku. Oblast číslo 1 je laserová hlava Polytec OFV-505 [4]. Ta emituje paprsek do deflektorové hlavy (oblast 2). Zde jsou umístěny dvě zrcátka GVS012 od firmy Thorlabs [5] pro odraz paprsku na měřený objekt a dále webové kamera Logitech HD Pro Webcam C920 pro optické sesouhlasení a nastavení měřicích bodů. V oblasti 3 je kontrolér Polytec OFV-5000 [6], který slouží jako rozhraní k ovládání a čtení dat z hlavy interferometru. V poslední oblasti 4 je šasi NI cDAQ™-9174 a v něm měřicí karty NI 9263 a NI 9234. V původní práci byla místo karty NI 9234 použita karta NI 9223 [7]. Poslední zmíněné komponenty slouží ke sběru dat z kontroléru, případných referenčních akcelerometrů, přímou komunikaci s kontrolérem a ovládání motorků zrcadel. Výstup ze zmíněného šasi pak vstupuje do PC, kde běží program v prostředí LabVIEW. Celý systém je řízen z PC programem v prostředí LabVIEW, které je napojeno pomocí USB na NI cDAQ™-9174, kontrolér Polytec OFV-5000 a webovou kameru.

1.1 Hardwarové vybavení

V této podkapitole se budeme zaměřovat na hardwarové vybavení, které je součástí vibrometru a měřicího setu. Je zde uveden výčet jednotlivých komponent, včetně zmínění základních parametrů a jejich vlastností. Do hardwarové výbavy lze jistě řaditi PC, avšak konfigurace PC není klíčová a není tak zde uvedena. Dnešní běžné počítače poskytují zcela postačují výkon pro běh aplikace a standardní výbavou jsou i USB porty pro komunikaci s měřicími periferiemi.

1.1.1 Součást setu

Pro lepší ilustraci a zároveň jako podklad pro zprovoznění přípravku slouží tato podkapitola. Je výčtem hardwarových komponent, kterým se věnují další jednotlivé podkapitoly.



Obr. 1.1: Ilustrační zapojení pracoviště - 1) laserová hlava Polytec OFV-5005, 2) deflektorové hlavy, 3) kontrolér Polytec OFV-5000, 4) NI cDAQ™-9174 včetně karet NI 9263 a NI 9234

Seznam hardwarových komponent:

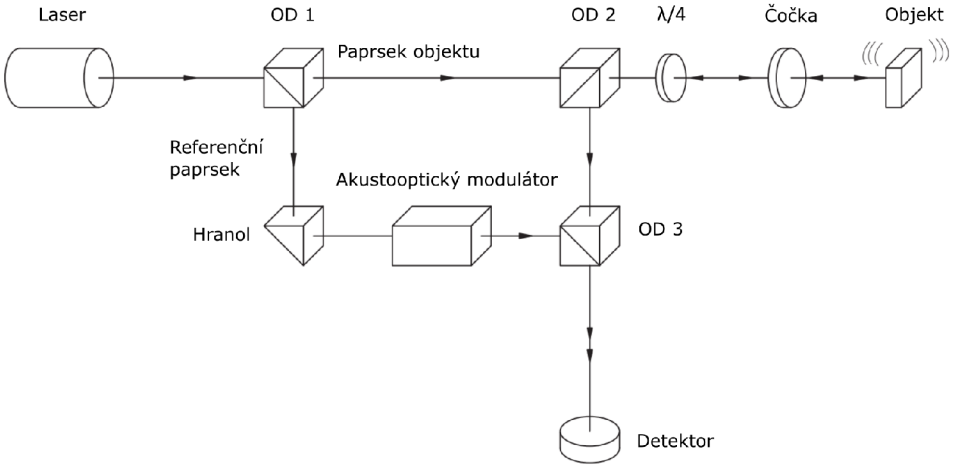
- PC (alespoň 3 USB) s Windows 10 a LabVIEW 2020,
- kontrolér Polytec OFV-5000,
- laserová hlava Polytec OFV-505,
- webové kamera Logitech HD Pro Webcam C920 (součást deflektorové hlavy),
- Rozmítací systém GVS012 (součást deflektorové hlavy),
- zdroj PCM50UD08,
- šasi NI cDAQ™-9174,
- měřicí karta NI 9234,
- měřicí karta NI 9263,
- převodník RS-232 na USB [1],
- propojovací BNC kabely [2],
- kabel pro RS-232 (samice-samice).

[1] slouží k připojení kontroléru k PC přes USB

[2] slouží k připojení výstupu z kontroléru na vstup měřicí karty, připojení referenční akcelerometrů

1.1.2 Laserová hlava Polytec OFV-505

Pro měření vibrací objektu slouží laserová hlava OFV-505 od firmy Polytec. Princip měření vibrací je zobrazen na obr. 1.2. Na začátku měřicího řetězce se nachází laser. Výrobce použil záření vycházející ze směsi HeNe. Laser emituje červené světlo ve viditelném spektru s vlnovou délkou 633 nm. Paprsek tedy putuje ze zdroje (laseru) přes optické děliče (OD) a čočku na měřený objekt. Zde se odráží od stěny objektu a dochází k Dopplerovu jevu. Odražený signál putuje zpět na optický dělič, kde je pak směrován na detektor. Ten vyhodnocuje rozdíl fází a frekvence mezi referenčním a odraženým signálem.

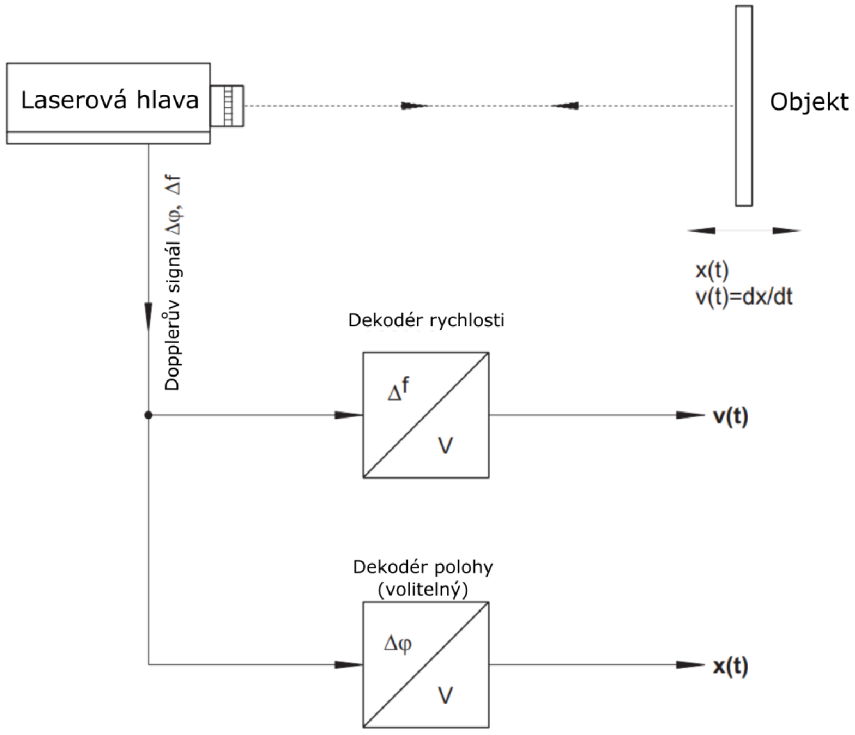


Obr. 1.2: Vnitřní princip měření laserové hlavy OFV-505, převzato a upraveno z [4]

Detektor laserové hlavy je tedy schopen detekovat referenční a odražený signál. Ten však sám o sobě není použitelný. Jak lze vidět na obr. 1.3, výstup z hlavy je rozdíl fáze a frekvence u referenčního a odraženého paprsku. Tyto informace jsou však ukryty v jednom výstupním modulovaném signálu. Tento signál pak putuje ke zpracování do kontroléru OFV-5000, kde je vyhodnocen, viz podkapitola 1.1.3. Výstup z kontroléru je rychlost $v(t)$ a poloha $x(t)$ kmitání měřeného objektu.

Princip měření byl zmíněn výše. Avšak nebyly zmíněny hlavní vlivy, které výsledek měření degradují. Jedním z nich je správné zaostření laserového paprsku. Pokud je paprsek špatně zaostřený, odstup mezi signálem a šumem je malý. To vede zašumění měřených dat. U použité hlavy OFV-505 je možné nastavený fokus měnit a to těmito způsoby:

- ručně na fokusačním kroužku hlavy,
- ručně skrze kontrolér OFV-5000,
- vzdáleně skrze odeslání příkazů na kontrolér OFV-5000 (RS-232).



Obr. 1.3: Ilustrace měření vibrací pomocí laserové hlavy OFV-505, převzato a upraveno z [4]

Dalším způsobem, kterým můžeme zlepšit kvalitu měřených dat, je nastavení vhodné vzdálenosti od snímací hlavy k měřenému objektu. Jak princip měření napovídá, dochází zde k interferenci dvou signálů. Ten má tak svá minima a maxima. Je pochopitelné, že při maximech bude větší odstup signálu od šumu. Snímač dokáže data měřit i při interferenčních minimech, avšak tato data budou více zašuměná. Výrobce uvádí v technickém listu vzorec pro zvolení vhodné vzdálenosti, viz rovnice 1.1. Tedy vzdálenost, při které bude odražený signál nejsilnější. Vzdálenost od fokusačního kroužku hlavy k měřenému objektu je d. Dále je zde uvedená konstanta výrobce 234 mm, l je také konstanta vycházející z parametrů optické dutiny (konstanta výrobce) a násobky n $\{0, 1, 2...\}$. Například chceme měřit objekt ve vzdálenosti přibližně jeden metr. K výpočtu využijeme vzorce 1.1. Výpočet je dle rovnice 1.2. Optimální vzdálenost d je tedy 1050 mm. [4]

$$d = 234mm + (n \cdot l)\ [mm] \tag{1.1}$$

$$d = 234 + (4 \cdot 204) = 1050\ [mm] \tag{1.2}$$

1.1.3 Kontrolér Polytec OFV-5000

V podkapitole 1.1.2 byla zmíněná laserová hlava (viz výše), která se používá pro snímání vibrací. Tato data je však třeba zpracovat. Za tímto účele je použit kontrolér OFV-5000 od firmy Polytec. Ten obsahuje převodníky pro ukazatele rychlosti a výchylky měřeného objektu. Dostupná verze kontroléru obsahuje tyto převodníky:

- dekodér rychlosti 1: VD-02,
- dekodér rychlosti 2: VD-06,
- dekodér výchylky 1: DD-300,
- dekodér výchylky 2: DD-500,
- výstupní filtr: LF-01.

Cest pro vyhodnocení dat z kontroléru je více. Kontrolér umožňuje výstup pomocí digitálního audio rozhraní kompatibilního s S/P-DIF. Dále pak pomocí elektrického nebo optického výstupu. Další možností je výstupní signálový dekodér, který je umístěný v obsluhovaném PC (měřicí karta). Pro účel této práce je nevhodnější možnost komunikace PC s kontrolérem skrze rozhraní RS-232. Skrze poslední zmíněné rozhraní je možné s kontrolérem obousměrně komunikovat. Toho se v programu využívá například u ostření laserového paprsku. Samotná data pak mohou být zpracována a interpretována přes software výrobce Polytec Vibrometer Software. V této práci se o interpretaci a zpracování dat stará program postavený na práci Ing. Tomka [1].

Výše bylo uvedeno, že je možné skrze kontrolér ovládat i samotnou laserovou hlavu. Byla zmíněna možnost ovládání skrze počítač (RS-232). Samotný kontrolér však obsahuje i ovládací obrazovku, skrze kterou lze všechny povely provádět také. Není tak nezbytné používat PC pro konfiguraci laserové hlavy. Pokud však jde o vyhodnocování dat, je PC nebo jiné výpočetní zařízení nevyhnutelné.

Tabulky 1.1, 1.2 a 1.3 obsahují přehled parametrů jednotlivých dekodérů. Z parametrů je patrné, že pro měření rychlosti při nižších kmitočtech je vhodnější VD-06. Dekodér výchylky DD-500 je vhodný pro měření s kmitočtem od 0 do 350 kHz. Pro vyšší kmitočty je vhodnější DD-300. Ten pracuje ve frekvenčním rozsahu od 30 kHz do 24 MHz. Podrobnější specifikace obsahuje dokumentace výrobce.[6]

Tab. 1.1: Přehled parametrů dekodéru VD-06

Rozsah [$\mathrm{mm \cdot s^{-1} \cdot V^{-1}}$]	1	2	10	50
Plný rozsah[$\mathrm{m \cdot s^{-1}}$]	0,01	0,02	0,1	0,5
f_{min} [Hz]	0	0	0	0
f_{max} [kHz]	20	350	350	350
Maximální zrychlení [g]	128	4 500	22 000	110 000
Rozlišení [$\mathrm{\mu m \cdot s^{-1} \cdot \sqrt{Hz}^{-1}}$] běžné	<0,02 0,01	0,01...1 0,05	0,01...0,1 0,05	0,04...0,2 0,06

Tab. 1.2: Přehled parametrů dekodéru VD-02

Rozsah [$\mathrm{mm \cdot s^{-1} \cdot V^{-1}}$]	5	25	125	1 000
Plný rozsah[$\mathrm{m \cdot s^{-1}}$]	0,05	0,25	1,25	10
f_{min} [Hz]	0,5	0,5	0,5	0,5
f_{max} [kHz]	250	1 500	1 500	1 500
Maximální zrychlení [g]	8 000	240 000	1 200 000	9 600 000
Rozlišení [$\mathrm{\mu m \cdot s^{-1} \cdot \sqrt{Hz}^{-1}}$] běžné	0,05...0,2 0,1	0,1...1 0,5	0,3...3 0,6	2...5 2,5

1.1.4 Měřicí karty

Pro manipulaci s laserovým paprskem je potřeba použití zrcadel. Aby byl systém co nejvíce automatický, jsou tato sklíčka ovládána skrze akční člen. Tento akční člen je však ovládán analogově. Řešení bylo zvoleno skrze vývojové prostředí LabVIEW. Byly použity vstupně-výstupní karty společnosti National Instruments. Hlavní výhodou je softwarová podpora a garantovaná kompatibilita s LabVIEW.

Jak bylo zmíněno výše v podkapitole 1.1.3 data z měřicí laserové hlavy jsou zpracována v kontroléru. Výstupní signál z kontroléru, který obsahuje informaci o rychlosti nebo výchylce měřeného objektu, je potřeba nahrát do PC. K tomuto účelu je použita měřicí karta NI 9234. Karta obsahuje 4 kanály. Každý kanál obsahuje 16-bitové A/D převodníky s vzorkovací frekvencí 51,2 tisíc vzorků za sekundu [8]. K těmto kanálům se pak připojuje výstupní signál z kontroléru, referenčního akcelerometru a v budoucnu také další signály.

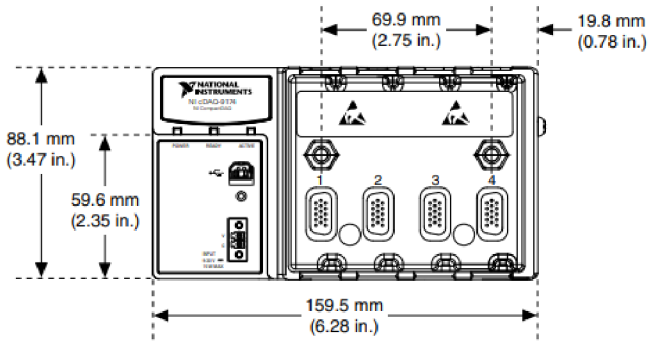
Ovládání zrcadel Thorlabs GVS012 je prováděno skrze napěťový signál. K tomuto účel je použita karta NI 9263. Karta obsahuje 4 kanály. Výstupní napětí je ± 10 V, vzorkovací frekvence je 100 tisíc vzorků za sekundu s rozlišením 16 bitů [9].

Tab. 1.3: Přehled parametrů dekodéru DD-500

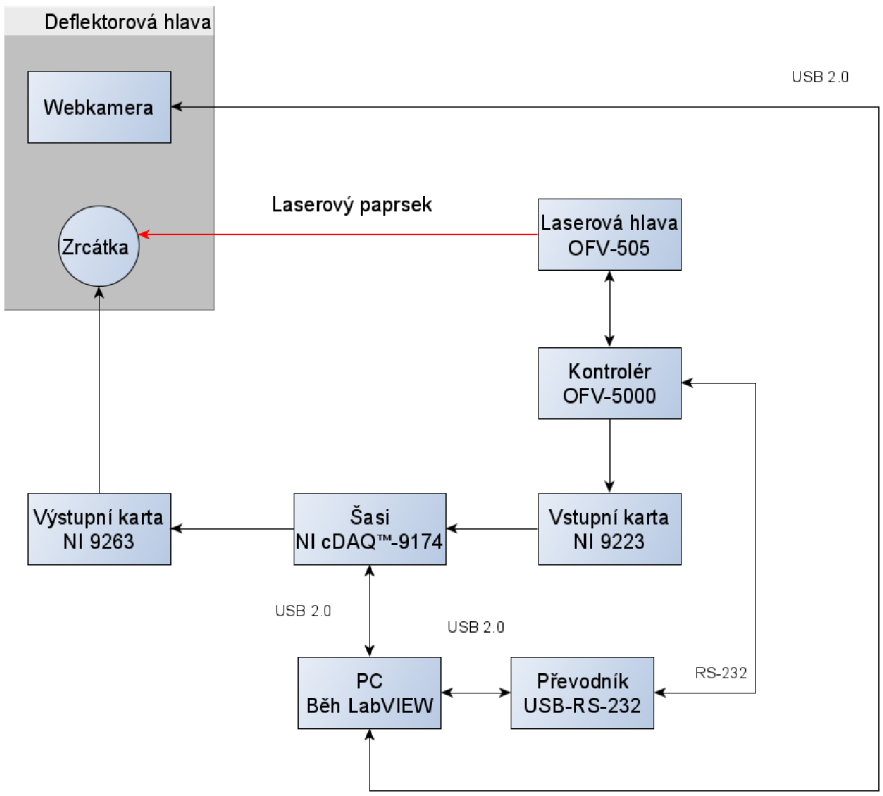
Měřicí rozsah [µm·V^{-1}]	Plný rozsah[µm]	Rozlišení [nm]
0,05	1,00	0,02
0,10	2,00	0,03
0,20	4,00	0,06
0,50	10,00	0,15
1,00	20,00	0,30
2,00	40,00	0,60
5,00	100,00	1,50
10,00	200,00	3,00
20,00	400,00	6,00
50,00	1 000,00	15,00
100,00	2 000,00	30,00
200,00	4 000,00	60,00
500,00	10 000,00	150,00
1 000,00	20 000,00	300,00
2 000,00	40 000,00	600,00
5 000,00	100 000,00	1 500,00

Pro ovládání zrcátek jsou využity dva kanály.

Tyto karty nekomunikují s počítačem přímo. Je zapotřebí šasi, které má rozhraní vhodné pro komunikaci s PC. Většinou se jedná o rozhraní USB. Na PC pak běží řídicí aplikace, která data z karet sbírá, zpracovává a zapisuje. V rámci této práce bylo zvoleno šasi NI cDAQ™-9174 [10]. To podporuje běh až 4 karet NI. Jako rozhraní s PC používá konkrétně USB 2.0 Hi-Speed. Ta má maximální teoretickou rychlost 480 Mbps. Šasi lze vidět na obr. 1.4. Pro názornost jsou všechny zmíněné komponenty a jejich logické spojení zobrazeny na obr. 1.5.



Obr. 1.4: NI cDAQ™-9174 [10]



Obr. 1.5: Přehledové schéma zapojení měřicích karet National Instruments (NI) a ostatních komponent

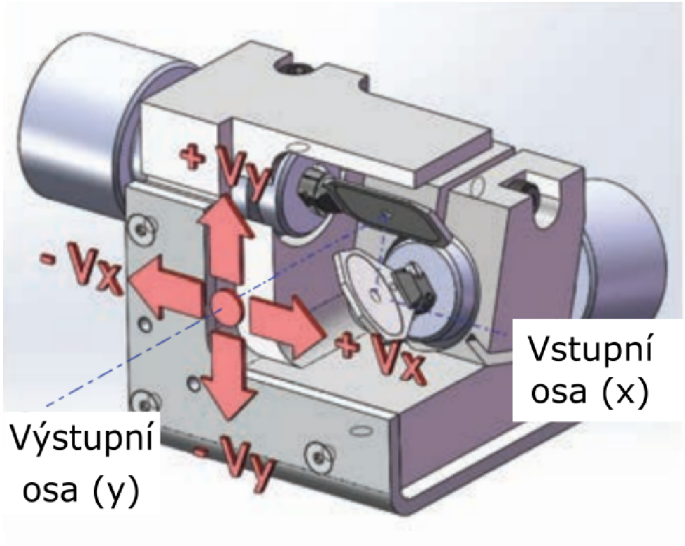
1.1.5 Webkamera Logitech HD Pro Webcam C920

Pro snadnější vytváření měřených bodů a optické sesouhlasení je třeba kamery. Zde je použita webová kamera Logitech HD Pro Webcam C920. Jejím výběrem se zabýval ve své diplomové práci Ing. Tomek [1]. Ve své práci uvádí, že kamera má snímací úhly 55° pro horizontální osu a 42° pro osu vertikální. Uvádí zde, že tento údaj získal experimentálně. Webové stránky výrobce uvádí snímací úhly 70,42° pro horizontální osu a 43,3° pro osu vertikální. [11]. Nejsem si však vědom, že by se přímo s těmito parametry počítalo a jejich nepřesnost měla vliv na výsledné měření.

