VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020

Bc. Martin Belica

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

VÝVOJ UNIVERZÁLNÍHO SOFTWAROVÉHO ROZHRANÍ PRO DETEKČNÍ JEDNOTKY V OPTICKÉ SPEKTROSKOPII

DEVELOPMENT OF UNIVERSAL SOFTWARE INTERFACE FOR DETECTION UNITS IN OPTICAL SPECTROSCOPY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Martin Belica**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. Luděk Žalud, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Martin Belica ***ID:*** 175040
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Vývoj univerzálního softwarového rozhraní pro detekční jednotky v optické spektroskopii

POKYNY PRO VYPRACOVÁNÍ:

1. Nastudujte problematiku instrumentace používané ve spektroskopii laserem buzeného plazmatu (angl. zkratka LIBS).
2. Prostudujte problematiku knihovny QT pro vytvoření grafického SW rozhraní.
3. Navrhněte řešení zadané problematiky.
4. Naprogramujte univerzální SW rozhraní pro vyčítání a zobrazovaní dat z optických spektroskopů podle návrhu v bodě 3.
5. Otestujte a zdokumentujte systém v reálném provozu.

DOPORUČENÁ LITERATURA:

[1] Lee Zhi Eng; Qt5 C++ GUI Programming Cookbook: 978-1783280278 Hahn and Omenetto: 10.1366/11-06574 Guillaume Lazar, Robin Penea; Mastering Qt 5: 978-1786467126

[2] HAHN, David W. a Nicoló OMENETTO. Laser-Induced Breakdown Spectroscopy (LIBS), Part I: Review of Basic Diagnostics and Plasma—Particle Interactions. Applied Spectroscopy [online]. 2010, 64(12), 335A-336A [cit. 2019-09-24]. DOI: 10.1366/000370210793561691. ISSN 0003-7028. Dostupné z: http://journals.sagepub.com/doi/10.1366/000370210793561691

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: prof. Ing. Luděk Žalud, Ph.D.
Konzultant: doc. Ing. Petr Číka, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Táto diplomová práca sa zaoberá návrhom a implementáciou univerzálneho užívateľského rozhrania pre detekčné jednotky v laserovej spektorskopii. Návrh a implementácia riešenia vychádza z analýzy súčasného stavu inštrumentácie používanej v oblasti laserovej spektroskopie. Užívateľské rozhranie musí byť schopné pracovať s vysokými opakovacími frekvenciami merania. Zachytené spektrá musia byť vhodným spôsobom užívateľovi zobrazené a taktiež je potrebné ich vhodným spôsobom uložiť na pevný disk. Okrem spomínaných hlavných vlastností, musí byť výsledná aplikácia univerzálna. To znamená, že musí umožniť použitie rôznych typov a značiek detekčných zariadení.

KĽÚČOVÉ SLOVÁ

LIBS, Spektroskopia laserom budenej plazmy, C++, Qt, QML, Softvérovo definované zariadenie, spektrum, grafické užívateľské rozhranie

ABSTRACT

This master's thesis deals with the design and implementation of universal user interface for detection devices used in Laser-induced breakdown spectroscopy. The design and implementation are based on analysis of current state of instrumentation used in laser spectroscopy. The user interface should be able to work with high repetition frequency of measurement. Acquired spectra should be visualised to user and it is also necessary to save this data on hard drive. The resulting application must be universal. It means the application must support various types and vendors of detection devices.

KEYWORDS

LIBS, Laser-induced breakdown spectroscopy, C++, Qt, QML, Software Defined Device, spectra, GUI

BELICA, Martin. *Vytvoření univerzálního softwarového rozhraní pro detekční jednotky v optické spektroskopii*. Brno, Rok, 66 s. Diplomová práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: prof. Ing. Luděk Žalud, Ph.D.

VYHLÁSENIE

Vyhlasujem, že svoju diplomovú prácu na tému „Vytvoření univerzálního softwarového rozhraní pro detekční jednotky v optické spektroskopii“ som vypracoval samostatne pod vedením vedúceho diplomovej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb. o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon) v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Ďakujem môjmu vedúcemu práce, pánovi prof. Ing. Luďkovi Žaludovi Ph.D., za vedenie tejto práce, za pomoc a cenné rady, ktoré mi poskytol pri jej tvorbe. Ďalej by som sa rád poďakoval všetkým svojim kolegom z tímu laserovej spektroskopie, hlavne Bc. Patrikovi Cebovi a Bc. Ondrejovi Beňušovi.