Dále je nutné doplnit základní parametry, které uvádí výrobce. Jako rozhraní je zde použito USB 2.0. Kamera má podporu Full HD (1080p@30FPS) a rozlišení 15 MPx. Podporuje autofokus, tedy automatické zaostření. Výrobce uvádí, že je schopen zaostřit ve 20 krocích. Podmínkou je, aby pozorovaný předmět byl minimálně 10 cm vzdálený od objektivu. Webová kamera dále obsahuje stereo mikrofon. Ten však není nijak využit [12].

1.1.6 Rozmítací systém Thorlabs GVS012

Již bylo naznačeno, že budeme chtít měřit bezkontaktně vibrace na vzdáleném objektu. Většinou však při měření vibrací neměříme jeden bod, ale měříme bodů více. Manipulace se samotnou laserovou hlavou vibrometru je nevhodná. Proto ve své práci Ing. Tomek [1] zvolil vhodnější řešení. Tímto řešením je rozmítací systém Thorlabs GVS012. Zmíněný systém se skládá ze dvou zrcátek a serv včetně jejich napájení. Výrobce u tohoto modelu uvádí vhodnost pro vlnové délky 500 nm až 2 µm, tedy tento parametr je splněn. Pro zajímavost uvádím, že plocha zrcadel je pokryta stříbrem. Na obr. 1.6 je zobrazen samotný systém. Zrcátka jsou natáčena přivedením napětí na napájení serv, obrácením polarity napětí se pak mění směr otáčení. Směr otáčení a polarita jsou naznačena šipkami, přičemž +Vx a −Vx jsou směry otáčení v ose x a obdobně pro osu y +Vy a −Vy. Původní napájení od výrobce bylo nahrazeno zdrojem PCM50UD08 [5].

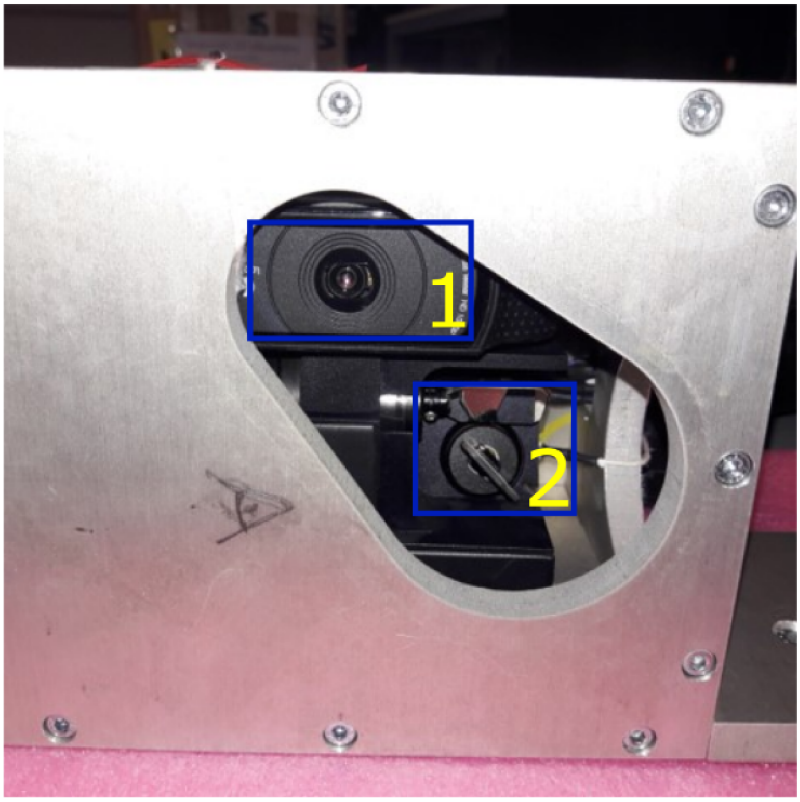


Obr. 1.6: Rozmítací systém Thorlabs GVS012 a naznačení os, převzato a upraveno z [5]

1.1.7 Deflektorová hlava

V předchozích podkapitolách byla zmíněna kamera Webcam C920 (viz podkapitola 1.1.5) a rozmítací systém Thorlabs GVS012 (viz podkapitola 1.1.6). Tyto části je vhodné spojit do kompaktního provedení, jež bylo vyrobeno Ing. Tomkem. Toto provedení je zobrazeno na obr. 1.7, kde laserový paprsek vchází z pravého otvoru a odráží se od zrcadel na snímaný objekt (viz oblast č. 2). Pro obsluhu a kalibraci je

uvnitř umístěna zmíněná kamera. Ta se pak dívá[3] ve směru odraženého paprsku na měřený objekt (viz oblast č. 1). Díky ní může obsluha programu přesně vědět, kde se daný paprsek vyskytuje, a provést měření nebo kalibraci.



Obr. 1.7: Boční pohled na deflektorovou hlavu

1.2 Softwarové vybavení

V podkapitole 1.1 je zmíněna problematika hardwarového vybavení. Pro kompletnost je třeba se vyjádřit k hlavní části celého systému - softwaru. Program byl vytvářen v prostředí LabVIEW. Jelikož jednotlivé verze LabVIEW mají zpětnou kompatibilitu. Byla zvolena pro práci s programem nejnovější verze 2020 a v ní byl projekt dále rozvíjen.

Program je převzat z původní práce. Jedná se však o podklad, který je rozšířen o jednotlivé funkcionality zmíněné v dalších podkapitolách. Nutno podotknout, že se nejedná o stejnou verzi programu, která je publikována v původní práci. Publikovanou verzi používali zaměstnanci Ústavu automatizace a měřicí techniky. Během používání si však některé části modifikovali, jako příklad uvedu problém se špatnou inicializaci rozsahu u měřicí karty v původní práci. Jelikož jsem měl k dispozici již

[3]obraz z kamery je otočen o 180°, v programu je provedena korekce

tuto modifikovanou (ne publikovanou) verzi, stavěl jsem svou práci na ní. Základní koncept stále vychází z publikované verze a jedná se spíše o implementační změny kódu.

Během oživování kódu jsem musel nainstalovat všechny povinné dostupné softwarové komponenty. Níže je uveden seznam těchto nezbytných softwarových komponent. Původní práce seznam těchto komponent neobsahovala.

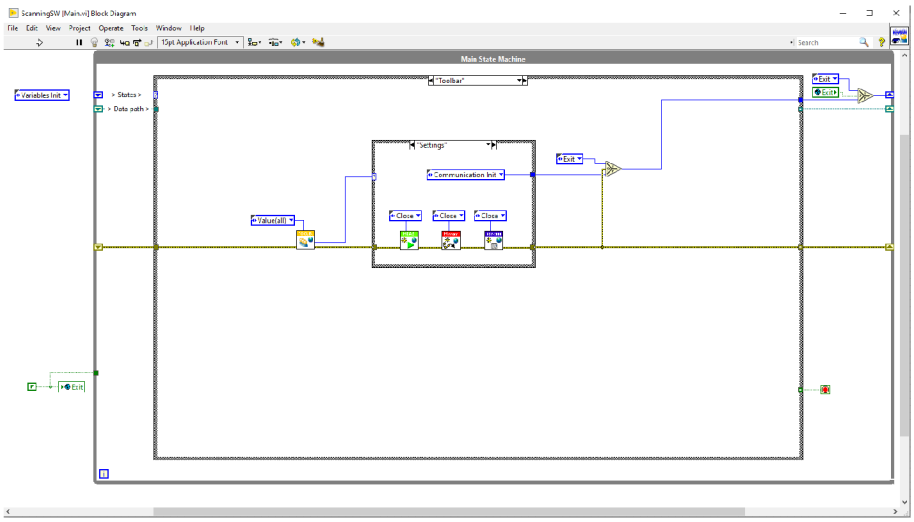
Potřebné softwarové komponenty:

- operační systém Windows 10 (64-bit s českým jazykovým balíčkem),
- LabVIEW 2020,
- NI Sound and Vibration 2020,
- Vision Development Module 2020,
- MathScript Module,
- Vision Acquisition Software 20.0 (IMAQdx),
- OpenG toolkit (dostupný skrze VI Packet Manager).

1.2.1 LabVIEW

LabVIEW neboli Laboratory Virtual Instrument Engineering Workbench je programové vývojové prostředí vyvinuté společností National Instruments v roce 1986. Je používáno zejména při měřicích a experimentálních úlohách. Má proto mnoho dobrých vlastností oproti jiným vývojovým prostředí, zejména velkou podporou SW knihoven a ovladačů (driverů), kompatibilitu s měřicími kartami NI a grafický programovací jazyk G. Ten ulehčuje vývoj, jelikož algoritmus je čitelnější na první pohled. Nutno však podotknout, že při větších projektech bez vhodného využití SubVI je program na první pohled často stejně nepřehledný jako v jiných jazycích. SubVI by se dal popsat jako podprogram, jedná se o „zabalení“ libovolné části kódu do jednoho bloku, který je pak zobrazen namísto původního algoritmu. Například na obr. 1.8 jsou SubVI vizualizovány jako barevné čtverce. Další výhodou vizuálního programovacího jazyku je jasné značení toku dat, tedy jak se data postupně zpracovávají, a zcela intuitivní programování paralelního běhu programu. Výhodou je možnost generování samostatného exe souboru, tedy kompilaci programu, která je samostatně spustitelná na operačním systému Windows.

LabVIEW má však i své nevýhody. Pro opravdu rozsáhlé projekty může být samotný grafický jazyk až nepoužitelný, jelikož textová forma může být čitelnější a hlavně vhodnější pro kooperaci (týmový vývoj). Jak lze předpokládat, jelikož LabVIEW má podporu téměř „všeho“ má i problém s optimalizací. Běh samotné aplikace může být pomalejší než nativně naprogramovaná aplikace například v programovacím jazyce C++. Je však nutno zvážit schopnosti programátora, zda je schopen



Obr. 1.8: LabVIEW s grafickým programovacím jazykem G

napsat kód více optimalizovaný než kód zkompilovaný LabVIEW. Za nevýhodu lze považovat i celkovou svázanost vývoje s ekosystémem LabVIEW, respektive National Instruments. Nelze jednoduše vzít kód naprogramovaný v LabVIEW a zkompilovat ho v jiném vývojovém prostředí, jako je to třeba možné u zmíněného C++ [13].

1.3 Popis programu

Hlavním cílem této práce je vylepšení stávajícího řešení vyvinuté Ing. Tomkem. Pro posouzení optimalizace a vylepšení je potřeba pochopit dosavadní program. Tato podkapitola si klade za cíl seznámit čtenáře se základními rysy. Popsat celý program se všemi aspekty je nevhodné, jelikož se jedná o opravdu rozsáhlý a komplexní projekt. Vhodnější variantou, dle mého názoru, je představit základní koncepci. Implementačním detailům se pak budou věnovat dílčí podkapitoly, pro které je tato znalost potřebná [1].

1.3.1 Základní přehled programu

Jak bylo výše zmíněno, tento program byl programován v prostředí LabVIEW. Program je rozdělen na jednotlivé programové bloky (SubVI). Jedná se prakticky o podprogram, celkový koncept je tedy navrhován modulárně. Nutno však podotknout, že moduly na sebe úzce navazují a tak má tento model svá omezení při dalším rozšiřování, například o podporu jiných měřicích hlav nebo kontroléru.

Programování je zde prováděno graficky, proto se zde zcela nabízí použití stavového automatu. Ten má 16 stavů a je uložen v souboru *Main.vi*. Na obr. 1.9 jsou zobrazeny vazby mezi jednotlivými stavy. Samotné měření probíhá ve stavu *Main*, ostatní stavy slouží k postupné konfiguraci měření. Například inicializace vstupních výstupní periférií, inicializace proměnných a ukládání.

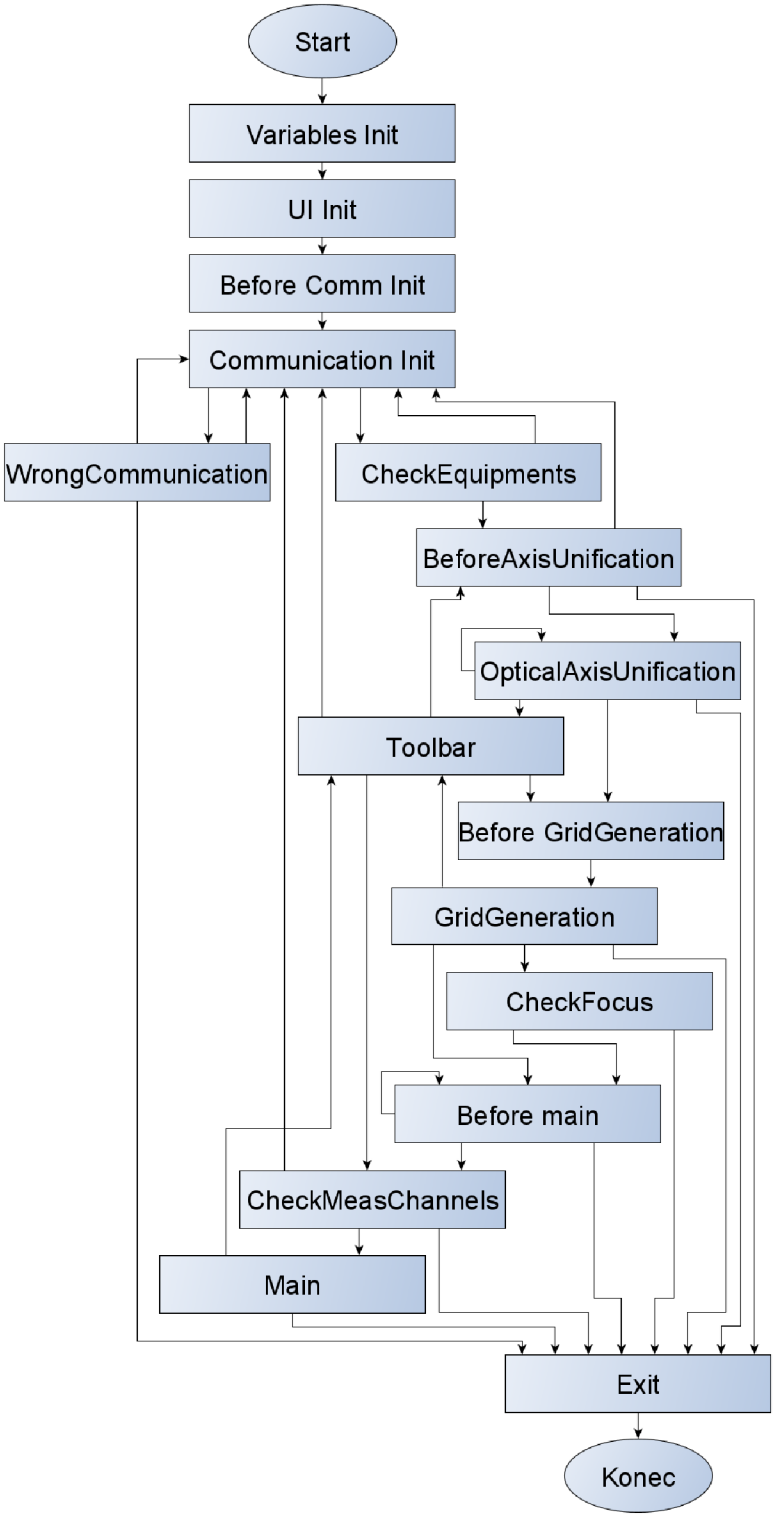
Pro lepší ilustraci vazeb mezi důležitými stavy jsou k dispozici následující obrázky. Na obr. 1.10 jsou znázorněny vazby na stav *Communication Init*, v němž probíhá inicializace vstupních a výstupních periférií. Na obr. 1.11 jsou obdobně zobrazeny vazby na stav *Main*, kde probíhá měření. Poslední obr. 1.12 také zobrazuje jednotlivé vazby, tentokrát stavu *Toolbar*, který slouží pro změnu nastavení.

1.3.2 Stav Variables Init

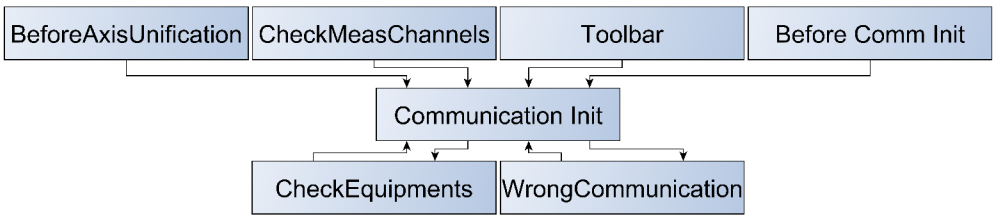
Při startu programu jde program do stavu *Variables Init*. V tomto stavu se jedná zejména o inicializaci proměnných, které jsou používány na pozadí, včetně nastavení zobrazovaného okna v Control Tabu na *Axis*.

1.3.3 Stav UI Init

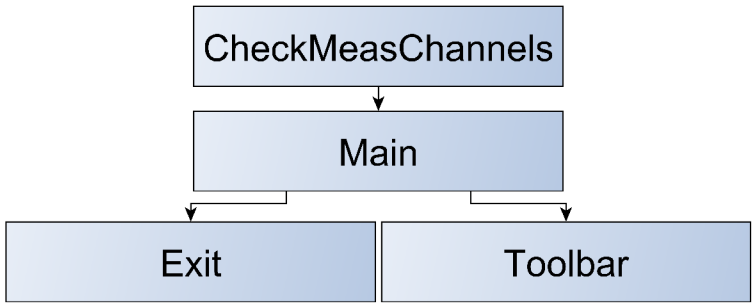
Po předchozím stavu *Variables Init* přechází program do stavu *UI Init*. Jak název napovídá, v tomto režimu probíhá inicializace proměnných sloužící primárně pro uživatelské rozhraní.



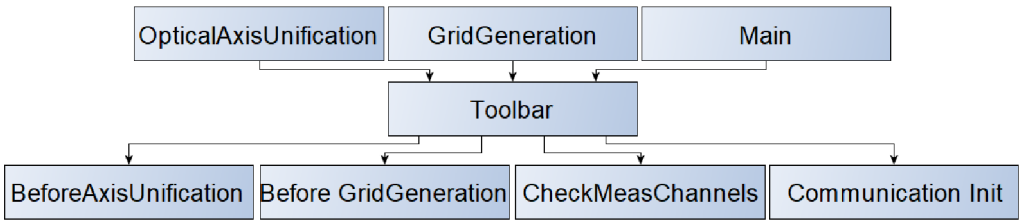
Obr. 1.9: Přehledové schéma stavového automatu programu



Obr. 1.10: Detail na relace stavu Communication Init



Obr. 1.11: Detail na relace stavu Main



Obr. 1.12: Detail na relace stavu Toolbar

1.3.4 Stav Before Comm Init

Ve stavu *Before Comm Init* dochází k resetování chybových stavů. Pak program přechází do stavu *Communication Init*.

1.3.5 Stav Communication Init

Ve stavu *Communication Init* je na rozdíl od příchozích stavů více činností. Na začátku se načtou data z konfiguračních souborů *Settings.ini* a *ControllerSettings.ini*. To je z důvodu, aby uživatel měl již nějakou předdefinovanou volbu. Dále pak pokračuje inicializace vstupních a výstupních karet, včetně inicializace komunikace s kontrolérem. Pokud došlo k chybě, pokračuje se stavem *WrongCommunication*, pokud nedošlo, je dalším stavem *CheckEquipment*.

1.3.6 Stav WrongCommunication

Ve stavu *WrongCommunication* dochází ke komunikaci s uživatelem, který je informován o chybě, a může program ukončit nebo znovu provést inicializaci.

1.3.7 Stav CheckEquipments

Pokud dříve v pořádku proběhla inicializace, následuje další stav *CheckEquipments*. Zde se kontroluje, zda došlo k inicializaci: zrcátek, vibrometru a měřicích kanálů. Není zde kontrolována dostupnost webkamery.

1.3.8 Stav BeforeAxisUnification

Ve stavu *BeforeAxisUnfication* je na začátku testováno připojení kamery. Pokud není, je uživatel vyzván k ukončení programu nebo provedení inicializace znovu. V případě, že je kamera dostupná se, pak uživatel může rozhodnout, zda provede sesouhlasení os nebo použije dřívější hodnoty. Na základě rozhodnutí pak program pokračuje do stavu *OpticalAxisUnification* (sesouhlasení os) nebo *Before GridGeneration* (příprava pro generování měřicí mřížky). Program dále změní zobrazované okno na *Axis*.

1.3.9 Stav OpticalAxisUnification

Stav *OpticalAxisUnification* má za úkol sesouhlasit obraz z webkamery s fyzickou pozicí. Děje se tak na základě čtyř zadaných bodů. Uživatel zadá v obraze programu bod, a pak pomocí šipek na obrazovce koriguje laser, aby paprsek byl na stejném místě jako je zvolený bod. O tuto úlohu a další nutné přepočty se zde stará *AxisUnificationMain.vi*. Pokud sesouhlasení proběhlo v pořádku, jsou parametry uloženy a pokračuje se stavem *Before GridGeneration*. V jiných připadech je program ukončen nebo upozorňuje uživatele na špatně zadaná vstupní data.

1.3.10 Stav Before GridGeneration

Ve stavu *Before GridGeneration* se resetují poruchová hlášení a aktivuje se okno *Grid*. Poté se přechází na stav *GridGeneration*.

1.3.11 Stav GridGeneration

Ve stavu *GridGeneration* má uživatel možnost vybrat si body, které budou měřeny. K dispozici jsou následující tvary: obdélník, čtverec a elipsa. Pak může nastavit až 2 000 bodů chce v tomto útvaru vygenerovat mřížku. Generování bodů a zpracování

pak probíhá v *GridGenerationMain.vi*. Pokud uživatel neukončil program nebo nenastala chyba, pokračuje se do další přípravné fáze. Pokud je přítomný vibrometr, přechází se do stavu *CheckFocus*. Pokud není, přechází se přímo na stav *Before main*.

1.3.12 Stav CheckFocus

Ve stavu *CheckFocus* probíhá kontrola zaostření jednotlivých dříve zvolených bodů. Je zde proveden i odhad času, který je potřeba pro ostření všech bodů. Fokusace bodů je volitelná a uživatel tento krok může přeskočit. Pokud se na konci tohoto stavu vyskytne chyba, program je ukončen. Pokud ne, program pokračuje do stavu *Before main*.

1.3.13 Stav BeforeMain

Data z měření je třeba ukládat pro účely dalšího vyhodnocení. Ve stavu *BeforeMain* je uživatel dotázán, kam se data mají ukládat. Jsou pak prováděny nezbytné úkony pro ukládání souborů, například vytvoření souboru, jeho předvyplnění a jiné.

1.3.14 Stav CheckMeasChannels

Ve stavu *CheckMeasChannels* dochází znovu ke kontrole připojených periférií. Pokud jsou zrcátka, vibrometr i měřicí karty připojeny, přechází program do stavu *Main*, kde dochází k měření vibrací u vybraných bodů.

1.3.15 Stav Main

V předchozí stavech probíhala konfigurace a inicializace parametrů nutných pro zahájení měření vibrací. Mezi ně patří například: volba měřicích karet, vzdálenost od měřeného objektu a další parametry. Ve stavu *Main* probíhá už samotné měření na základě dříve zvolených bodů. Všechny tyto operace jsou prováděny v SubVI *MeasurementMain.vi*. Z tohoto stavu je pak možné buď přejít do stavu *Toolbar* a změnit nastavení, nebo ukončit program (stav *Exit*).

1.3.16 Stav Toolbar

Ve stavu *Toolbar* je možné se přesouvat mezi stavy. To znamená, že pokud je program již ve stavu měření (*Main*), může uživatel zvolit například sesouhlasení os a nemusí kvůli tomu restartovat program. Z nově vybraného stavu pak pokračuje podle stavového automatu, viz obr. 1.9.

1.3.17 Stav Exit

Ve stavu *Exit* dochází k ukončení programu, tedy k ukončení komunikace s periféri-emi a zastavení programu. Jedná se o poslední stav.

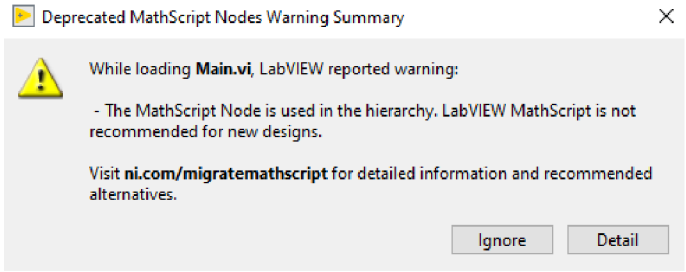
2 Slabá místa stávajícího řešení

Jak název samotné práce napovídá, je třeba analyzovat současný stav řešení. Samotná analýza je komplikovaná, protože pro odhalení určitého typu nedostatků je třeba praxe s přístrojem. Vzhledem k nečekanému vývoji pandemie COVID-19 a zavedení nouzového stavu, bylo řešení praktické části velmi komplikované, jelikož je software velmi těsně propojen s hardwarem. Nebylo možné plnohodnotně pracovat s přístrojem distanční formou. Proto některým úkolům nebyl věnován větší prostor [14], [15].

2.1 Použití MathScript Nodes

Při zapnutí původního programu LabVIEW vždy upozorňuje uživatele o použití MathScript Nodes viz obr. 2.1. Tento modul umožňuje programovat matematické operace textově. U složitějších algoritmů se jedná o jednoznačně komfortnější metodu. Od verze LabVIEW 2018 je možnost použití zmíněného Matlab Script Nodes. Jedná se o následníka MathScript Nodes. Ovšem k jeho zprovoznění je potřeba mít na daném PC nainstalovanou instalaci Matlab 2018a a novější [16].

Kromě sníženého komfortu uživatele při zapínání programu lze v budoucnu očekávat úplné zrušení podpory MathScript Nodes. Proto je potřeba nalézt způsob, jak plnohodnotně nahradit současné provedení výpočtu transformační matice pro sesouhlasení os. Cesty jsou dvě. První je přeprogramovat MathScripty Nodes na Matlab Script Nodes. Nevýhoda je ovšem ve zmíněné potřebě mít nainstalovaný Matlab, a tedy nutnost další licence. Druhou možností je naprogramovat jednotlivé komponenty v jazyce G a vytvořit vhodné SubVI.



Obr. 2.1: Upozornění LabVIEW na použití MathScript nodes

2.2 Neintuitivní uživatelské rozhraní

Při návrhu softwarového vybavení programátoři často dávají velký důraz na vnitřní mechaniky programu. Avšak uživatelskému rozhraní bývá často věnována minimální pozornost. Nutno podotknout, že dobře navržené uživatelské prostředí pomáhá předcházet chybovým stavům, tak i rychlejšímu zaučení obsluhy a díky tomu snižuje náklady na provoz.

Na obr. 2.2 je snímek obrazovky pro sesouhlasení os, který může být považován za nejvíce neintuitivní z celého konceptu programu. Tato obrazovka se uživateli zobrazí poté, co nastaví vstupní a výstupní periférie. Na obrázku jsou označené jednotlivé ovládací prvky.

Pro testovací účely byla před kameru postavena papírová krabice, na kterou svítil paprsek. Na začátku má uživatel možnost si vybrat, jestli použije uložené parametry nebo provede kalibraci. Pokud uživatel provádí kalibraci, musí v prvním kroku kliknout na přidání bodu (oblast 1), poté musí uživatel kliknou na vybrání bodu (oblast 2). Teprve potom je možné manipulovat s laserem pomocí šipek (oblast 8). V oblasti 3 jsou pak zobrazeny souřadnice bodu. S bodem lze také manipulovat skrze tlačítka (oblast 6). Kdykoliv během nastavování je možné měnit zaostření (fokusace) kamery (oblast 5) nebo laserové hlavy (oblast 7). Předposledním krokem je volba jednoho bodu (oblast 4), který se stane referenční pro určení vzdálenosti (na základě zvoleného fokusu). Poslední krok je potvrzení volby pomocí tlačítka v oblasti 9. Z tohoto popisu by se mohlo zdát, že jde o intuitivní řešení. Nyní je však třeba osvětlit slabá místa tohoto konceptu.

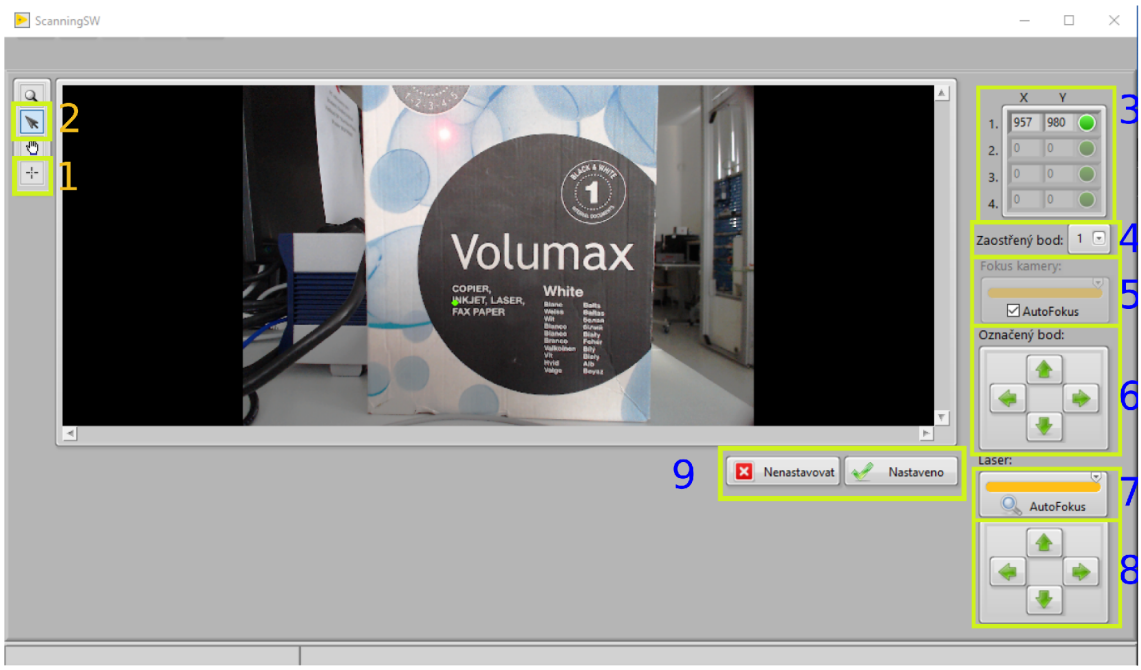
2.2.1 Navigace laseru pomocí tlačítek na obrazovce programu

S laserem není možné pohybovat, dokud není vybrán bod. O tom však uživatel není nijak informován. Pohyb laseru lze provádět pouze pomocí kliknutí a *držením* vybraných šipek na obrazovce. To může být matoucí, protože laser občas odskočí mimo zorné pole kamery. Uživatel pak mačká tlačítka šipek, ale laser se nepohybuje (resp. se pohybuje o příliš malý kousek). Může tak snadno dojít k závěru, že mechanika přístroje nefunguje. Obdobně je to u možnosti manipulace s body.

Řešením tohoto problému je rozšířit kalibrační blok *AxisUnificationMain.vi* o události stisknutí kláves, tedy ovládat paprsek pomocí klávesových šipek na klávesnici. Tím se jednoznačně zlepšuje komfort práce s programem.

2.2.2 Přemístění bodů pomocí myši

U dnešních programů je zvykem ovládat prvky pomocí myši, zejména, pokud se jedná o přesun objektů. Při kalibraci je potřeba pohybovat s referenčními body a vhodně



Obr. 2.2: Snímek úvodní kalibrace sesouhlasení os - 1) přidání bodu, 2) vybrání bodu, 3) souřadnice bodu, 4) referenční bod pro zaostření laseru, 5) fokus obrazu kamery, 6) manipulace s bodem, 7) fokus laseru, 8) manipulace s laserovým paprskem, 9) potvrzení sesouhlasení

je rozmístit. Toto rozmístění však probíhá nikoliv pomocí myši, ale pomocí šipek na obrazovce. Zmíněné provedení není příliš komfortní, a je proto potřeba přepracovat a doprogramovat podporu myši, při zachování zpětné kompatibility,tedy možnosti pohybovat s bodem pomocí šipek. Například pro jemnější korekce.

2.2.3 Inicializace kalibračních bodů

Výše bylo zmíněno, že uživatel při kalibraci musí provést dva hlavní kroky - vložit bod a pak jej vybrat, přičemž pro každý úkon je potřeba použít jiný prvek na obrazovce. Rozdělení tohoto kroku není příliš intuitivní. Vhodnější by bylo tuto tyto kroky eliminovat.

Při vhodném designu je uživatel velmi intuitivně naváděn k různým úkonům. Když se uživatel poprvé setká s programem, netuší, že má vkládat body a až pak s nimi manipulovat. Řešením je tedy uživateli tyto body při inicializaci automaticky vložit. Uživatel tak bude mít vždy vložen správný počet bodů, tedy 4 body, není tak třeba kontrolovat platný počet vložených bodů. Uživatel je zároveň intuitivně naváděn k volbě bodů. Nakonec není potřeba dvou prvků (vložení a volba), stačí pouze prvek pro výběr bodu.

V případě potřeby je možné vkládat více bodů pro sesouhlasení, pak je však nutné upravit: SubVI pro výpočet transformační matice, SubVI pro inicializaci bodů a grafické rozhraní, tak aby zobrazovalo všechny body.

2.2.4 Referenční fokusace bodu

Výše na obrázku 2.2 v podkapitole 2.2 byla zmíněna volba referenčního zaostření paprsku. Správné nastavení tohoto bodu je naprosto klíčové pro korektní fungování programu. Z tohoto údaje je pak dále v programu odhadována vzdálenost. Pokud uživatel zadá špatný údaj, je špatně vypočítána vzdálenost a tato chyba se pak řetězí v programu. Projevuje se například naprostým rozhozením jinak správně provedené kalibrace. Při výpočtu transformační matice a dalších výpočtech jsou nastavené chybné parametry a kalibrace není úspěšná. Nešťastné na této chybě je to, že uživatel o této skutečnosti není nijak informován.

2.2.5 Umístění prvků

Již dříve v podkapitole 2.2 bylo zmíněno ovládání u okna pro sesouhlasení os. Zde byla navržena změna pořadí ikon v sloupci pro oblast 1 a 2, pokud to bude možné. Dále by bylo vhodné doplnit potvrzovací tlačítko, které by bylo uprostřed šipek pro pohyb s laserem. Uživatel by pak kliknul do obrazu a tím umístil bod, potom by uživatel korigoval laser pomocí tlačítek nebo kláves.

V oblasti 7, je k dispozici možnost volit pomocí posuvníku ostření laserového paprsku. Někdy je však obtížné zvolit vhodné zaostření, zvláště, pokud je potřeba změnit hodnotu jenom nepatrně. Z toho důvodu byla vložena hodnota zaostření vedle posuvníku.

2.2.6 Doporučení vhodné vzdálenosti

Pro přesnější měření je vhodné objekt umístit ve vzdálenosti, v níž odražený signál interferuje v maximech. O této problematice pojednává podkapitola 1.1.2. Uživatel, který program obsluhuje, nemusí o této vzdálenosti vůbec vědět, a může proto kvůli tomu docházet k mírně nepřesnějším výsledkům.

Vhodným řešením by bylo uživateli nabídnou nejvhodnější vzdálenost na základě jím zadané vzdálenosti. To znamená, že kdyby uživatel například zadal vzdálenost 1 000 mm, systém by mu doporučil, že nejvhodnější vzdálenost je 1 050 mm. Program by však uživatele nijak neblokoval, pokud by tuto vzdálenost nezadal. Tímto způsobem bude uživatel naváděn k volbě vhodné vzdálenosti objektů. Bude však potřeba změřit vzdálenost od čočky k okraji deflektorové hlavy. Vzdálenost mezi zrcátky je 17 mm [1]. Očekává se, že vzdálenost kterou uživatel zadá, bude měřena

právě od tohoto okraje. Nutno podotknout, že tyto vzdálenosti jsou spíše přibližné. Není potřeba pro měření mít objekt od okraje deflektorové hlavy ve vzdálenosti s přesností na milimetry.

2.3 Problém s opakovaným ukládáním dat

V podkapitole 1.3 bylo popsáno fungování programu. V něm je zejména zmíněný stavový automat, dle kterého celý program funguje (obr. 1.9). Problém však nastává při ukládání dat.

Před samotným měřením je program ve stavu *BeforeMain*, kde je uživatel vyzván k volbě adresáře pro ukládání souborů (naměřených dat). Program pak pokročí do stavu *Main*. V tomto stavu program data změří a uloží. Problém však nastává, pokud je měření opakované. Program totiž používá stejnou cestu k TDMS souboru pro uložení a jsou tak přepsána předchozí naměřená data. Uživatel o této skutečnosti není nijak informován.

2.4 Export snímků měření

V původním programu jsou po ukončení exportována data s naměřenými hodnotami a parametry. Mezi soubory chybí obrázek z webkamery, která snímá měřený objekt/y. Pro správné vyhodnocení dat nestačí data samotná. Je potřeba mít k dispozici záznam o tom, kde byly umístěny měřené body a jak vypadala scéna měření.

2.5 Export dat

Po ukončení měření je potřeba exportovat data týkající se hodnot a nastavených parametrů. V původním programu export dat existuje. Nutno podotknou, že není dostatečně promyšlený. Například soubory se neukládají na jednotné místo. Některé soubory se ukládají do složky „data“, jiné na místo umístění programu nebo dle zvolené cesty uživatele. Data jsou tak roztříštěná ve složce aplikace. Uživatel je pak po ukončení měření musí všechny dohledat a zálohovat. To není uživatelsky příliš přívětivé. Navíc uživatel může snadno některé soubory zapomenout zálohovat, a tím prakticky znehodnotit výsledky měření.

Navíc názvy souborů obsahují generické elementy. Jako je datum a čas vytvoření. V principu je správné mít záznam o datu vytvoření souboru. Avšak tato informace je již obsažena ve vlastnostech souboru a není potřeba zapisovat do názvu souboru. Navíc tím, jak jsou soubory generovány postupně, liší se jejich názvy (časy) a neexistuje žádný klíč, který umožní jednoduše strojově identifikovat soubory jako součást

jednoho měření. Tím se vylučuje strojové zpracování všech souborů „současně“. Uživatel je tak nucen při větším množství měření nejen všechny soubory najít, ale ještě je správně kompletovat.

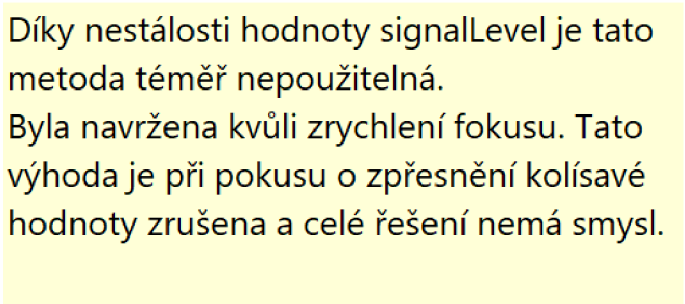
2.6 Prohlížecí režim

Výsledky měření pro jednotlivé body je možné zobrazit bezprostředně po ukončení měření. Není je však možné zobrazit po uzavření programu. K jejich zobrazení je potřeba externí program. Toto lze považovat za nevýhodu, protože není možné rychle zkontrolovat data po ukončení programu samostatně. Bylo by tak vhodné rozšířit původní program o prohlížecí režim, kde by se uživatel mohl podívat alespoň na základní výsledky měření. Aniž by musel procházet celým stavovým automatem měření nebo potřebovat mít připojené periférie.

2.7 Softwarový autofokus

Měření vibrací probíhá bezkontaktně pomocí laseru. Aby byly výsledky měření co nejpřesnější, je potřeba vhodně zaostřit tento paprsek. V podkapitole 1.1.2 jsou zmíněny způsoby, jak provádět fokusaci laserové hlavy. V programu je ostření řešeno pomocí komunikace s kontrolérem Polytec OFV-5000. Program odešle skrze RS-232 povel kontroléru, aby provedl autofokusaci. Problém je však v tom, že měření probíhá v celém rozsahu a tak ostření jednoho bodu trvá několik vteřin. Pokud je měřen jeden bod, není to problém. Pokud však toto ostření probíhá, řekněme u desítek bodů, pak celý proces trvá řádově i minuty.

Nabízí se tedy možnost vytvořit vlastní ostřicí (autofokusační) algoritmus. V programu je část kódu (viz obr. 2.3), která se zabývá softwarovým autofokusem. Je u něj poznámka, že signál byl velmi zašuměný a nestálý.