Obsah

Zoznam obrázkov

Zoznam výpisov

Úvod

Skúmanie obsahu chemických prvkov v rôznych materiáloch je veľmi dôležitou súčasťou mnohých vedných a priemyselných odvetví. Existuje množstvo spôsobov, ako určiť prvkové zloženie skúmaného predmetu. Jednou z metód je spektroskopia laserom indukovanej plazmy (Laser-Induced Breakdown Spectroscopy) alebo skrátene LIBS. V prípade technológie LIBS sa jedná o moderný spôsob materiálovej analýzy, pričom je možné rýchlo a bezpečne analyzovať vzorky vo všetkých skupenstvách.

Moderná doba so sebou prináša pokrok vo vede a technike a ani v prípade laserovej spektroskopie sa nejedná o výnimku. Aby bolo možné nasadiť tento typ spektroskopie do širokej škály priemyselného a vedeckého využitia, je potrebné splniť isté požiadavky. Jednou z požiadaviek (ktorou sa bude táto práca zaoberať) je zvyšovanie opakovacej frekvencie meraní. Vysokofrekvenčná analýza je nevyhnutná v aplikáciách ako sú napríklad triediace linky, určovanie inklúzie v metalurgii alebo toxikológia v medicíne. Vysoká frekvencia meraní však so sebou prináša aj isté problémy resp. potreby, ktoré je potrebné vyriešiť. Hlavným problémom je veľké množstvo dát, ktoré je nutné v relatívne krátkej dobe spracovať a následne ich užívateľovi zobraziť, príp. niekam uložiť.

Po analýze potrieb na zvyšovanie rýchlosti je ďalej vhodné zvážiť, aké technológie zvoliť a použiť na implementáciu riešenia. Výsledné riešenie by malo byť vo forme univerzálneho softvérového rozhrania, ktoré umožní vizualizovať a ukladať dáta z vysokofrekvenčnej analýzy. Navyše by výsledná aplikácia mala zvládať opakovaciu frekvenciu merania aspoň $50\,Hz$.

Naštudovaním zvolených technológií a problematiky LIBS je možné vytvoriť finálny návrh aplikácie. Návrh by mal pozostávať zo štyroch základných častí. Prvou časťou je spôsob komunikácie medzi počítačom a spektrometrom. Druhou časťou je vizualizácia zachytených dát formou čiarového grafu. Ďalšou časťou je spôsob, akým budú zachytené dáta uložené na pevný disk. Všetky tieto operácie zastrešuje grafické užívateľské rozhranie, ktoré je poslednou časťou návrhu.

Na základe návrhu aplikácie je možné postaviť implementáciu zadanej aplikácie. Vývoj aplikácie by mal teda pozostávať z rovnakých bodov ako jej návrh. Výsledkom implementácie budú dva funkčné programy. Prvým programom bude ovládač, ktorý komunikuje so spektrometrom a odosiela namerané dáta do druhej aplikácie. Druhou aplikáciou bude grafické užívateľské rozhrania, pomocou ktorého bude možné na diaľku ovládať spektrometer a zároveň spracovávať prijaté dáta.

Po úspešnej implementácii, ale aj počas nej, je nevyhnutné aplikáciu testovať. V rámci implementácie sa jedná o testovanie správnosti jednotlivých častí. Na záver, po dokončení vývoja programu je potrebné vykonať testovanie v reálnych podmienkach.