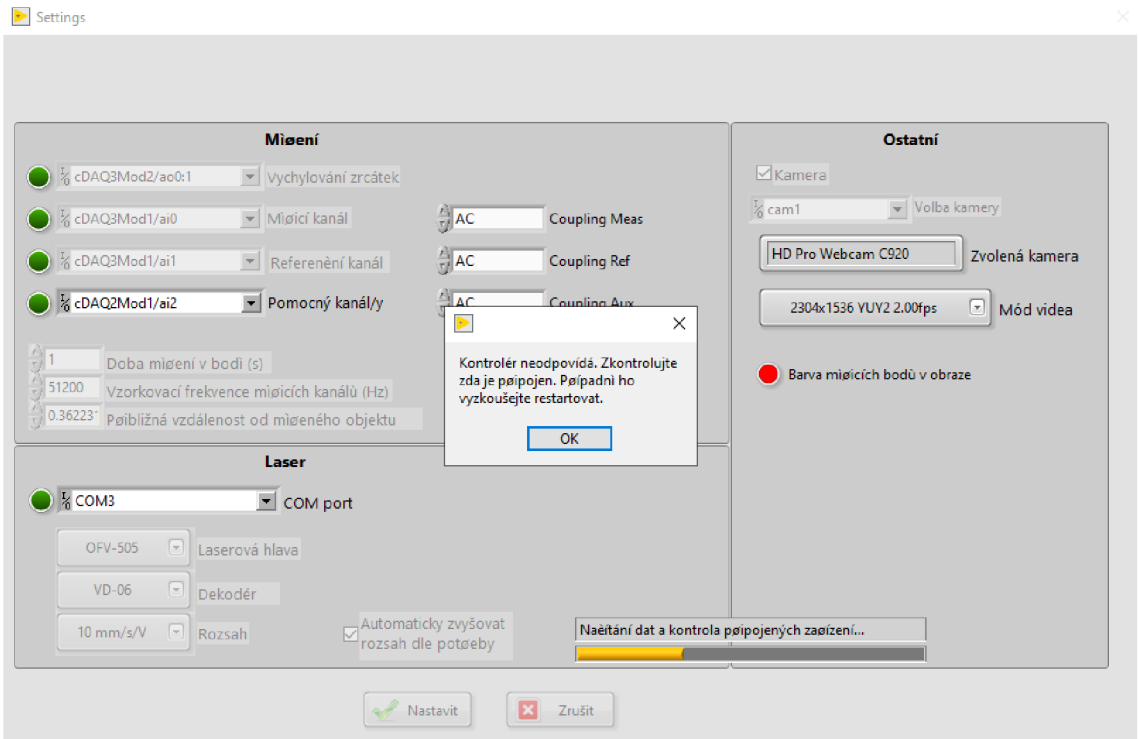


Obr. 2.3: Původní poznámka v SubVI PSV_5000_FGV.vi

2.8 Podpora anglické lokalizace systému

Práce byla vytvořena na české vysoké škole a tak její lokalizace do češtiny je logická. Problém však nastává, pokud je aplikace spuštěna pod cizojazyčnou verzí operačního systému Windows. Níže na obr. 2.4 je vidět příklad zkomoleného textu s českou diakritikou.

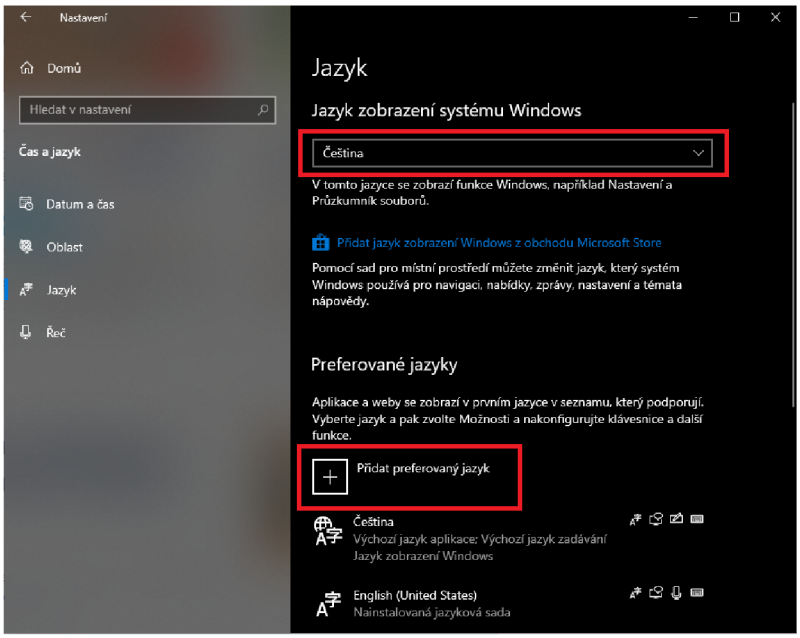
Řešit problém s chybějící diakritikou lze pomocí stažení jazykového balíčku pro operační systém. Tyto jazykové balíčky jsou od verze Windows 7 volně stažitelné ve všech licenčních verzích [17]. Příklad změny lokalizace je vidět na obr. 2.5. V posledních letech však stále platí, že angličtina je dominantní jazyk pro techniku. Proto by bylo vhodné aplikaci lokalizovat do anglického jazyka. Tedy mít možnost volby mezi českou a anglickou lokalizací. Pokud by s programem pracoval zahraniční pracovník, byl by schopen se systém pracovat samostatně. Kromě jazyku bude potřeba zkontrolovat a sjednotit práci s desetinnými tečkami a čárkami. Jelikož LabVIEW odděluje čísla a může brát ohled na lokalizaci systému.



Obr. 2.4: Zkomolený text pod Windows 10 s anglickou lokalizací

2.9 Simulace meřicí úlohy

V podkapitole 2.15 byl zmíněn odpojený režim. Kdy program nekomunikuje s fyzickými zařízeními. Byla by možnost k programu připojit „virtuální“ měřicí karty a



Obr. 2.5: Možnost změny jazykové lokalizace u Windows 10

v nich simulovat reakci vibrometru. Jako příklad měřicí úlohy by mohl být zvolený vibrující nosník. Program by zobrazoval statický obrázek měřené úlohy. Při pohybu virtuálního paprsku po obrázku by se měnila vstupní data, která by simulovala reálná měření.

2.10 Optimalizace a chyby programu

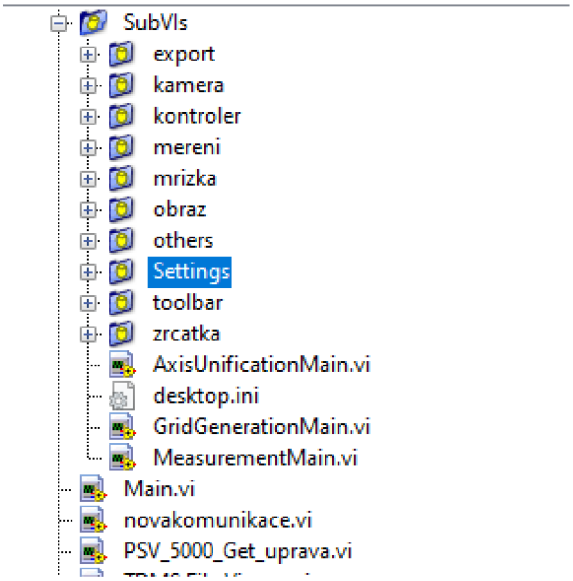
V rámci každého většího vývoje dochází k chybám. Někdy chyby nebo chybějící optimalizace jsou na první pohled patrné. Mnohem častěji chyby nejsou hned vidět, a tak si autor ani neuvědomuje nefunkčnost. V dalších podkapitolách budou popsány tyto dílčí chyby menšího rozsahu.

2.11 Nekonzistentní pojmenování

Program vykazuje nejednotnost ve formátu názvů, pravděpodobně z důvodu delší doby vývoje a práce více osob bez držení se jednotného formátu. V tomto projektu se žel tento jev objevil také. Příkladem může být stavový automat viz obr. 1.9. Zde jsou některé názvy dohromady a jiné rozdělené mezerami. Například *OpticalAxisUnification* a *Before Comm Init*. Zde je vhodnější zvolit první možnost a přejmenovat ostatní stavy na jeden formát jednoho spojeného textového řetězce.

Další roztříštěnost je v pojmenování složek samostatných SubVI. Vidět je to na obr. 2.6. Zde jsou vidět názvy složek a samotných SubVI. První jmenované jsou

povětšinou v češtině, ale samotná SubVI v angličtině. Zde bych názvy sjednotil do angličtiny. Otázka je i v jakém jazyku ponechat komentáře v samotném kódu. Zde se kloním k tomu je ponechat a dále tvořit v češtině. Jelikož nevhodným překladem se zde může ztratit část původního významu poznámky. Dále nepředpokládám vývoj této práce zahraničními studenty. Pokud by i tak k tomu došlo, jistě se najdou rodilí mluvčí, kteří budou schopni správně případné poznámky interpretovat.



Obr. 2.6: Ukázka původního pojmenování souborů

2.12 Práce s periériemi

Původní práce obsahovala měřicí kartu NI 9223. Aktuální verze programu pracuje s s kartou NI 9234. A je pravděpodobné, že v budoucnu bude použita karta jiná. Z toho vyplývá potřeba naprogramovat práci s periériemi více obecně. Program je stavěný pro práci s 4 kanály a v případě většího množství karet je potřeba program aktualizovat. Jednak o rozsah, kanály a prvky na uživatelském rozhraní.

2.13 Servisní režim

Dnešní zařízení, která jsou kombinací softwaru a hardwaru, často obsahují servisní režim. Jedná se o diagnostický nástroj, který pomůže diagnostikovat chybu. Zda je závada v softwarovém řešení nebo vadném hardwaru. Tato možnost je ideální, pokud zařízení nebylo delší dobu používáno a je třeba otestovat, zda je v pořádku.

Servisní režim by mohl otestovat komunikaci s kartami a webovou kamerou. Poté by otestoval komunikaci s kontrolérem. Nakonec by pomocí zrcátek obkroužil

maximální rozsah vychýlení. Během toho by program zobrazoval měřené hodnoty. Tím by se ověřilo, že jsou všechny komponenty v pořádku a přístroj je připraven pro měření.

2.14 Automatická sesouhlasení

Během provádění sesouhlasení je potřeba opakovaně vybrat bod, a poté posunout paprsek do tohoto místa. Jistou výhodou by byla automatická sesouhlasení. Ta by mohla probíhat ve dvou krocích. V prvním kroku uživatel vybere bod. Poté zapne uživatel automatická sesouhlasení. Program by pak v obraze nalezl laserový bod a fyzicky přemístil do vybraného bodu. Uživatel by pak nemusel pomocí šipek dokalibrovávat pozici bodu v obraze a laseru na skutečné ploše. Případně by udělal drobnou korekci.

2.15 Demo (odpojený) režim

Před samotným měřením je třeba inicializovat a ověřit dostupnost vstupních a výstupních periférií. Problém však nastává, pokud tyto periferie nemáme nebo nemůžeme mít dostupné. V tu chvíli není možné program dále používat. Tato vlastnost by za normálních událostí byla zcela v pořádku. Avšak v době distanční výuky byla tato vlastnost přímo nežádoucí. Nebylo možné například ponechat zapnutý kontrolér po celý den. Jelikož se tím značně snižovala životnost přístroje. Například vyšším zahříváním laserové hlavy. Kvůli tomu nebylo možné kód trasovat a zejména ze začátku bylo studium programu velmi náročné.

Odpojený režim, dále jen Demo režim. Má i další uplatnění. Je vhodný pro seznámení se softwarem bez jakéhokoliv HW kromě PC. Není tak potřeba připojovat a chystat celý systém pro omezené seznámení nebo testování. Z podstaty věci nelze všechny věci testovat bez periférii. Jako příklad lze uvést sesouhlasení os nebo komunikace s kontrolérem (pokud není k dispozici simulátor kontroléru). Vhodný je i pro prohlížecí režim. Kdy je třeba data pouze číst. Z výše zmíněných důvodů je odpojený režim nezbytný.

2.15.1 Komunikace kontroléru a měřicí laserové hlavy při Demo režimu

V rámci programu probíhá komunikace mezi kontrolérem a počítačem skrze RS-232. Přímá komunikace se samotnou měřicí laserovou hlavou neprobíhá. Komunikace je vždy zprostředkovaná skrze kontrolér. Míst, kde dochází ke komunikaci s kontrolérem, je v programu mnoho. Cest, jak tuto komunikaci vyřešit, je více. Jedna z nich je

simulace kompletní komunikace s kontrolérem Polytec OFV-5000. Druhou možností je v programu *deaktivovat* místa, kde ke komunikaci dochází.

2.15.2 Komunikace měřicích karet při Demo režimu

Při manipulaci s paprskem nebo sběru dat z kontroléru je třeba měřicích karet NI 9234 a NI 9263. LabVIEW má možnost měřicí karty simulovat. Program nerozezná, jestli se jedná o skutečné nebo virtuální karty a není tak potřeba jej přepracovávat. Vstupní hodnoty na tyto karty lze pak softwarově generovat. Pro potřeby Demo režimu to však není nezbytné. V případě simulace měření by však tato vlastnost byla nezbytná.

2.15.3 Komunikace s kamerou při Demo režimu

Kamera je vstupní periférie používaná na několika místech programu. Zejména pro sesouhlasení os. Kdy uživatel vidí, kde je bod umístěn v programu a kde je ve skutečnosti. Je tedy nutné komunikaci s kamerou nezahajovat při aktivním Demo režimu. Zároveň však je potřeba nahradit obraz z kamery jiným statickým obrázkem. Tento obrázek by si uživatel mohl jednoduše vybrat a vložit do programu.

2.15.4 Virtuální paprsek laseru při Demo režimu

Při běžném režimu je vidět snímek z webové kamery. Na snímku je vidět měřený objekt a laserový paprsek (pokud není posunut mimo zorné pole kamery). Byla by možnost při Demo režimu také do obrázku vložit virtuální paprsek, který by se překresloval do virtuálního obrázku. Ovládal by se stejně jako při reálném měření.

3 Realizace

Kapitola 2 se věnovala prvotní analýze programu. Byl zde uveden i nástin řešení. Tato kapitola se věnuje realizaci některých z výše uvedených nedostatků. Popisovány jsou zde hlavní rysy a problémy se kterými jsem se setkal. Během práce jsem narazil na některé dílčí problémy, které zde nejsou uváděny. Jako příklad uvedu hledání vhodných SubVI pro načítání obrázků, korektní pochopení původního řešení a jiné.

3.1 Realizace Demo (odpojeného) režimu

Při zapnutí programu je uživatel vyzván k vybrání vstupních a výstupních periférií. Původní program však nepočítal s variantou, že by bylo potřeba pracovat s programem bez periférií. Proto ladění programu při absenci připojených zařízení nebylo možné.

Kvůli potřebě pracovat s programem distanční formou bylo nutné vytvořit Demo režim. V práci i programu jej nazývám jako Demo režim, protože se nejedná o plnohodnotnou náhradu a simulaci původního fungování. Cílem tedy bylo umožnit spuštění programu bez jakýchkoliv periferií a s možností si projít celý proces měření. Navíc díky tomuto režimu je možné obsluhu seznámit s fungováním programu na téměř „libovolném“[1] počítači.

3.1.1 Příprava Demo režimu

Jedním z prvních kroků bylo umožnit uživateli zapnout a vypnout Demo režim. K tomuto účelu bylo rozšířeno úvodní okno o indikaci, zda je režim aktivní či nikoliv. Dále o možnost vložení obrázku, který se bude zobrazovat místo záznamu z kamery, a hlavně samotné ovládání tohoto režimu. Toto okno je vidět na obr. 3.1.

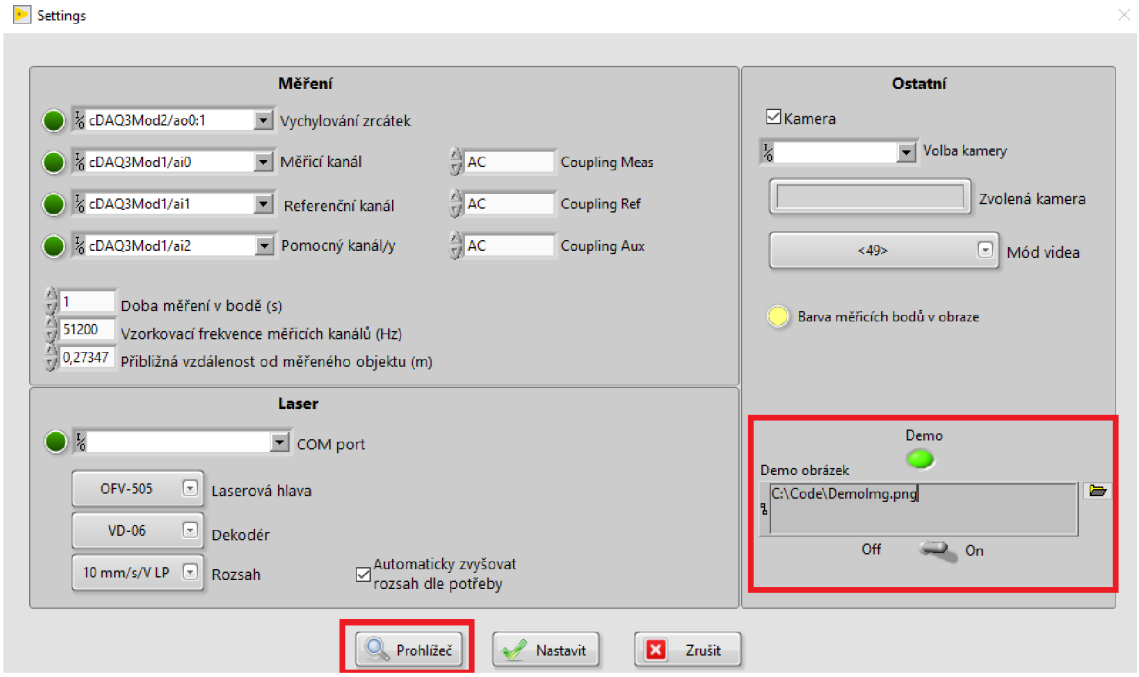
V rámci této úpravy bylo třeba rozšířit SubVI, která se starají o export i import dat ze souboru „Settings.ini“. Aby byla cesta a údaje o obrázku přednačteny. Dále bylo potřeba rozšířit definici clusteru *SettingsCluster*, kterým jsou předávány informace nejen o periferiích v celém programu. Do nabídky s výběrem stavů byl umístěný indikátor (horní lišta programu), který indikuje, zda je zapnutý Demo režim. Tato lišta je vidět ve všech oknech programu.

3.1.2 Komunikace s kontrolérem a měřicí laserovou hlavou

V předchozí podkapitole 3.1.1 byly uvedeny kroky, které jsou spíše přípravou pro samotný režim. Nyní bylo potřeba ošetřit práci s periférie mi. Program ve stavu

[1] musí podporovat běh LabVIEW 2020 a novější, včetně potřebných modulů

Communication Init kontroluje, zda jsou zařízení skutečně připojená. Výsledek pak program zapsal to výše uvedené struktury. Ta pak dále informuje další komponenty, že zařízení nejsou připojena, a nedovolí uživateli pokračovat. Rozhodl jsem se projekt doplnit o SubVI *SetDemoMode.vi*. To v případě aktivovaného Demo režimu, hodnoty o připojení nastaví jako připojené. Jednotlivé SubVI, která provádí kontrolu připojených periférií, si skutečný stav pamatují. Takže skutečný stav připojení HW není přepisovaný a nedochází k pádům programu. Přepisují se pouze hodnoty, které slouží pro logiku přechodu stavů v hlavním stavovém automatu. Většinu práce jsem prováděl v tomto režimu a nezaznamenal jsem žádné chyby, které by vedly k nežádanému chování nebo pádům.



Obr. 3.1: Nová podoba okna pro výběr periférií (červeně jsou označeny nově přidané prvky)

3.1.3 Komunikace s kamerou

V předchozí podkapitole bylo popsáno, jak probíhala „deaktivace“ periférií. Na rozdíl od ostatních periférií v Demo režimu s kamerou, respektive jejím obrazem, potřebujeme pracovat. Kamera se využívá ve stavech *OpticalAxisUnification* , *GridGeneration* a *Main*. Přičemž v poslední stavu je pouze na začátku měření vyfoceno měření, a pak tento obrázek zobrazuje. Na rozdíl od ostatních stavů, kde se jedná o video.

Pro účely Demo režimu bylo vhodné kameru nepoužívat a místo obrazu z kamery použít statický obrázek. Kromě cesty k obrázku bylo při nutné implementaci načíst rozměry vloženého obrázku a načítat jej ze souboru. Původní plán byl SubVI pro vykreslování obrazu z kamery zcela deaktivovat při Demo režimu a vytvořit samostatné SubVI pro práci s obrazem. Později jsem zhodnotil, že je lepší původní SubVI pro práci s kamerou upravit, aby podporovalo zmíněný režim. Tedy v případě, že program běží standardně, bude na obrazovce aktuální záznam. Pokud je Demo režim, bude pouze vykreslovat obrázek vložený ze souboru. Toto SubVI bylo poté potřeba aktualizovat ve 3 výše uvedených stavech.

Testoval jsem Demo režim s různými obrázky a poté přepnutí na běžný režim. Nezaznamenal jsem žádné komplikace. Až na jednu chybu. Při špatně zvolené cestě k obrázku LabVIEW zobrazí nic neříkající hlášení. Program je rozšířen, že pokud je chyba ve špatně zvolené cestě, program tuto chybu smaže a sdělí uživateli, že zvolil špatnou cestu. Uživatel má pak na výběr mezi opravou chyby nebo ukončením programu.

3.2 Nahrazení MathScript Nodes

Před zahájením měření je potřeba provést sesouhlasení os. To znamená převést bod z obrazové roviny do roviny skutečné. Díky tomu je možné vybrat body na obrazovce počítače a vychylovací systém pak laserový paprsek na tyto body namíří.

V původní práci je SubVI naprogramované pomocí LabVIEW MathScript Nodes. Ten nebude v budoucnu podporovaný a je třeba jej nahradit [16].