1 Spektroskopia laserom budenej plazmy

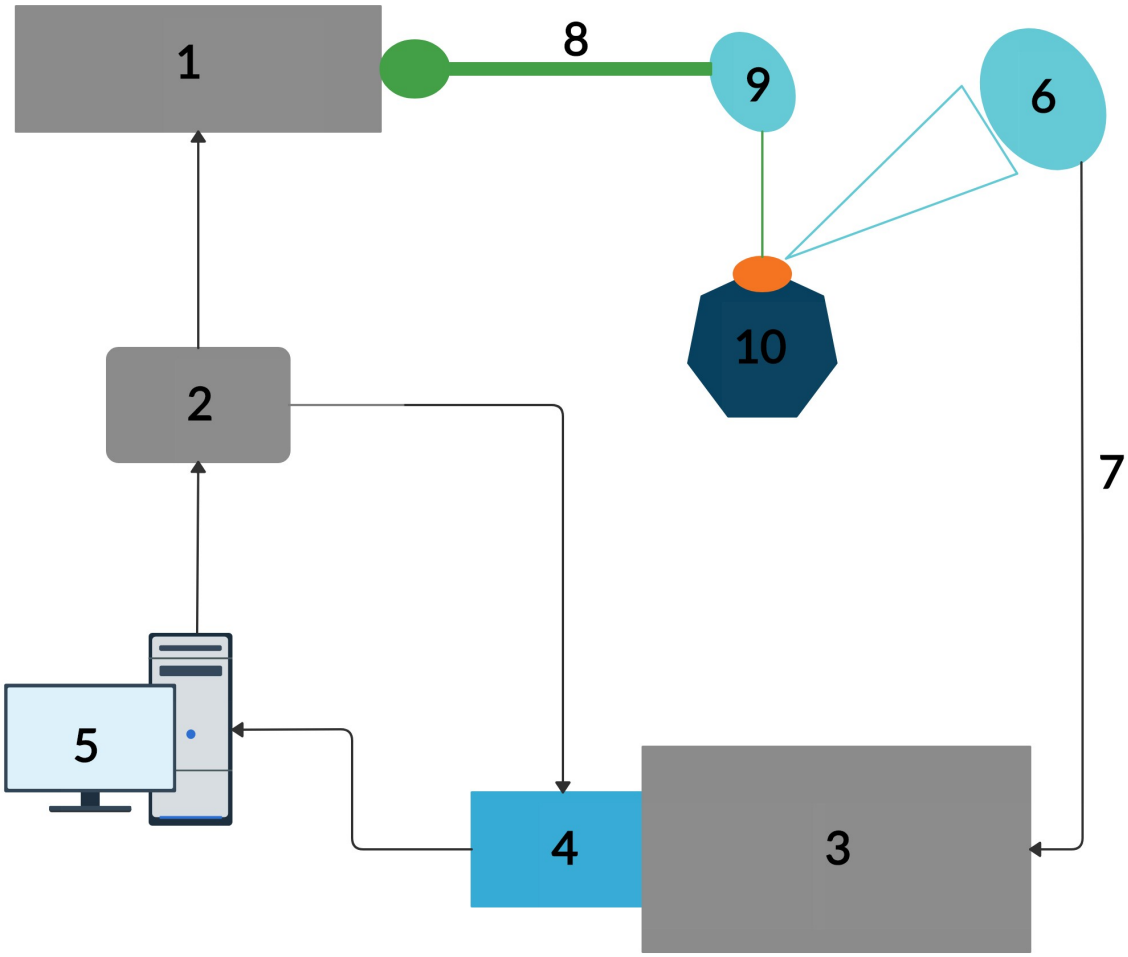
Spektroskopia laserom budenej plazmy alebo skrátene LIBS (Laser Induced Breakdown Spectroscopy) je technológia na analýzu prvkového zloženia materiálov. Skúmaný materiál môže byť pevného, kvapalného alebo plynného skupenstva. Tento typ spektroskopie je založený na princípoch atómovej emisnej spektroskopie (AES). Podstatou atómovej emisnej spektroskopie je prerušenie chemických väzieb v skúmanej vzorke na voľné atómy a ióny. Deštrukcia väzieb sa dosahuje použitím rôznych zdrojov vysokej teploty. Vďaka vysokej teplote dochádza k ionizácii a excitácii atómov do vyšších energiových stavov, čoho výsledkom je plazma, ktorej žiarenie sa zachytáva pomocou detektora [13] [14].

V prípade laserovej spektroskopie sa na vytvorenie plazmy využíva krátky fokusovaný laserový pulz (s dĺžkou trvania typicky v jednotkách ns). Schéma celej zostavy, ktorá je potrebná na skúmanie materiálov pomocou tohto typu spektroskopie, je znázornená na obrázku 1.1.