Výše zmíněnou operaci převodu z jedné roviny do druhé je možné vyjádřit maticově dle rovnice 3.1. Ve skutečnosti se však pracuje pouze s rovinami a přidání další dimenze pomáhá k nalezení lepšího řešení. Díky převedení úlohy do trojrozměrného prostoru lze i posouvat střed báze. Nikoliv jen otáčet nebo zvětšovat vektory. Symboly x_z a y_z jsou souřadnice zrcátek (vychylovacího systému), x_k a y_k souřadnice obrazu z kamery. Cílem je tedy nalézt matici T (viz rovnice 3.2), respektive její koeficienty a až f.

$$\begin{bmatrix} x_z \\ y_z \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_k \\ y_k \\ 1 \end{bmatrix} \tag{3.1}$$

$$T = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Z výše zmíněných rovnic je patrný závěr, že bude potřeba alespoň 6 rovnic pro určení koeficientů. Přičemž je zapotřebí znát alespoň tři souřadnice bodů z obrazu kamery (x_k i y_k) a vychylovacího systému zrcátek (x_z i y_z). Původní práce obsahuje 4 zvolené body (obou prostorů). Rozhodl jsem se tento počet zachovat, protože se v praxi osvědčil.

Poté je třeba udělat pomocný výpočet pro získání koeficientů do matice T. Dle rovnice 3.3 matice A obsahuje souřadnice zvolených bodů z vychylovacího systému. Matice B souřadnice bodů z obrazu kamery a poslední matice C obsahuje hledané koeficienty. Rozepsaný formát matic obsahuje rovnice 3.4. Pro samotný výpočet koeficientů je třeba rovnici upravit do tvaru dle rov. 3.5. Výsledné hledané koeficienty z matice C pak je nutné upravit do výchozí tvaru matice T.

$$A \cdot C = B \tag{3.3}$$

$$\begin{bmatrix} x_{k1} & y_{k1} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{k1} & y_{k1} & 1 \\ x_{k2} & y_{k2} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{k2} & y_{k2} & 1 \\ x_{k3}+x_{k4} & y_{k3}+y_{k4} & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{k3}+x_{k4} & y_{k3}+y_{k4} & 2 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x_{z1} \\ y_{z1} \\ x_{z2} \\ y_{z2} \\ x_{z3}+x_{z4} \\ y_{z3}+y_{z4} \end{bmatrix} \tag{3.4}$$

$$C = A^{-1} \cdot B \tag{3.5}$$

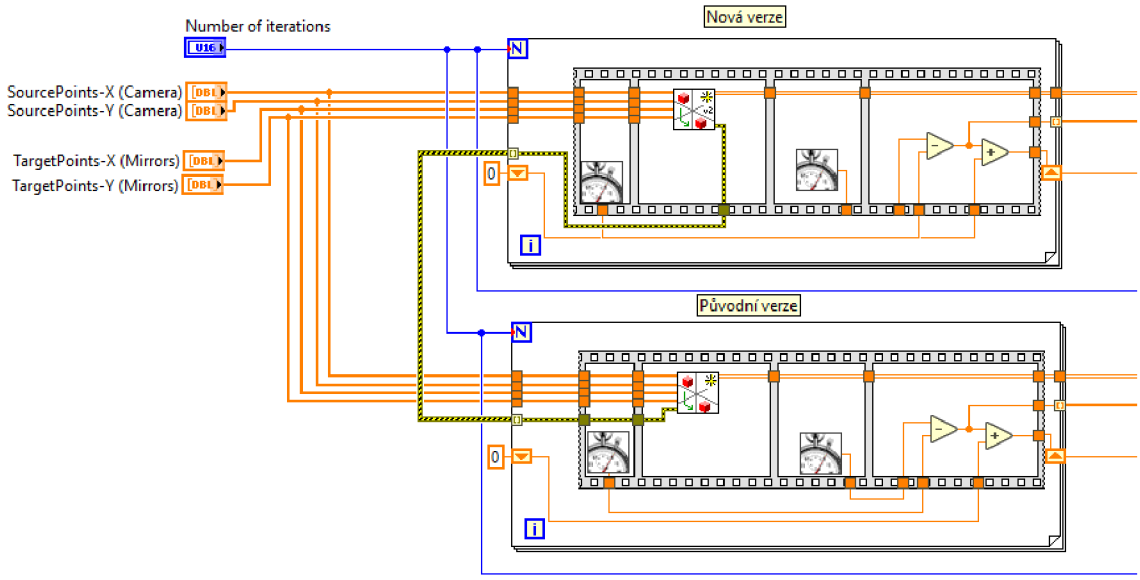
Výsledná implementace pak vychází z výše uvedeného postupu, kdy výpočet je rozdělen na jednotlivé kroky. I nově vytvořená SubVI se snaží odpovídat zmíněným krokům, aby řešení bylo modulární a na první pohled čitelné. Způsob řešení v původní práci nebyl na první pohled zřejmý a v dokumentaci byl postup popsán velmi obecně. Proto cílem této podkapitoly bylo lépe zdokumentovat řešení a přepracovat ho pro použití bez MathScript nebo Matlab Script Nodes.

3.2.1 Porovnání původní a nové verze výpočtu transformační matice

V předchozí podkapitole byl popsán postup výpočtu sesouhlasení. Po dokončení nové verze SubVI bylo potřeba ověřit zda-li výpočet probíhá korektně. Použil jsem tedy původní verzi programu a provedl sesouhlasení 4 bodů. Vstupní a výstupní data pro výpočet transformační matici jsem si uložil.

Poté jsem vytvořil testovací VI, kde jsem vložil tato data do původního a nového SubVI pro výpočet sesouhlasení. Část tohoto VI je vidět na obr. 3.2. Výslednou

transformační matici jsem si zobrazil a vizuálně zkontroloval. Numerické výsledky výpočtu byly totožné.



Obr. 3.2: Ukázka kódu, který porovnával rychlost výpočtu

Pro porovnání rychlosti jsem ještě vytvořil smyčku, která 100krát výpočet opakovala. Díky tomu bylo možné „objektivněji“ porovnat rychlost původního a výsledného výpočtu. Porovnal jsem i výpočet na školním (procesor i3-8100, RAM 8 GB) a osobním počítači (procesor i5-4310U, RAM 8 GB). Výsledky porovnání rychlosti výpočtu jsou vidět v tabulce 3.1. Histogramy výpočtů rychlosti pro oba počítače jsou vidět na obr. 3.3 a 3.4.

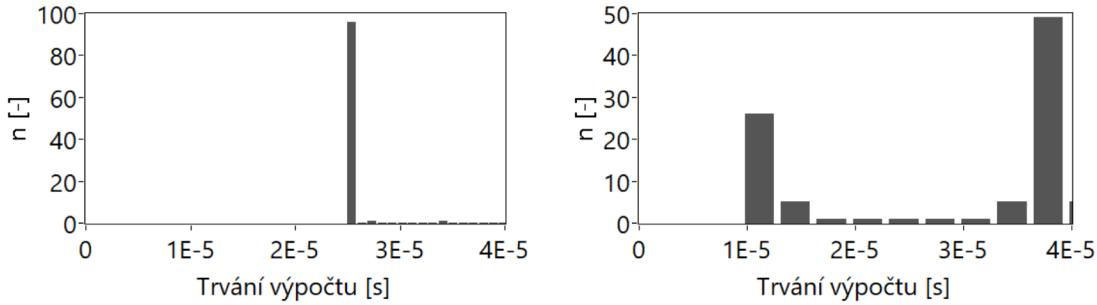
Tab. 3.1: Srovnání průměrné rychlosti původního a nového řešení při 100 iteracích

Konfigurace PC	Původní verze [μs]	Nová verze [μs]
procesor i5-4310U, RAM 8 GB	25,60	30,10
procesor i3-8100, RAM 8 GB	3,07	6,98

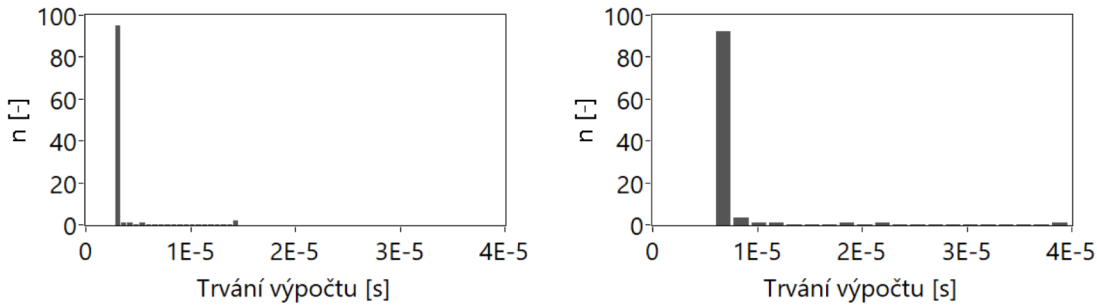
Z výše uvedených podkladů je jasně vidět, že mé řešení je pomalejší. Je to dáno několika faktory. Prvním je, že původní SubVI používá MathScript Nodes, který je optimalizovaný pro výpočty. V něm bylo zadáno pár rovnic a výsledek byl přiveden na výstup. U mé implementace pomocí jazyka G nebylo možné takhle jednoduše provést výpočet. Bylo potřeba sestavit jednotlivé matice a naplnit je daty. Dále průběžně kontrolovat, zda nedošlo k chybě atd. Výpočet byl modulární a bylo použito

více SubVI. Každé předávání dat mezi SubVI stojí výpočetní čas. Bylo potřeba se rozhodnout, zda zrušit modularitu a zrychlit výpočet nebo modularitu zachovat. Rozhodl jsem pro zachování modularity, protože budoucí údržba a rozšíření budou podstatně jednodušší. Výpočet sesouhlasení probíhá během kalibrace i celého programu pouze jednou. Navíc výpočet probíhá v řádu mikrosekund, což lze považovat stále za rychlý výpočet.

Hlavním cílem bylo odstranit původní MathScript Nodes, protože ho výrobce nebude v budoucnu podporovat. A nahradit původní řešení jinou alternativou, která bude dlouhodobě podporovaná. To se podařilo a nové řešení fungovalo i při reálném nasazení.



Obr. 3.3: Srovnání trvání výpočtu původního (vlevo) a nového (vpravo) SubVI (100 iterací, procesor i5-4310U, RAM 8GB)



Obr. 3.4: Srovnání trvání výpočtu původního (vlevo) a nového (vpravo) SubVI (100 iterací, procesor i3-8100, RAM 8 GB)

3.3 Neintuitivní uživatelské rozhraní

V podkapitole 2.2 je rozepsána problematika neintuitivního uživatelské rozhraní. To je vstupní branou pro práci s programem. Jeho komfortní užívání je často rozhodující pro celkový dojem z hotového produktu.

V této podkapitole jsou rozepsány jednotlivé komponenty, které byly upraveny nebo přidány pro lepší práci s programem a usnadnily tak práci obsluze při každém měření. U návrhu uživatelského rozhraní jsem se snažil postupovat podle obecného postupu vývoje [18].

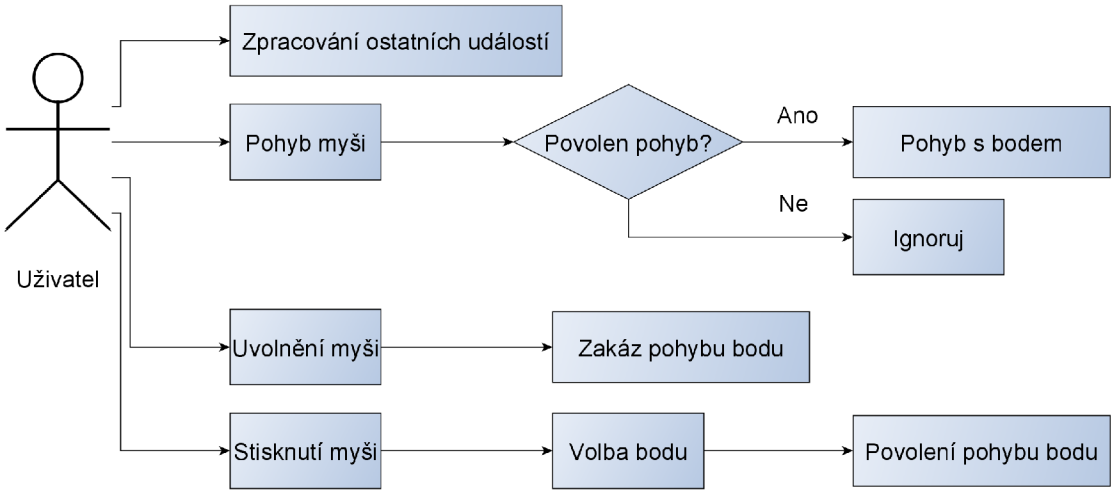
3.3.1 Přemístění bodů pomocí myši

Jedním z prvních kroků před zahájením měření je provedení sesouhlasení. V původní verzi programu bylo potřeba s body a laserem pohybovat pomocí tlačítek na obrazovce. Uživatel sice provedl sesouhlasení, ale pomocí šipek na obrazovce to nebylo příliš komfortní.

Zcela se nabízelo pohybovat s body pomocí myši. Úkol tedy bylo umožnit přemisťování bodů pomocí myši a zároveň navigovat souřadnice paprsku do právě vybraného bodu. To je kvůli rychlému provedení kontroly, zda je korektně provedené sesouhlasení. Tedy při pohybu bodu na obrazovce se pohybuje i laser. A pokud je sesouhlasení provedeno správně, tak se laser se zvoleným bodem překrývají.

V SubVI *AxisUnificationMain.vi* a *GridGenerationMain.vi* byly přidány stavy a SubVI pro detekci vybrání bodu a manipulaci s ním. Zjednodušenou logiku algoritmu je vidět na obr. 3.5.

Nyní je možné s body pohybovat pomocí myši. Jak při sesouhlasení, tak při nastavování měřicí mřížky. Při pohybu bodu se pohybuje i laser a je možné rychle zkontrolovat správné fungování sesouhlasení. Zároveň byla zachována možnost pohybovat s bodem skrze tlačítka v podobě šipek na obrazovce.



Obr. 3.5: Zjednodušený vývojový diagram algoritmu pro manipulaci s bodem pomocí myši

3.3.2 Navigace laseru pomocí kláves

Výše bylo uvedeno, že s paprskem je možné manipulovat pouze skrze tlačítka v podobě šipek na obrazovce. Kvůli jednodušší obsluze by byla vhodnější volba pohybu skrze šipky na klávesnici.

Rozšířil jsem tedy SubVI *AxisUnificationMain.vi* a *GridGenerationMain.vi* o detekci stisku nebo držení těchto kláves. Uživatel tak může pohybovat s paprskem pomocí šipek na klávesnici nebo obrazovce. Opět je zvýšen komfort při provádění sesouhlasení.

3.3.3 Inicializace kalibračních bodů

Při provádění sesouhlasení musel uživatel nejprve vložit 4 body do obrazu. Poté provedl kalibraci pro každý bod. Jak bylo uvedeno v podkapitole 2.2, uživatel nebyl nijak naváděn, že má vkládat nějaké body. A teprve poté s nimi pracovat.

Přišlo mi logičtější uživateli při spuštění tyto body inicializovat. Tím je zajištěno, že bude zadaný správný počet bodů. Navíc, když uživatel uvidí čtyři body na obrazovce, intuitivně ho to navede ke korektnímu provedení sesouhlasení.

Pro tento účel jsem vytvořil SubVI *ImageInitPoints.vi*, které body inicializuje. K dosažení co nejpřesnějšího sesouhlasení je vhodné vkládat body co nejdále od sebe. Proto se inicializují body u okraje obrazu. Není vhodné vykreslovat body přímo na okraji, protože jsou ořezány a nejsou dobře vidět. Zvolil jsem vzdálenost 30 pixelů od okraje obrazu. Z toho vyplývá, že program nepodporuje vstupní obrázek nebo záznam kamery menší než 100 x 100 pixelů (mnou zvolený limit). Nutno podotknout, že v době 4k monitorů požadavek na obraz větší než 100 x 100 pixelů je naprosto legitimní. Příklad, jak vypadá inicializace bodů, je vidět na obrázku 3.6.

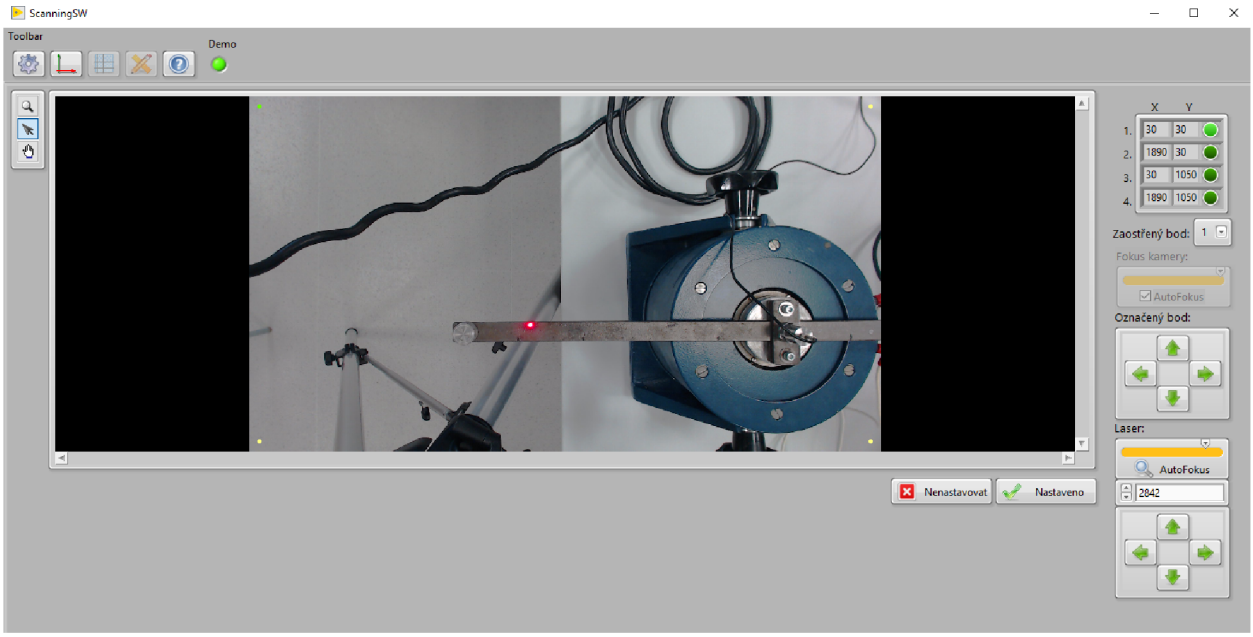
3.3.4 Umístění prvků

V předchozích podkapitolách byly zmíněny úpravy, které zvýšily uživatelský komfort. Díky inicializaci bodů u sesouhlasení již nebylo potřeba vkládat body do obrazovky. Nabídka se zjednodušila.

Nastavování fokusu laseru bylo možné dělat pouze pomocí posuvníku. Toto řešení mělo pár nevýhod. Uživatel nevěděl, jakou má právě nastavenou hodnotu, a menší korekce hodnoty pomocí myši byly problém. Proto jsem doplnil možnost vkládat hodnotu fokusu číselně. Jak je vidět na obrázku 3.6.

3.3.5 Kontrola nastaveného fokusu u referenčního bodu

Při provádění sesouhlasení uživatel běžně sesouhlasí zvolené body na obrazovce s reálnou polohou paprsku. Potom tyto hodnoty potvrdí a pokračuje v nastavování



Obr. 3.6: Nové rozmístění prvků pro sesouhlasení

mřížky. V původní verzi dojde k neočekávanému chování. Laserový bod se neobjevuje na zvolených pozicích, i když bylo sesouhlasení provedeno „správně“. A systém se chová, jak kdyby k žádnému sesouhlasení nedošlo.

Začal jsem pátrat po příčině. Program po potvrzení sesouhlasení začne počítat transformační matici ze 4 zvolených bodů. Dochází zde i k výpočtu odhadu vzdálenosti od laserové hlavy k měřenému objektu. Tato hodnota se pak ukládá do *SettingsFGV.vi* a je dále používána při generování mřížky a měření. Odhad vzdálenosti je počítán na základně nastaveného fokusu u zvoleného referenčního bodu. A zde bylo místo vzniku nevhodného chování.

Program na základě zvoleného fokusu odhaduje vzdálenost od měřeného objektu. Využívá k tomu matematickou funkci, která nabývá i záporných hodnot vzdálenosti. Pro zaostření na vzdálenost větší než 7,5 cm (od deflektorové hlavy) jsou hodnoty kladné. Pokud uživatel o tomto fungování nevěděl a nastavil špatně fokus pro referenční bod. Funkce vypočítala zápornou vzdálenost a práce s body nefungovala správně. V praxi bývá předmět vzdálený půl metru až metry. Proto upravovat funkci pro odhad vzdálenosti u menších vzdáleností je bezpředmětné.