Spektroskopia LIBS poskytuje rýchlu, jednoduchú, relatívne bezpečnú a *in-situ* (v porovnaní napr. s röntgenovou spektroskopiou) chemickú analýzu skúmaného materiálu s primeranou presnosťou za rozumnú cenu. Pričom nie je nutná špeciálna príprava skúmaných vzoriek (okrem správnej fokusačnej vzdialenosti). Vďaka spomenutým faktorom sa laserová spektroskopia stáva čoraz populárnejšou v rôznych vedeckých a priemyselných odvetviach (napr. geológia, archeológia, metalurgia, forenzné vedy, vesmírne aplikácie a pod.) [1] [5].

1.1 Inštrumentácia v laserovej spektroskopii

Zariadenia potrebné na laserovú spektroskopiu sú znázornené na obrázku 1.1. Jedná sa teda o vysokovýkonné pulzné lasery, ktorých lúč je zaostrený do malého priestoru (typicky v jednotkách μm). Zaostrený laserový pulz ablatuje malé množstvo skúmaného materiálu (plošná hustota žiarenia dosahuje jednotky $GW \cdot cm^{-2}$). Vyžiarená plazma je zachytená pomocou zbernej optiky a následne je prenesená do spektroskopu. Spektroskop pomocou disperzných elementov rozloží zachytené svetlo na jednotlivé farebné zložky, ktoré sú premietnuté na detektor. Detektor určí intenzity jednotlivých vlnových dĺžok a namerané dáta sú prenesené do počítača, ktorý ich ďalej spracováva pomocou analyzačného softvéru. Celý proces je riadený generátorom pulzov, ktorý odštartuje výstrel laseru a s určitým oneskorením vyšle impulz do detektora, ktorý nasníma aktuálne spektrum. Na riešenie tejto práce sú hlavnými komponentami spektroskop a detektor.



Obr. 1.1: Schéma LIBS aparatúry. 1) hlavica laseru, 2) generátor pulzov, 3) spektroskop, 4) detektor, 5) PC s príslušným softvérom, 6) zberná optika, 7) optické vlákno, 8) laserový pulz, 9) fokusačná optika, 10) skúmaná vzorka

1.1.1 Typy spektroskopov

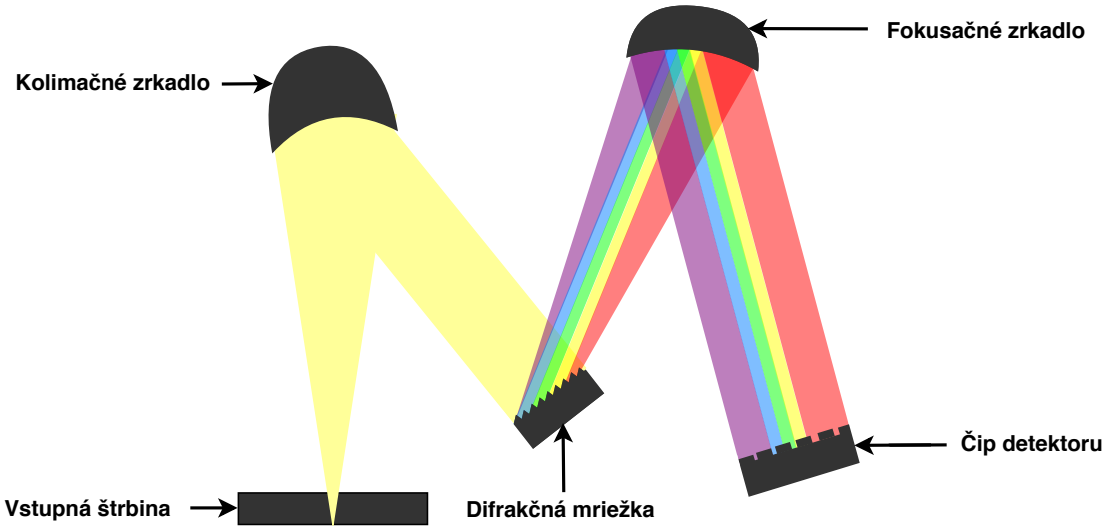
Ako už bolo spomenuté v podkapitole 1.1, spektroskop slúži na rozklad svetla na jednotlivé farebné zložky. V minulosti sa na disperziu svetla využíval optický hranol, ktorý je v súčasnosti nahradený optickou (difrakčnou) mriežkou. Mriežka je tvorená radom úzko rozmiestnených čiar, ktoré sú vyryté alebo vyleptané na povrch zrkadla. Medzi klasické typy patria spektroskopy typu Ebert, Paschen-Runge, Seya-Namioka a Czerny-Turner, ktorého princíp je znázornený na obrázku 1.2.

Hlavnou charakteristikou spektroskopov je ich rozlíšenie R, ktoré je definované nasledovným vzťahom:

$$R = \frac{\lambda}{\Delta\lambda}, \tag{1.1}$$

kde $\Delta\lambda$ je najmenšia vzdialenosť dvoch rozlíšiteľných vlnových dĺžok v okolí vlnovej

dĺžky λ.



Obr. 1.2: Vnútorné usporiadanie spektroskopu typu Czerny-Turner

Ďalšou charakteristikou je uhlová disperzia D definovaná nasledovne:

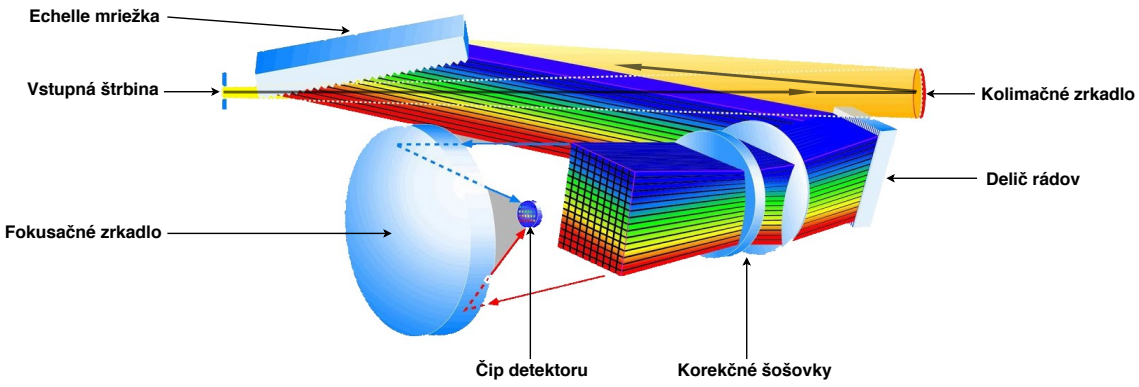
$$D = \frac{d\delta}{d\lambda}, \tag{1.2}$$

kde $d\delta$ je uhol, ktorý zvierajú lúče s rozdielom medzi susednými vlnovými dĺžkami $d\lambda$. Medzi charakteristiky patrí aj ohnisková vzdialenosť f.

V prípade, že chceme analyzovať vzorky, ktoré obsahujú veľa rôznych prvkov, ich spektrá budú veľmi husté. Na takéto experimenty je potrebný spektroskop s vysokým rozlíšením a s veľkým spektrálnym rozsahom. Rozlíšenie je úmerné počtu čiar vrytých, resp. vyleptaných do difrakčnej mriežky. Mriežky v spektroskopoch typu Czerny-Turner dosahujú hustotu 100 až 4800 rýh na milimeter.

Potreby na zvyšovanie spektrálneho rozsahu spektroskopov vyústili do nového usporiadania typu Echelle. Tento typ usporiadania má stupňovitú difrakčnú mriežku, ktorá má hustotu vrypov menšiu ako 300 rýh na milimeter a ktorá spolu s deličom rádov rozdeľuje energiu žiarenia do vyšších difrakčných rádov a vytvára dvojrozmerný vzor. Vnútorné usporiadanie typu Echelle je znázornené na obrázku 1.3. Tento systém teda poskytuje široký spektrálny rozsah medzi 200 nm až 780 nm [14].

Nevýhodou tohto typu spektroskopu je, že spektrálna citlivosť nie je rovnaká v celom rozsahu spektra a znižuje sa pre vlnové dĺžky nad 600 nm. Disperzia navyše obsahuje tzv. *mŕtve zóny* medzi jednotlivými difrakčnými rádmi. V týchto mŕtvych zónach je signál stratený. Informácie v tomto oddieli boli čerpané z literatúry [11], [13] a [12].