Uživatel není nijak konfrontován s touto skutečností a jejími důsledky. Proto jsem do programu zakomponoval kontrolu fokusu. Vycházel jsem z naměřených hodnot v původní práci Ing. Tomka [1]. Zde autor uváděl bezrozměrnou hodnotu fokusu 2724,9 pro vzdálenost 7,5 cm od objektu k hraně deflektorové hlavy. Proto jsem zvolil hraniční hodnotu 2725. Jestliže uživatel zadá hodnotu 2725 a větší, program se bude chovat jako doposud. Program plynule přejde do nastavení mřížky. V opačném

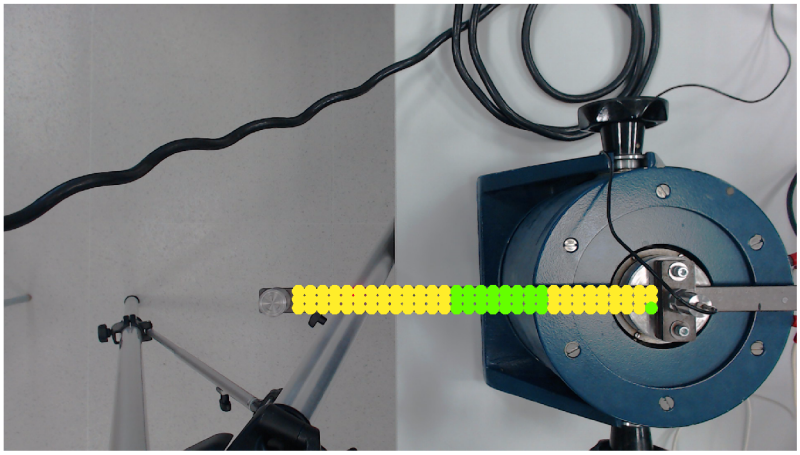
případě program uživatele informuje, že zvolil hodnotu mimo rozsah a jaké to má důsledky. Uživatel si pak může vybrat, jestli bude pokračovat, nebo hodnotu změní. Zvolená hraniční hodnota se v praxi osvědčila a byla tedy otestována jako vhodná.

3.4 Export snímků měření

Během měření je na obrazovce programu vidět obrázek z kamery včetně zvolených bodů. Uživatel má tak přehled, které body měřil, a může odhalit chybně naměřená data. Tento obrázek se neukládá společně s naměřenými daty. Uživatel jej může ručně uložit. Exportovat se mu však podaří pouze samotný obrázek bez zvolených bodů. V praxi se často stává, že člověk zapomene pokaždé exportovat i tento samotný obrázek. Pokud jsou data otevřená s delším časovým odstupem, je jejich interpretace bez obrázku komplikovanější.

Řešením je exportovat automaticky obrázek po každém měření. Vhodnější je však exportovat obrázky dva. Jeden z kamery, čistě bez bodů, může být v budoucnu použit jako podklad, nad kterým se budou vykreslovat další údaje. A druhý, který bude obsahovat i měřené body. Díky nim bude hned jasné, co a kde se měřilo.

V programu jsem tedy rozšířil původní SubVI *MeasurementMain.vi*. V ukončovacím stavu zmíněného SubVI se nejdříve exportuje snímek z webkamery s názvem „Img.png“ do složky měření. Poté je třeba exportovat snímek znovu i s body. Tato úloha je komplikovanější, protože měřené body se nezakreslují přímo do obrázku, ale do vrstvy nad obrázkem. Je nutné tedy podkladovou vrstvu (obyčejný snímek) a vykreslené body sloučit. Vznikne tak nový obrázek, který je do složky exportován s názvem „ImgWithPoints.png“. Příklad takového výstupu je na 3.7. Tyto názvy jsou schválně statické. Vysvětlení je v podkapitole 3.5.



Obr. 3.7: Ukázka snímku „ImgWithPoints.png“

3.5 Export dat

Cílem každého měření je získat data. Proto by součástí každého měřicího softwaru měla být možnost exportu získaných dat. Tento software tuto možnost obsahuje. Ovšem není zcela optimální. Hlavním nedostatkem považuji roztříštěnost ukládání souborů. Část souborů se exportovala do adresáře *data*, který je automaticky vytvořen ve složce se souborem aplikace. Přehled souborů, které jsou důležité pro vyhodnocení měření, je v tabulce 3.2. Soubory označené * byly uloženy do adresáře dle uživatele. Navíc názvy souborů obsahují generickou část (datum), která znemožňuje soubory strojově shromáždit a zpracovat.

Tab. 3.2: Přehled názvů souborů z výsledku měření

Původní názvy	Nové názvy
points_*RRRR-MM-DD_HH-MM-SS*.ini	PointsConfigurationStructure.ini
Structure_*DD-MM-RRRR_HH-MM-SS*.txt*	PointsStructure.txt
data_*RRRR-MM-DD_HH-MM-SS*.tdms	data.tdms
data_*RRRR-MM-DD_HH-MM-SS*.tdms_index	data.tdms_index
-	Img.png
-	ImgWithPoints.png
Data_ModalVIEW_*INDEX*.uff*	Data_ModalVIEW_*INDEX*.uff
Settings.ini*	Settings.ini

Prvním logickým krokem bylo data exportovat do samostatných složek. Co měření, to jedna složka. Rozhodl jsem se pro názvy složek „data_*RRRR-MM-DD_HH-MM-SS*“. Přičemž *R* symbolizuje jednu cifru roku. Tedy *RRRR* je myšleno například 2021. Obdobně je to myšleno u ostatních písmen.

Pro práci s měřenými daty slouží *TDMSSavingFGV.vi*. Ten je na začátku programu inicializován pro vytvoření TDMS souboru. Toto SubVI jsem upravil, aby vytvořilo nejen soubor, ale i adresář. Zároveň výstupem z něho byla cesta k nově vytvořenému adresáři. Toto SubVI bylo zdrojem problémů, proč nefungovalo korektně ukládání dat při opakovaném měření. O tom více v podkapitole 3.6.

Po úspěšné realizaci předchozího kroku bylo třeba sjednotit ukládání zbývajících souborů. V rámci tohoto kroku byly přemístěny všechny nové a původní SubVI pro export do SubVI *MeasurementMain.vi*. Některá původní SubVI musela být pro korektní chování pozměněna. Například *Structure_ModalVIEW.vi* exportoval nově pojmenované soubory *PointsStructure.txt* a *PointsConfigurationStructure.ini*. Přičemž první soubor exportoval do adresáře dle vstupní cesty, ale druhý soubor exportoval

do pevně zvoleného adresáře. Program tedy nepoužíval vstupní cestu jako cílovou u obou souborů.

Výsledné řešení problematiky exportu vypadá následovně: při zahájení měření je vytvořena jedna složka (ve složce *data*), která v názvu obsahuje datum. V ní jsou uloženy soubory z provedeného měření, které jsou připravené pro strojové čtení.

Platí tedy, jedno měření, jedna složka. Do složky jsou uloženy obrázky z kamery a také zkopírován platný konfigurační soubor, viz tabulka 3.2. Nyní má uživatel z každého měření jednu složku se všemi důležitými soubory. Což u původní verze neplatilo. Dále jsou soubory připravené pro strojové čtení.

3.5.1 Export pozic bodů ve stopách

V předchozí podkapitole 3.5 byl uveden seznam exportovaných souborů. V souboru „PointsStructure.txt“ jsou uvedeny data o odhadu souřadnic ve stopách. Stopy zde byly použity kvůli kompatibilitě s programem ModalVIEW. Přičemž je to jediné místo v programu, kde se místo metrů používají stopy.

V rámci konzultací mi bylo sděleno, že je důležitější ukládat data v jednotce SI, než v jednotce, kterou očekává ModalVIEW. Proto byl program upraven, aby data generoval v metrech.

3.6 Problém s opakovaným ukládáním dat

V původním programu je po provedení nastavení řada na samotné měření. Uživatel zapnul pomocí tlačítka měření a pokud předchozí kroky provedl správně, došlo k automatickému měření. Uživatel však může chtít měření opakovat. Proto znovu zadal povel programu, aby měřil. Program však nevytvořil nový soubor pro druhé měření, ale použil stávající soubory. Tedy uživatel si opakovaným měřením přepisoval soubory, aniž by byl jakkoliv programem o tomto faktu informován.

Cílem je tedy, aby program pro každé měření vytvářel samostatné složky a soubory. Export hlavního TDMS souboru provádělo SubVI *TDMSSavingFGV.vi*. To bylo modifikováno i kvůli exportu dat, viz podkapitola 3.5.

Ve zmíněném SubVI byla uložena cesta k souboru, která se přepisovala při inicializaci měření. První myšlenkou tedy bylo, toto SubVI upravit, aby při druhém a dalším měření se vždy generovala nová cesta. Myšlenka byla taková, že při každé práci s daty by se soubor otevřel a po ukončení daného kroku zase uzavřel. V původní verzi se otevřel TDMS soubor při inicializaci a byl otevřený do konce měření.

Při dokončení implementace končil program v chybě, že má neplatnou referenci souboru. I když na první pohled bylo vše dobře implementováno. Důvod chyby byl

v tom, že SubVI *TDMSSavingFGV* bylo voláno asynchronně z různých části nadřazeného souboru. Více SubVI nemůže v jeden moment zapisovat nebo číst TDMS soubor. Můžou však v různém pořadí otevírat a zavírat soubor, a tím mít uloženou neplatnou referenci na něj. Program paralelně s měřením sleduje události (Eventy) uživatele a zobrazuje aktuálně naměřená data.

Oprava fungování skrze *TDMSSavingFGV* se ukázala jako velmi komplexní problém. Zvláště díky velmi stručné dokumentaci v samotném kódu. Ve výsledku by to znamenalo prakticky přeprogramovat celé SubVI *MeasurementMain.vi*. Navíc pro testování by bylo potřeba provádět řadu kontrolních testů s fyzickým přístrojem.

Z výše zmíněných důvodů jsem zvolil jinou cestu pro řešení tohoto problému. Upravil jsem vnitřní stavový automat a nadřazený systém, aby se při každém měření znovu inicializoval a hlavně ukončil měřicí SubVI *MeasurementMain.vi*. Díky tomu úprava původního kódu není tak radikální. A program si nepřepisuje data. Ilustrační podoba algoritmu je vidět na obr. 3.8. Program se nyní chová žádaným způsobem a uživateli při opakovaném měření již nejsou přepisována data.

3.7 Optimalizace a chyby programu

Během vývoje rozsáhlejšího programu je velmi náročné odhalit všechny chyby a nedostatky. V praxi jsou součásti firem celá oddělení, která se zabývají pouze testováním. Jak bylo zmíněno dříve, vývoj programu se po obhajobě původní práce nezastavil. Program nadále rozvíjeli akademičtí pracovní Ústavu automatizace a měřicí techniky. Tím, jak na projektu začalo pracovat více lidí, je umocněn prostor pro vytvoření nových chyb.

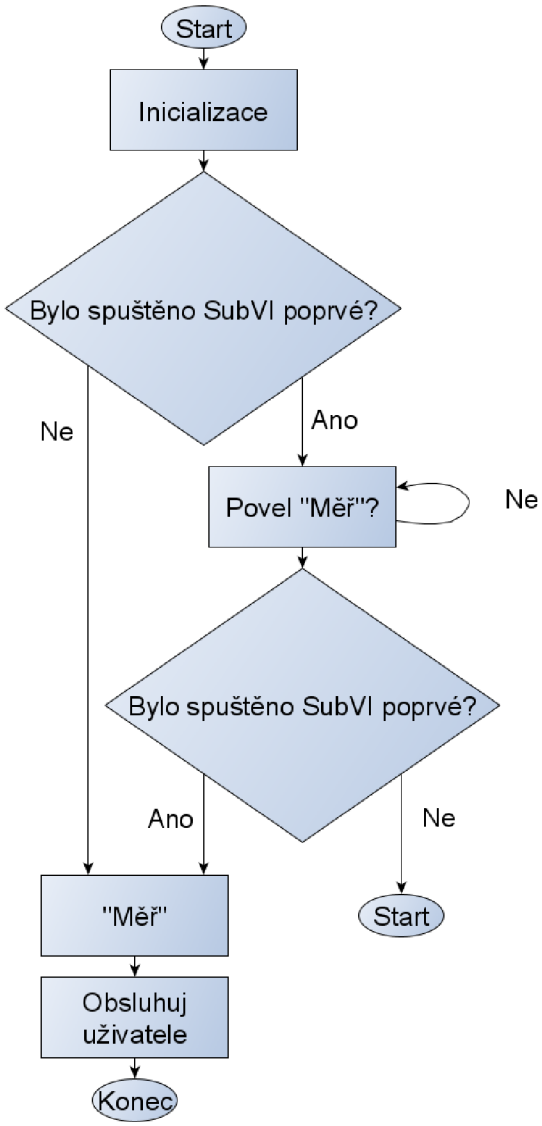
3.7.1 Chybějící načítání Pomocného kanálu

Během inicializace si program načítá data ze souborů „ControllerSettings.ini“ a „Settings.ini“. Díky tomu uživatel může použít přednastavenou konfiguraci z minulého měření. Problém byl však s načítáním nastavení pro Pomocný kanál. Ten program ze souboru nenačítal a ani ho neexportoval.

Bylo tak nutné opravit tuto chybu. Načítání dat ze souboru bylo opraveno v *ReadConfigFiles.vi* a *SettingsMain.vi*. Následně proběhla kontrola, že načítání skutečně funguje korektně.

3.7.2 Blokování zadávání Referenčního kanálu

Při spuštění programu je vyzván uživatel k nastavení parametrů a vstupních i výstupních periférií. Hodnoty jsou přednačteny ze souboru „Settings.ini“. Program



Obr. 3.8: Zjednodušený vývojový diagram pro opakování měření

konkrétně u Referenčního kanálu kontroloval, zda je připojen. Pokud připojen nebyl, políčko s výběrem bylo zašedlé a uživatel tak nemohl tento kanál změnit. Toto chování považuji za nelogické, protože když mám nastavený špatný kanál, tak ho přeci potřebuji změnit. Pravděpodobně algoritmus měl fungovat jinak. Domnívám se tedy, že se jedná o nedokončenou historickou změnu.

Bez zásahu do programu bylo možné referenční kanál měnit pouze přes soubor „Settings.ini“, kde v souboru uživatel ručně napsal periférii s měřicí kartou. To bylo poněkud kostrbaté. Rozhodl jsem se proto prvky, které provádí blokaci pomocí „Diagram Disable Structure“, deaktivovat. Tím jsem toto nevhodné chování eliminoval. Zároveň je možné rychle obnovit původní fungování, pokud by se prokázaly nějaké nedostatky. Během kontrolního měření se však nežádoucí chování neprojevilo.

3.7.3 Nemožnost korektně vypnout okno s nastavením periférií

Během nastavování periférií se často stane, že uživatel potřebuje vypnout program v této úvodní fázi. Problém však nastává v návrhu celého stavového automatu. Uživatel nemůže ukončit program během úvodního nastavování. Jediná cesta, jak program ukončit, je „proklikat“ se do stavu pro sesouhlasení, kde ukončení probíhá korektně. Nutno podotknout, že dokud nebyl dokončený Demo režim, uživatel mohl ukončit program často jen pomocí systémového ukončení programu. Program mu totiž hlásil, že nemá připojené periférie a nechtěl mu umožnit přechod do dalšího stavu.

Řešením bylo tedy rozšířit SubVI *SettingsMain.vi* tak, aby při stisku tlačítka „Zrušit“ vydalo nadřazenému systému povel, že se má program ukončit. Nadřazený program vyčistil všechna chybová hlášení, která jsou generována při špatně zvolené konfiguraci, a korektně celý program ukončí.

3.7.4 Nepřetržité sledování asynchronních událostí

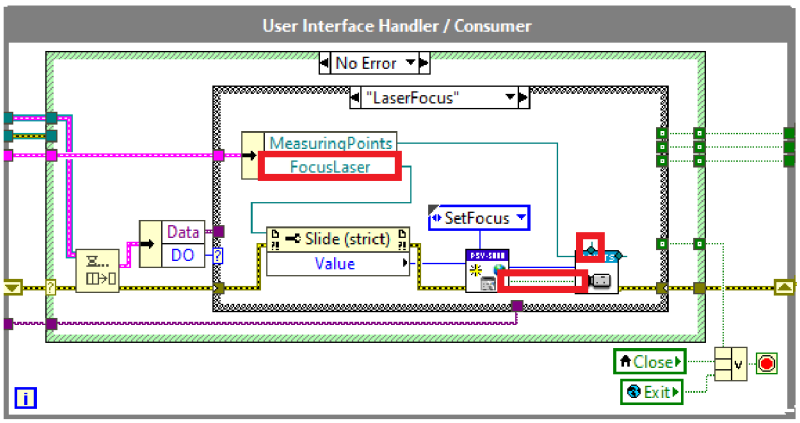
Během implementace Demo režimu bylo potřeba přidat tlačítko pro přepínáním mezi Demo režimem a normálním fungováním. Zjistil jsem však, že v SubVI *SettingsMain.vi* je Event struktura, která nemá nastavený timeout. To znamená, že běží nepřetržitě. Díky tomu nebylo možné paralelně indikovat, jestli uživatel má nebo nemá aktivní Demo režim. Řešením bylo přidat timeout po každých 100 milisekundách.

3.7.5 Špatná implementace ručního nastavení fokusu

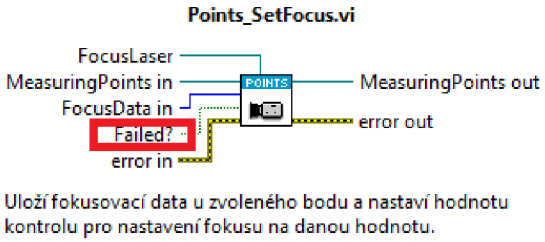
Při prvním seznamování s projektem jsem si zkoušel ruční fokus paprsku. Ten se podle zvolené hodnoty měnil a měl jsem za to, že tato část kódu je v pořádku. Při implementaci Demo režimu však program končil v chybě a snažil jsem se dohledat zdroj chyby. Zjistil jsem, že zdrojem chyby bylo SubVI *Points_SetFocus.vi*.

Na obrázku 3.9 je vidět část kódu, kde vznikají chyby. Červeně jsou zvýrazněna místa, která jsou zdrojem chyb. První chybou je chybějící připojení reference. Data se pak nemohla ukládat, protože chyběla adresa na pole.

Druhá chyba je závažnější a hlavně není na první pohled vidět. Prvně je potřeba se podívat na obr. 3.10. Na něm je zvýrazněný vstup *Failed?* Ten indikuje, zda došlo k chybě. Před tímto SubVI je připojené SubVI *PSV_5000_FGV.vi*, které pošle příkaz pro provedení změny fokusu. Dále na obr. 3.11 je vidět zvýrazněný výstup *Connected?*. Z výše uvedeného je zřejmé, že při připojeném kontroléru SubVI *PSV_5000_FGV.vi* generuje na výstupu *Connected?* hodnotu True do vstupu *Failed?*. SubVI *Points_SetFocus.vi* je tedy při připojeném kontroléru v chybě a data o nastavení zaostření neukládá. A naopak v Demo režimu se program hroutil.

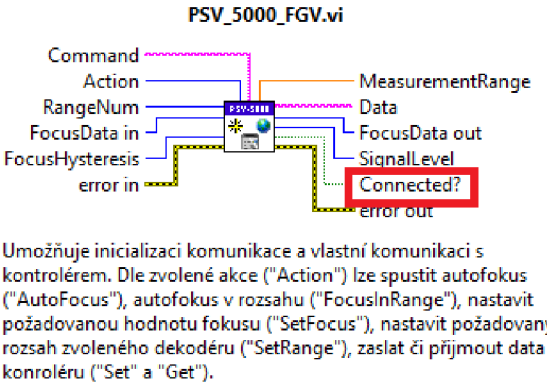


Obr. 3.9: Ukázka části původního kódu s chybnou implementací ručního fokusu (červeně je zdůrazněna chyba)



Obr. 3.10: Dokumentace pro SubVI *Points_SetFocus.vi*

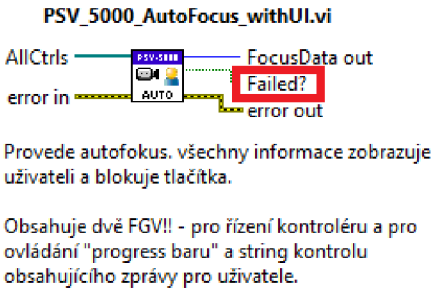
Tato chyba byla vyřešena trvale přivedenou konstantou False na vstup *Failed?*. Do zmíněného SubVI vstupuje i Error drát, který v případě chyby funkci SubVI také deaktivuje. Navíc pokud by se stalo, že program neindikuje chybu pomocí Error drátu, jediné co by se stalo, by bylo zapsání nesmyslné hodnoty do pole. Ale tento akt nepovede ke zhroucení programu. A program bude vypnut nadřazeným SubVI.



Obr. 3.11: Dokumentace pro SubVI *PSV_5000_FGV.vi*

Tato chyba pravděpodobně vznikla při kopírování části kódu, která se stará o automatický fokus. Zde je kód téměř stejný. Akorát zde místo SubVI *PSV_5000_FGV.vi* je použito SubVI *PSV_5000_AutoFocus_withUI.vi*. Na obr. 3.12 je vidět, že zde je výstup *Failed?* a tak pravděpodobně došlo k záměně. Tento výstup je naopak korektně použit u provádění automatického fokusu laserového paprsku.

Obě zmíněné chyby byly v SubVI *AxisUnificationMain.vi* a *GridGenerationMain.vi*. Na obou místech byly chyby opraveny a otestovány.



Obr. 3.12: Dokumentace pro SubVI *PSV_5000_AutoFocus_withUI.vi*

3.8 Prohlížecí režim

V předchozích podkapitolách byla zmíněna možnost exportu dat. Tato data je možné dále zpracovat pomocí externích programů jako je například ModalVIEW. Původní měřicí software samostatně neumožňuje interpretovat získaná data. Proto by bylo vhodné doplnit Prohlížecí režim. Díky němu by bylo možné získaná data interpretovat a provést alespoň základní analýzu.

Na začátku řešení tohoto režimu bylo potřeba rozšířit původní stavový automat o 2 nové stavy „PreBrowser“ a „Browser“. Přičemž první měl sloužit pro přepnutí okna Control Tabu a vyčištění potenciálních chyb. Uživatel může díky tomu spustit Prohlížecí režim i bez správně zvolených periférií. Druhý zmíněný stav slouží pro samotný běh prohlížeče.

Při vývoji jsem měl k dispozici VI od vedoucího práce. To byl velmi účelový prohlížeč dat. Nebyl však připravený pro integraci do celého projektu. Bylo potřeba tedy oddělit výpočty do samostatného SubVI a použít jej jako výpočetní jádro. To se povedlo a vzniklo SubVI „DataCalc.vi“.

Po získání výpočetního jádra bylo potřeba vytvořit vrstvu programu, která je schopná komunikovat s uživatelem. K tomuto účelu slouží SubVI *BrowserMain.vi*. To pracuje s referencemi všech prvků na hlavním Front panelu programu (jedná se o Front panel nadřazeného VI). A zároveň obsluhuje všechny události uživatele i programu. Kostra byla použita z původního *MeasurementMain.vi*.

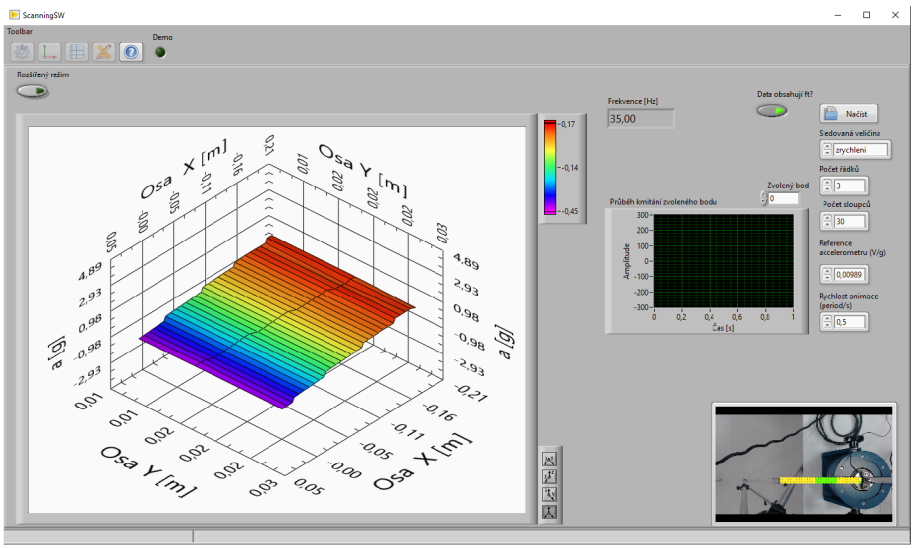
Hlavní program je velmi komplexní a není vhodný pro testovací vývoj, proto byl Prohlížecí režim vyvíjen samostatně mimo projekt. Zde jsem naprogramoval a otestoval jednotlivá SubVI, která zpracovávala dílčí úkoly. Od začátku byla tato část kódu programovaná modulárně, aby její budoucí rozšíření nebylo příliš komplikované.

Po otestování prototypu přišla řada na integraci. Během integrace jsem narazil na pár problémů s referencemi. Jednalo se o reference na jednotlivá okna prvku Tab Controlu, při přidání dalšího okna se změnil datový typ reference. Bylo pak potřeba na více místech tuto definici přetypovat. Obdobné potíže byly s vytvořením reference na ImageControl.

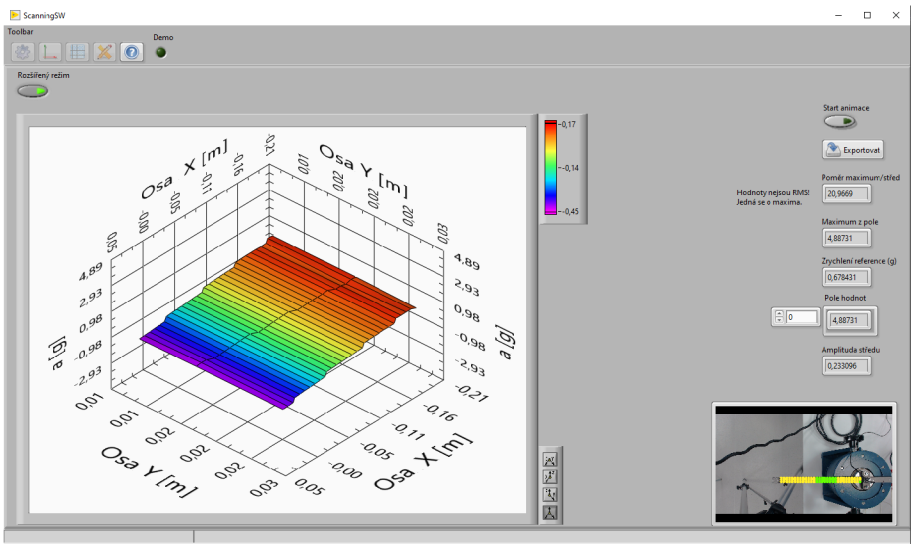
Integrace prototypu díky vhodně navrženému konceptu nevyžadovala mnoho úprav. Koncept pracuje s modelem producent-konzument. Při vzniku události producent vloží do fronty povel a konzument pak tuto událost obslouží. Například při stisku tlačítka *Načti* producent vytvoří povel *LoadData* a konzument pak pomocí „LoadDataBrow.vi“ tuto úlohu vykoná.

Pro Prohlížecí režim byl opět použit, tentokrát vnořený, Tab Control. Tedy na jedné stránce (*Browser*) je vložen další Tab Control. Není totiž vhodné mít rozmístěno příliš mnoho prvků na jedné obrazovce. Proto bylo vhodné prvky rozdělit na 2 stránky. Přidat další stránky je možné a dá se tak snadno koncept dále rozpracovávat.

První stránka je vidět na obrázku 3.13. Obsahuje základní údaje, jako je například frekvence objektu, snímek pracoviště načtený ze souboru, volba zobrazované veličiny atd. Druhá stránka slouží pro rozšířené funkce. Mezi tyto funkce patří spuštění animace, export animace v podobě série obrázků a další údaje, viz obrázek 3.14.

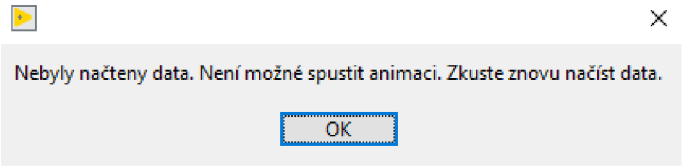


Obr. 3.13: Základní stránka Prohlížecího režimu



Obr. 3.14: Stránka s rozšířenými parametry Prohlížecího režimu

Program řeší i různé chybové stavy, jako je třeba zadání hodnot mimo rozsah, chybějící soubory ve složce atd. Na obr. 3.15 je například upozornění, kdy uživatel chtěl spustit animaci, i když neměl načtené soubory. K načítání souborů bych doplnil ještě jednu poznámku. V podkapitole 3.5.1 bylo zmíněno, že původní program exportoval data ve stopách, místo metrů. Kvůli zpětné kompatibilitě může uživatel nahrát i soubor s imperialistickými jednotkami a program jej převede do jednotek SI. Uživatel tuto možnost může vybrat pomocí tlačítka „Data obsahují ft?“.



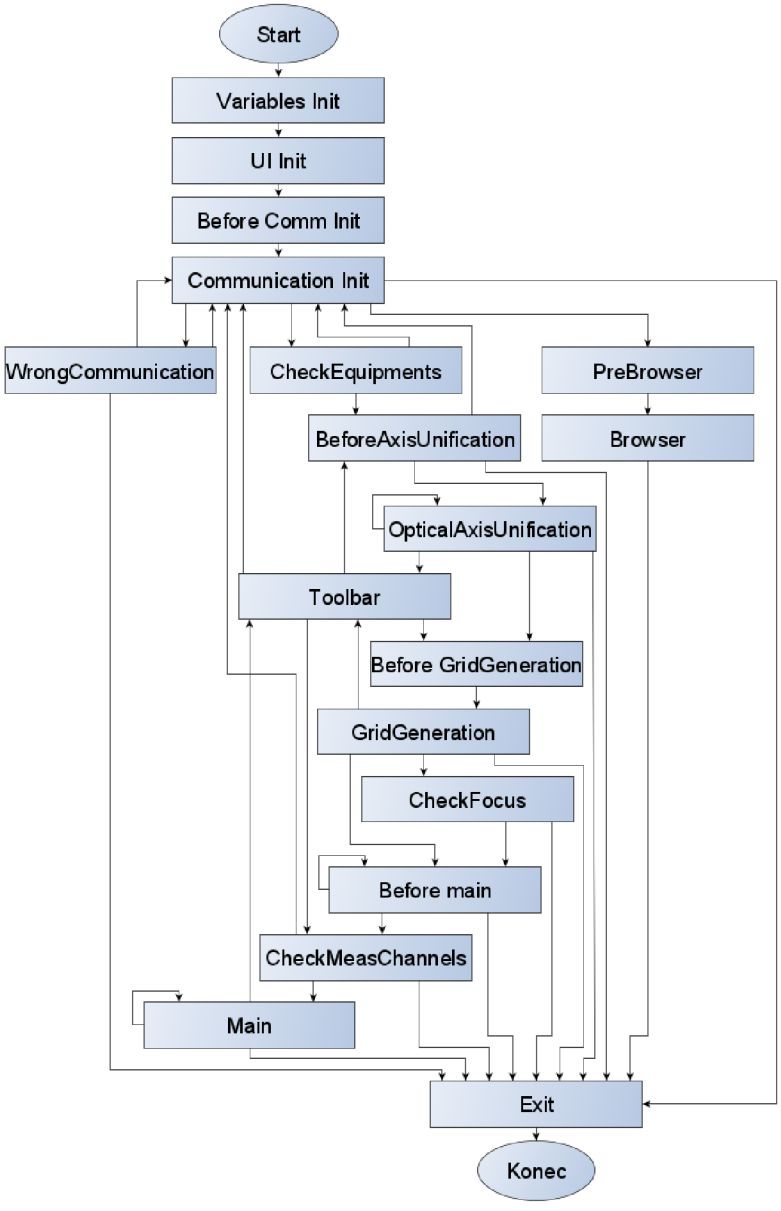
Obr. 3.15: Informační hlášení při spuštění animace bez vstupních dat

3.9 Výsledná konfigurace

Ve výše uvedených podkapitolách byly zmíněný úpravy a vylepšení původního programu. Pro úplnost jsou níže uvedeny parametry finálního řešení. Tím je soupis potřebných softwarových komponent a výsledný stavový automat (obr. 3.16).

Potřebné softwarové komponenty:

- operační systém Windows 10 (64-bit s českým jazykovým balíčkem),
- LabVIEW 2020,
- NI Sound and Vibration 2020,
- Vision Development Module 2020,
- Vision Acquisition Software 20.0 (IMAQdx),
- OpenG toolkit (dostupný skrze VI Packet Manager).



Obr. 3.16: Výsledný stavový automat programu

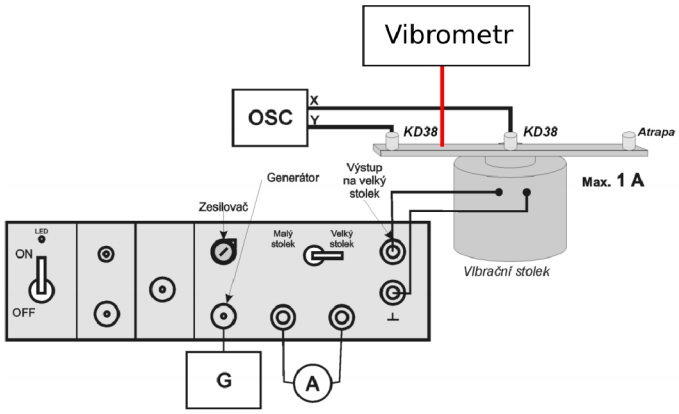
4 Kontrolní měření

V rámci kapitoly 3 proběhla řada úprav. Tyto úpravy většinou probíhaly v Demo režimu. V něm nebylo možné ověřit, zda řešení fungují i v běžném režimu. Zároveň nebyla jiná možnost jak zkontrolovat, že původní funkcionalita byla zachována a nebyla poškozena úpravami. Kvůli výše uvedeným důvodům bylo nutné provést kontrolní měření. Hlavní cílem nebylo samotné měření, ale validace správného chování a dosazení podobných výsledků jako původní verze.

4.1 Příprava pracoviště

Pro kontrolní měření byla zvolena úloha měření vibrací nosníku. Před zahájením samotného měření bylo potřeba nejdříve celé pracoviště zapojit. Přístroje byly zapojeny dle obrázku 4.1. Pomocí generátoru byl nastavený harmonický signál, který byl pomocí zesilovače přiveden na vstup vibračního stolku. Změnou frekvence generátoru jsem hledal frekvenci, kdy nosník začal rezonovat.

Podmínky měření jsou v tabulce 4.1. Seznam použitých přístrojů je pak v tabulce 4.2. Výsledné pracoviště se všemi přístroji je pak vidět na obr 4.1. Kde jsou části vibrometru i měřené úlohy [19].



Obr. 4.1: Schéma zapojení pro měření rezonančních kmitočtů nosníku, převzato a upraveno z [19]

4.2 Průběh měření

Po nastavení pracoviště bylo potřeba připravit program pro měření. Prvním krokem bylo vybrat a nastavit všechny periférie v inicializaci programu. Druhým krokem bylo provést sesouhlasení 4 bodů a zvolení vhodného fokusu laseru. V dalším kroku bylo

Tab. 4.1: Podmínky kontrolního měření

Datum:	18.3.2021
Teplota:	24,9 °C
Vlhkost:	15 %

Tab. 4.2: Seznam přístrojů pro kontrolní měření

Přístroj	Identifikátor
Kontrolér Polytec OFV-5000	S/N: 0101998
Laserová hlava Polytec OFV-505	S/N: 0101999
Vibrační stolek, typ 11077	SAP: 1000054191
Osciloskop Siglent SDS 1102X+	SAP:00100283964-0000
Generátor Picotest G5100A	inventární číslo: 6S25/023
Multimetr UNI-T UT804	inventární číslo: 6S24/019
Akcelerometr KD38 (ve středu nosníku)	S/N: 6305
Akcelerometr KD38 (na konci nosníku)	S/N: 6302
Měřicí karta NI cDAQ™-9174	S/N: 17D9876
Měřicí karta NI 9234	S/N: 1434F72
Měřicí karta NI 9263	S/N: 168AF53
Zesilovač	inventární číslo: E1175
Rozmítací systém Thorlabs GVS-012 (zrcátka)	S/N: TSH34441/TSH34442
PC	SAP:001000283062-0000

potřeba ověřit, zda sesouhlasení bylo provedeno správně. To jsem provedl tak, že jsem v obrazu kamery vybral bod a vizuálně jsem ověřil, že bod na obrazovce i v obraze na sobě leží. Po ověření sesouhlasení jsem zvolil mřížku 3 x 30 (řádek x sloupec) na měřeném nosníku. Tuto mřížku jsem exportoval a uložil, abych ji mohl opakovaně použít pro zbývající měření.

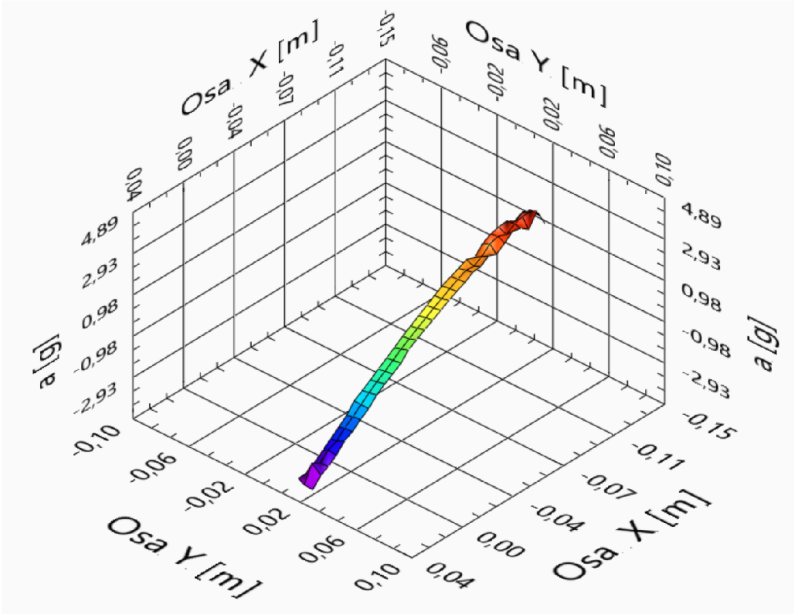
Poté jsem zahájil měření pro nižší frekvenci nosníku 35 Hz. Po dokončení měření jsem vypnul novější verzi programu a zapnul původní verzi. Naimportoval jsem konfiguraci včetně mřížky z předchozího měření, aby byly parametry identické. Po skončení programu jsem naměřená data u původní verze ručně nakopíroval do složky po vzoru nového řešení. Viz podkapitola 3.5.

Obdobným postup jsem udělal u vyšší frekvence nosníku 204 Hz. Měření proběhlo prvně novější verzí a poté původní. Data z původní verze programu bylo potřeba ručně sjednotit s názvy nové verze, aby mohlo dojít k importu dat do Prohlížecího režimu.

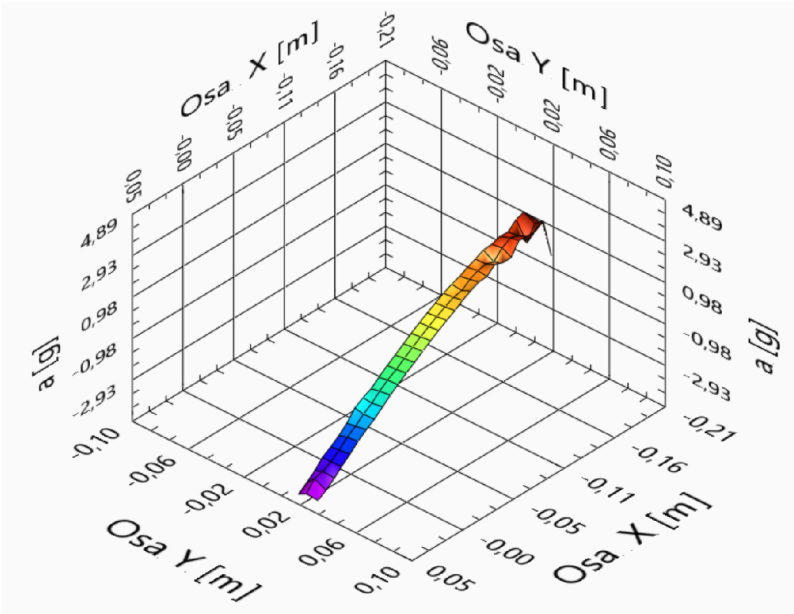
Obr. 4.2: Pracoviště kontrolního měření

4.3 Výsledky

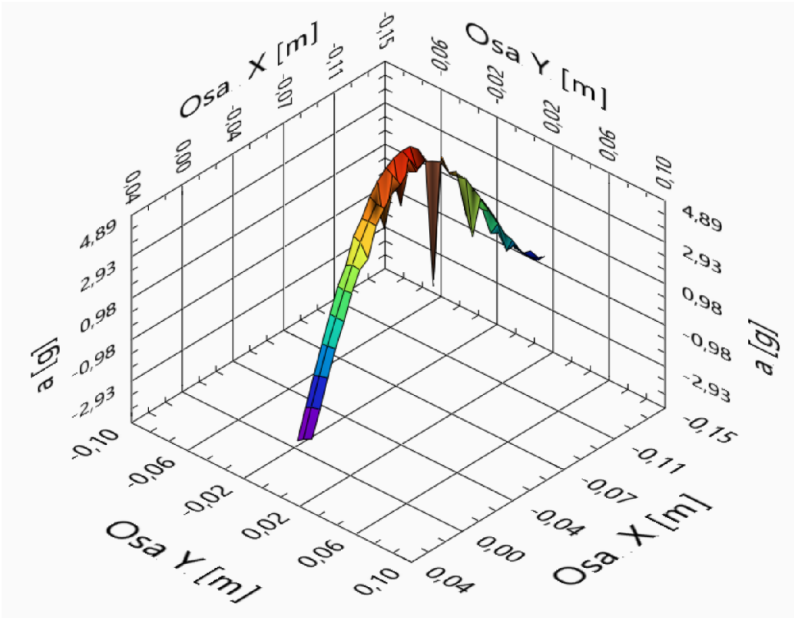
Po získaní dat z měření je bylo třeba vyhodnotit. K tomuto účelu byl využit Prohlížecí režim. V něm byla data jednotlivých měření postupně načtena. Výsledky první sady měření pro frekvenci 35 Hz jsou na obrázcích 4.3 a 4.4. Výsledky druhé sady měření pro frekvenci 204 Hz je vidět na obrázcích 4.5 a 4.6. Při jejich porovnání lze konstatovat, že jsou prakticky identické a tedy integrace nových funkcí neovlivnila výsledné měření. Data jsou nepatrně jiná, protože fokus nebyl nastavený pro všechna měření stejně. Během kontrolního měření byly testovány i nově implementované části programu a nebyly nalezeny žádné zjevné nedostatky.