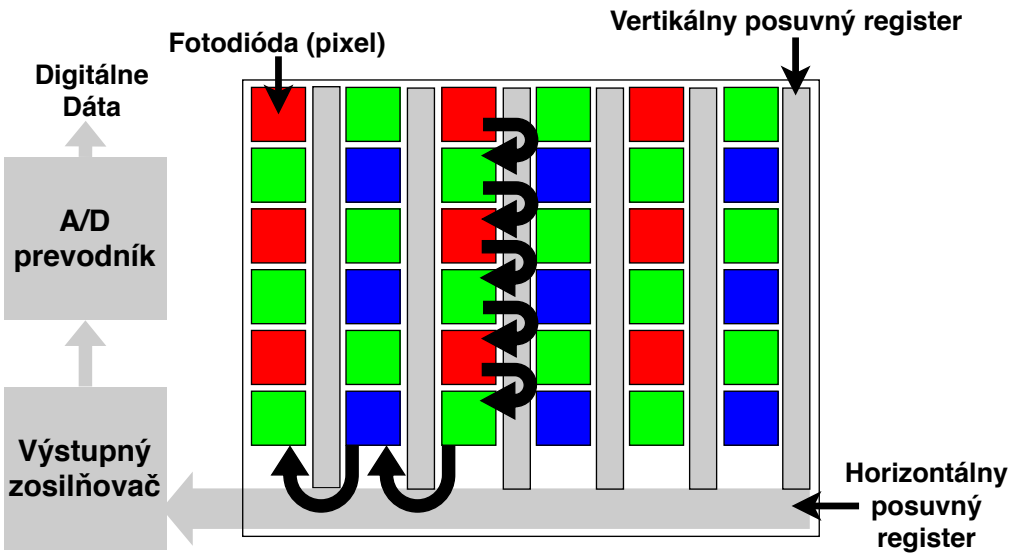


Obr. 1.3: Vnútorné usporiadanie spektroskopu typu Echelle

1.1.2 Typy detektorov

Po rozložení svetla spektroskopom je potrebné tento výstup zachytiť a ďalej ho analyzovať. Na tento účel slúži detektor, ktorý sa pripája na výstupnú štrbinu spektroskopu, prípadne je do spektroskopu zabudovaný napevno.

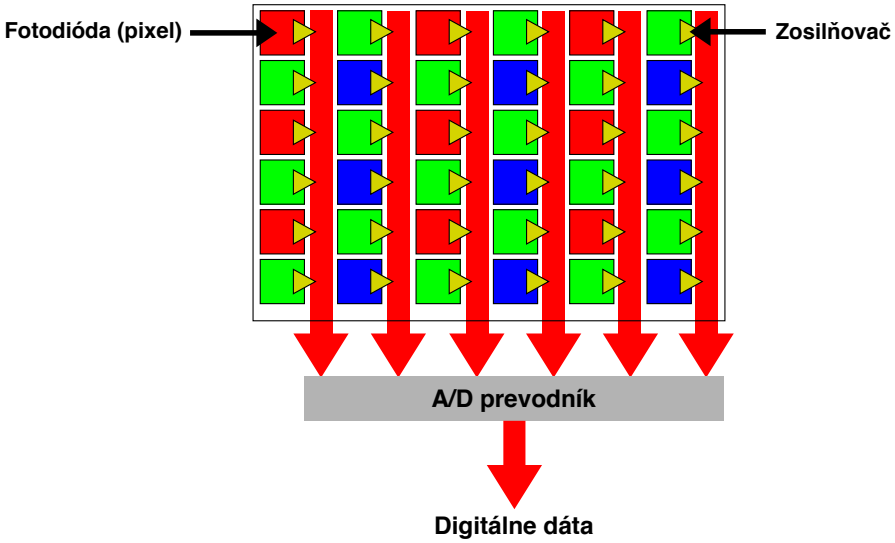
Jedným z typov detektorov je fotonásobič (PMT), ktorý je v súčasnej dobre nahradený modernejšími technológiami, ako napríklad detektor na princípe fotodiód PDA (Photodiode Arrays). Detektor tohto typu sa skladá zo skupiny fotocitlivých diód umiestnených v jednom rade. Avšak v posledných rokoch začali prevládať detektory na princípe nábojovo viazaných štruktúr alebo skrátene CCD (Charge-Coupled Devices).