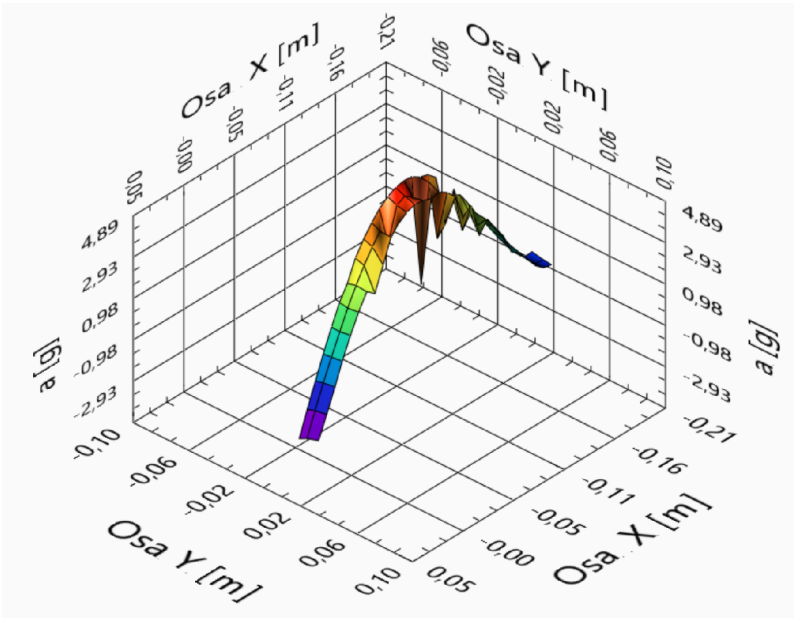
Obr. 4.3: Výsledky měření nosníku při rezonanční frekvenci 35 Hz (původní verze programu)



Obr. 4.4: Výsledky měření nosníku při rezonanční frekvenci 35 Hz (nová verze programu)



Obr. 4.5: Výsledky měření nosníku při rezonanční frekvenci 204 Hz (původní verze programu)



Obr. 4.6: Výsledky měření nosníku při rezonanční frekvenci 204 Hz (nová verze programu)

5 Náměty pro další rozvoj

S postupujícím časem mě napadaly nová vylepšení nebo zlepšení. Postupně jsem tato vylepšení implementoval do původního projektu. V rámci omezeného času a vlivem jiných omezení nebylo možné všechny nápady realizovat. Realizovány byly ty nejdůležitější ve shodě s vedoucím práce.

Vidím přidanou hodnotu v tom uvést možný směr pro další rozvoj. Níže v podkapitolách uvádím hlavní myšlenku nové funkcionality a představení prvních kroků pro její realizaci. Díky tomu mohou být tyto nápady podnětem pro další rozvoj.

5.1 Softwarový autofokus

V současném projektu je možné měnit ostření fokusu buď ručně nebo automaticky. Při automatickém fokusu program pouze zašle příkaz kontroléru, který provede fokusaci sám. Tento způsob je ovšem velmi pomalý. Kontrolér hledá nejsilnější signál v celém rozsahu a tak tato fokusace někdy trvá i minutu. Při větším množství bodů pak program provádí ostření řádově i desítky minut. Proto je snaha tento způsob nahradit.

Softwarový fokus je součástí původního projektu, ale nebyl použit, jelikož je u něj poznámka, že nefunguje korektně. O pokusech realizace softwarového autofokusu se Ing. Tomek zmiňuje i ve své bakalářské práci [2]. Ten je použit v SubVI *PSV_5000_FGV.vi*. V něm byl použit i režim „FocusInRange“. Zmíněný režim však během kontrolního měření nefungoval a tak byl v projektu deaktivován na původní způsob.

Z výše zmíněných důvodů by bylo vhodné podrobněji proměřit fokusaci v závislosti na vzdálenosti a stálost i sílu signálu laserové hlavy. Díky ní bude možné vhodněji určit rozmezí vhodné fokusace při často používaných vzdálenostech. Program by tak nemusel prohledávat nejsilnější signál v celém rozsahu, ale pouze v předdefinovaném intervalu. Nakonec pak tento režim vložit do původního SubVI a integrovat do celého projektu.

5.2 Podpora anglické lokalizace systému

V podkapitole 2.8 je uveden problém s lokalizací. Program má uživatelské rozhraní částečně česky a částečně anglicky. Byla otázka, do jakého jazyka práci vlastně lokalizovat. Ve výše zmíněné podkapitole je uvedeno, že LabVIEW má problém s diakritikou, pokud operační systém neobsahuje český jazykový balíček. Naproti tomu má angličtina daleko lepší podporu.

Druhou variantou je mít program dvojjazyčný a mít tak verzi programu v anglickém i českém jazyce. LabVIEW podporuje export všech popisů a dialogů. Háček je ovšem v tom, že součástí exportu nejsou popisy a dialogy vnořených SubVI. Tedy v případě tohoto projektu by to znamenalo prakticky exportoval a přeložit textové řetězce téměř všech použitých SubVI. To je časově velmi náročné. Navíc import probíhá stejným způsobem. Automatické přepínání je možné, ale nejedná se o triviální řešení [20], [21].

Dalším aspektem je samotný jazyk. Většina slov má jinou délku v češtině a jinou v angličtině. V důsledku toho nestačí pouze přeložit text, ale je potřeba potom změnit rozložení dalších prvků, aby se texty korektně zobrazovaly a nepřekrývaly se s jiným objekty.

Jak jsem dříve uvedl, při špatně zobrazené diakritice stačí pouze doinstalovat český jazykový balíček pro operační systém. Samotná lokalizace programu do anglického jazyka je poměrně pracná záležitost, protože nestačí pouze hromadně exportovat názvy a přeložit je. Nutné je ještě zkontrolovat zda-li se všechny prvky korektně zobrazují. Rozhraní je z devadesáti procent v češtině. Proto mi přišlo jednodušší přeložit pouze zbytek do českého jazyka, aby byla práce jednotná. Anglická lokalizace by byla výhodou, nikoliv však podmínkou nutnou.

5.3 Prohlížecí režim

Program nyní obsahuje Prohlížecí režim, kde jsou základní údaje a možnosti. Přesto tento režim byl programován tak, aby bylo možné jej v budoucnu jednoduše rozšiřovat. Jednotlivé prvky jsou v Tab Controlu a lze tedy jednoduše přidat další okna s nabídkou.

Jedna z možných funkcí je podpora výběru bodu pomocí myši. Na okraji obrazovky je fotka s body. Uživatel by tak mohl kliknout na vybraný bod na obrázku a zobrazit si podrobnější jeho naměřená data. V současné podobě se do programu nahrává obrázek s vyznačenými body. Data z měření obsahují i obrázek bez bodů. Je tedy možné importovat základní obrázek a body nad tímto obrázkem vykreslit.

Dále program dokáže vykreslovat animace pouze pro body, které jsou orientované do čtvercové nebo obdélníkové matice. V případě výběru bodů ve tvaru například elipsy nedojde ke korektnímu zobrazení. Pro podporu dalších tvarů je potřeba rozšířit SubVI „DataCalc.vi“.

Další vlastnosti a parametry je možné přidat. Vše záleží zejména na potřebách uživatele.

5.4 Simulace meřicí úlohy

Současný projekt obsahuje Demo režim, ve kterém je možné si vyzkoušet obsluhu programu. Přidaný benefit by bylo nejen si program vyzkoušet, ale vyzkoušet si i zkušební měření.

Pro tento úkol je potřeba detailně proměřit měřenou úlohu, která se bude simulovat. Například vibrace nosníku z kontrolního měření. Z naměřených dat je pak nutné vytvořit matematický model. Pro tento model by bylo dobré vytvořit SubVI.

Do programu by bylo potřeba poté doplnit „virtuální paprsek“, který by simuloval pohyb skutečného laseru. Při měření v Demo režimu by výše uvedené SubVI generovalo data do nadřazeného systému. A program by se tak k uživateli tvářil, jak kdyby skutečně něco měřil.

5.5 Práce s perifériemi

Projekt pracuje až s 4 měřicími kanály u karty NI 9234. V současnosti je toto řešení postačující. V rámci budoucího rozvoje by bylo dobré současné řešení pro práci s kartami více zobecnit.

Zobecnění by znamenalo snadnější rozšíření v případě více karet. Dále by to však znamenalo i přepracování grafického rozhraní u inicializace periférií. Viz obrázek 3.1. Bylo by potřeba vymyslet rozhraní tak, aby se do něj dynamicky daly přidávat další kanály. Variantou by mohlo být samostatné okno pro výběr kanálů. Kde by se zobrazovaly právě dostupné kanály.

5.6 Servisní režim

Při vývoji programu je úzká vazba mezi hardwarem a softwarem. Softwarové chyby lze pomocí různých nástrojů nalézt distančně. Nalezení chyby hardwaru distančně není příliš jednoduché.

Napadlo mě proto vytvořit Servisní režim, který by sám otestoval všechny důležité periférie a dokázal ověřit funkčnost. Díky tomu by bylo jednodušší odhalit chyby a zjistit, zda je původ v SW nebo HW.

V rámci stavového automatu by se přidaly dva nové stavy. Jeden by sloužil pro přepnutí obrazovky programu a případné inicializace. Druhý stav by pak vykonal samotný test. Tento test by v základu měl obkroužit pomocí zrcátek a laseru svou maximální výchylku, tím by se otestoval mechanismus zrcátek.

Dále by pak proběhla kontrola komunikace s kontrolérem. Zároveň by se zkušebně měnil fokus v celém rozsahu. Tím by se otestovala práce s kontrolérem a hlavou. Případně by se na obrazovce vizualizovala aktuálně změřená hodnota signálu.

Obdobně by proběhl i test měřicích karet. Servisní režim by tak značně pomohl při vývoji a údržbě celého zařízení. Není však nezbytný.

5.7 Automatické sesouhlasení

V současné podobě sesouhlasení dělá uživatel kompletně sám. Vize je, aby sesouhlasení mohl přístroj udělat částečně sám. Uživatel by vybral bod na obrazovce, který chce kalibrovat. Zapnul by „automatické sesouhlasení“ a program by se snažil přivést paprsek laseru na zvolené místo. Uživatel by vždycky vybral bod a program by sám laserový paprsek do tohoto místa navedl.

Vytvoření takové funkce má však mnoho úskalí. Pokud sesouhlasení nemá přibližně správou transformační matici, laser je často v poloze mimo zorné pole kamery a není tak v obraze vidět. Pokud by algoritmus čistě hledal paprsek na obraze, nenašel by ho.

Dalším úskalím je detekce samotného paprsku. Výhodou je, že tento objekt je jasně červený. Lze jej v obraze hledat jako červený objekt. Potíž ovšem nastává, pokud na scéně budou další kulaté objekty a zároveň na scéně budou červené body.

Nejnáročnější patrně bude navrhnout a odladit algoritmus pro přemístění laseru. Ten by detekoval laser v obraze a přesunul na místo vybraného bodu v obraze. Reakce zrcátek je skutečně rychlá, ale stále zde dochází k časovému zpoždění. Nemusí být úplně jednoduché algoritmus navrhnout tak, aby paprsek dopravil do „cílového místa“ v krátkém časovém úseku a zároveň nerozkmital soustavu.

5.8 Doporučení vhodné vzdálenosti

Výše v podkapitolách (např. 1.1.2) bylo uvedeno, že vhodně zvolená vzdálenost má vliv na kvalitu signálu. Dle rovnice 1.2 lze vypočítat vhodnou vzdálenost od objektu a tím zlepšit výsledky.

Pokud uživatel chce najít vhodnou vzdálenost, musí si ji v současnosti vypočítat. Vhodné by bylo přidat algoritmus pro výpočet nejvhodnější vzdálenosti. Uživatel by například v inicializačním okně zadal současnou vzdálenost a program by mu hned navrhnul vhodnější.

Závěr

Tato diplomová práce navazuje na předchozí práci Ing. Tomka [1], která byla nadále rozvíjena akademickými pracovníky Ústavu automatizace a měřicí techniky. Ta se věnovala vytvoření laserového 2D vibrometru. Nepracoval jsem s verzí programu, která byla publikována ve zmíněné práci, ale s poslední interní nepublikovanou verzí. Cílem mé práce bylo vylepšit stávající řešení. K tomu bylo zapotřebí zanalyzovat současný stav. Jednak z hardwarového, tak i softwarového pohledu.

Ve své práci jsem tuto analýzu rozdělil na dvě hlavní kapitoly. První kapitola 1 se věnuje popisu celého systému. Jak z softwarového, tak i hardwarového pohledu. Obsahuje soupis všech komponent, které jsou k zprovoznění původního řešení zapotřebí. Včetně obecné architektury programu.

Další kapitola 2 se věnuje analýze slabých míst původního programu. Vzhledem k mimořádné situaci s epidemii COVID-19 jsem se snažil hledat takové nedostatky a vylepšení, která jsem schopen odhalit distanční formou. Přesto jsem zde uvedl i některé body, které nelze provádět distančně. Navrhoval jsem zde například zjednodušení, uživatelského rozhraní. Nahrazení MathScript Nodes nebo přidání Prohlížecího režimu.

Hlavní částí práce je kapitola 3. Ta se věnuje realizaci slabých míst vyjmenovaných v předchozí kapitole 2. U jednotlivých bodů je řečeno, co mě vedlo k tomu se mu věnovat, a postup u realizace. Řešení je zde popsáno stručněji, protože bez podrobné znalosti programu se v architektuře programu i řešení lze snadno ztratí. Na konci kapitoly je soupis potřebných komponent pro softwarové vybavení 2D vibrometru. Stěžejní body se podařilo úspěšně realizovat.

Uživatel nyní může přesouvat body (zejména u sesouhlasení) pomocí myši. Je možné program používat bez připojení periférií pro seznámení s programem nebo ladění. Byly odstraněny hlavní chyby z původního programu (nedokončené historické změny). Přepracované ukládání dat a přidání Prohlížecího režimu pro jejich čtení. Výpočet transformační matice při sesouhlasení je nyní přepracován do verze bez MathScript Nodes. A mnoho dalších užitečných vylepšení.

Jednotlivé nové funkcionality a opravy byly vytvářeny v Demo režimu, kdy program nepracuje s periférierni. Nebylo tak možné ověřit, zda řešení fungovalo správně i v běžném režimu. Zejména z tohoto výše uvedeného důvodu vznikla kapitola 4 nazvaná „Kontrolní měření“. Zde bylo hlavním úkolem ověřit, že jednotlivé nové funkcionality byly správně implementované i mimo Demo režim. Závěrem je, že zařízení funguje správně v Demo i běžném režimu. Ověření proběhlo úspěšně a jeho výsledky uvedeny v této kapitole.

Během práce mě průběžně napadaly různé další nápady na zlepšení funkcionality programu. Nebylo však možné všechny nápady stihnout realizovat v rámci diplomové

práce. Po konzultaci s vedoucím jsem realizoval ty klíčové. Zbylé nápady a první kroky pro realizaci jsem sepsal v poslední kapitole 5. Věřím, že se jedná o užitečné podněty pro základ dalších bakalářských nebo diplomových prací.

Nová přepracovaná verze programu oproti původní zlepšuje uživatelský komfort, zejména u provádění sesouhlasení. Podporuje Demo režim, kdy program funguje i bez periférií. Odstraňuje MathScript Nodes z projektu. Je v ní přepracovaný export dat, který je nyní jednotný a obsahuje všechny důležité údaje. Včetně obrázku měření. Obsahuje Prohlížecí režim, který umí automaticky zpracovat data a přehledně je interpretovat uživateli. Odstraňuje chyby v původním projektu. A další funkcionality, které ulehčují práci s programem a přidávají nové možnosti.

Literatura

[1] TOMEK, Tomáš. *Laserový 2D skener*. Brno, 2016, 89 s. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace?zp_id=94471. Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce Ing. Petr Beneš, Ph.D.

[2] TOMEK, Tomáš. *Bezkontaktní měření provozních tvarů kmitů*. Brno, 2014. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace?zp_id=73333. Bakalářská práce. Vysoké učení technické v Brně. Vedoucí práce Doc. Ing. Petr Beneš, Ph.D.

[3] VYBÍRAL, Ondřej. *Analýza slabých stránek laserového 2D vibrometru a jeho vylepšení*. Brno, 2020. Semetrální práce. Vysoké učení technické v Brně. Vedoucí práce Ing. Stanislav Pikula, Ph.D.

[4] POLYTEC. *Vibrometer Single Point Sensor Head OFV-505/-503: User Manual*. Waldbronn, Německo, 2004.

[5] GVS012/M - 2D Large Beam (10 mm): Diameter Galvo System, Silver-Coated Mirrors, Metric. *Thorlabs* [online]. 2010 [cit. 2020-10-24]. Dostupné z: https://www.thorlabs.com/thorproduct.cfm?partnumber=GVS012/M

[6] POLYTEC. *Vibrometer Controller OFV-5000: User Manual*. Waldbronn, Německo, 2004.

[7] NI 9223 Datasheet. *National Instruments* [online]. 2016, 12 [cit. 2020-10-17]. Dostupné z: http://www.ni.com/pdf/manuals/374223a_02.pdf

[8] NI 9234: Datasheet. *National Instruments* [online]. 2015 [cit. 2021-03-23]. Dostupné z: https://www.ni.com/pdf/manuals/374238a_02.pdf

[9] NI 9263: Datasheet. *National Instruments* [online]. 2016, 12 [cit. 2020-10-17]. Dostupné z: http://www.ni.com/pdf/manuals/373781b_02.pdf

[10] NI cDAQ-9174: Specifications NI CompactDAQ Four-Slot USB Chassis. *National Instruments* [online]. 2013, 10 [cit. 2020-10-17]. Dostupné z: https://www.ni.com/pdf/manuals/374045a.pdf

[11] C920 Technical Specifications. *Logitech* [online]. [cit. 2020-11-04]. Dostupné z: https://support.logi.com/hc/en-us/articles/360023307294-C920-Technical-Specifications

[12] LOGITECH® HD PRO WEBCAM C920. *Logitech* [online]. 2 [cit. 2020-11-04]. Dostupné z: https://bit.ly/2RYb8C8

[13] LabVIEW. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-, 2020 [cit. 2020-10-24]. Dostupné z: https://en.wikipedia.org/wiki/LabVIEW

[14] Vládní usnesení související s bojem proti epidemii koronaviru. *Vláda České republiky* [online]. 20.11.2020 [cit. 2020-12-01]. Dostupné z: https://www.vlada.cz/cz/epidemie-koronaviru/dulezite-informace/prehled-vladnich-usneseni-od-vyhlaseni-nouzoveho-stavu-180608/#zari

[15] Vláda na dva týdny pro studenty zcela uzavírá vysoké školy. *Vysoké učení technické v Brně* [online]. 8.10.2020 [cit. 2020-12-01]. Dostupné z: https://www.vutbr.cz/vut/aktuality-f19528/vlada-na-dva-tydny-pro-studenty-zcela-uzavira-vysoke-skoly-d204068

[16] Moving to MATLAB Script Nodes from MathScript Nodes in LabVIEW — Benchmark Examples. *National Instruments* [online]. 2020, 25.10.2019 [cit. 2020-12-22]. Dostupné z: https://forums.ni.com/t5/Example-Code/Moving-to-MATLAB-Script-Nodes-from-MathScript-Nodes-in-LabVIEW/ta-p/3980734?profile.language=en

[17] Jazykové sady pro Windows. *Microsoft* [online]. [cit. 2021-03-28]. Dostupné z: https://support.microsoft.com/cs-cz/windows/jazykové-sady-pro-windows-a5094319-a92d-18de-5b53-1cfc697cfca8#ID0EBBD=Windows_10

[18] PECL, Richard. *Základní metodické postupy při tvorbě uživatelského rozhraní.* Praha, 2006, 62 s. Dostupné také z: https://is.cuni.cz/webapps/zzp/detail/27012/?lang=en. Bakalářská práce. Univerzita Karlova. Vedoucí práce Doc. PhDr. Richard Papík, Ph.D.

[19] BENEŠ, Petr. *Měření fyzikálních veličin: návody do laboratorních cvičení.* Brno, 2018.

[20] Localizing Your LabVIEW Application to Different Languages. *National Instruments* [online]. 2020 [cit. 2021-03-17]. Dostupné z: https://bit.ly/3qOAo9O

[21] Exporting and Importing VI Strings. *National Instruments* [online]. 2018 [cit. 2021-03-17]. Dostupné z: https://zone.ni.com/reference/en-XX/help/371361R-01/lvhowto/exprt_imprt_vi_strings/

A Obsah přiloženého CD

Přiložené CD obsahuje:

- digitální verze diplomové práce,
- zkompilovaná verze programu.