Obr. 1.4: Konštrukcia plošného CCD snímača

Detektor typu **CCD** sa skladá podobne ako to bolo v prípade PDA z fotocitlivých diód, s tým rozdielom, že CCD snímač má fotodiódy rozmiestnené vo forme matice (obrázok 1.4). Jednotlivé fotodiódy sa tiež označujú pojmom pixely. Na každom pixeli dochádza k premene fotónu (ktorý dopadne na túto plochu) na elektróny. Po skončení expozície sa jednotlivé riadky postupne posúvajú smerom dolu pomocou vertikálnych posuvných registrov. Posledný (najspodnejší) riadok je pomocou horizontálneho posuvného registra postupne prenesený do výstupného zosilňovača, kde dochádza k zosilneniu elektrického náboja, ktorý je vzápätí prevedený na digitálne dáta pomocou A/D prevodníka.

Alternatívou k CCD detektorom je snímač typu **CMOS** (Complementary Metal Oxide Semiconductor). Táto technológia funguje na podobnom princípe, ako to bolo v prípade CCD. Rozdielny je spôsob vyhotovenia, kde v prípade CMOS technológie dochádza k premene fotónu na elektrický náboj a zároveň k zosilneniu elektrického náboja priamo v každom pixeli. Konštrukcia a princíp funkcie tohto typu detektora sú znázornené na obrázku 1.5.



Obr. 1.5: Konštrukcia plošného CMOS snímača

Medzi výhody tohto typu snímača patrí jeho rýchlosť spracovania obrazu, nízka cena, malá spotreba energie a malé rozmery. Na druhej strane, CCD detektory majú lepší dynamický rozsah, vyššie rozlíšenie a všeobecne kvalitnejší obraz. Z tohto dôvodu sa CMOS snímače využívajú iba v lacnejších spektrometroch alebo v spektrometroch, kde sú vyžadované miniatúrne rozmery prípadne menšia spotreba energie. [8] [10]

1.2 Trendy v LIBS

Laserová spektroskopia sa stáva každým rokom čoraz populárnejšou a to nielen medzi vedcami, ale v posledných rokoch sa začína presadzovať aj v iných oblastiach. Ako už bolo spomenuté v úvode tejto kapitoly, jedná sa o bezpečnú, rýchlu a relatívne lacnú technológiu a preto je vhodná aj na priemyselné využitie. Aby bolo možné využívať laserovú spektroskopiu v priemysle, je potrebné zabezpečiť vysokú frekvenciu výstrelov (napr. použitie v triediacich linkách a podobne). Vývoj v oblasti LIBS inštrumentácie (lasery, spektroskopy, detektory atď.) umožňuje zvyšovanie presnosti, rozsahu a rýchlosti meraní. V dnešnej dobe dosahujú zostavy bežne opakovaciu frekvenciu do $20\,Hz$. Ak by sme chceli detekovať inklúziu napr. v oceli, je potrebné preskenovať celú plochu vzorky. V prípade, že chceme vytvoriť mapu zo vzorky, ktorá má plochu $1\,cm^2$ a veľkosť kroku je $20\,\mu m$, je potrebných 250000 meraní. Zostave, ktorá dosahuje frekvenciu $10\,Hz$ by to zabralo takmer 7 hodín. Z tohto dôvodu je zvyšovanie frekvencie merania v laserovej spektroskopii veľmi žiadané. [3] [11]

1.3 Potreby na implementáciu

Ku každej LIBS zostave neodmysliteľne patrí aj počítačový softvér. Aby bolo vôbec možné spustiť meranie, je potrebné správne nastaviť parametre lasera, spektroskopu, detektora a prípadne iných pomocných zariadení. Následne je možné odštartovať generátor pulzov, ktorým sa spustí celý proces merania. Počas merania sú detektorom zaznamenávané dáta, ktoré je potrebné previesť na spektrá a zobraziť ich vhodnou formou užívateľovi (najčastejšie formou grafu). Všetky tieto úkony sú realizované pomocou špeciálnych počítačových aplikácií.

Detektory dokážu počas merania vygenerovať až desiatky gigabajtov. Bežné detektory majú $2000 * 256 = 512000$ pixelov[1] s digitalizáciou 16 bitov (tzn., že každý pixel môže dosiahnuť hodnotu od 0 - 65535). Jedno meranie teda vygeneruje $512000 * 16b \doteq 1\,\text{MB}$. Ak sa použije rovnaký príklad ako z podkapitoly 1.2 (s počtom meraní 250000 a frekvenciou $10Hz$) bude výsledný dátový tok $1\,\text{MB} * 10 = 10\,\text{MB}/s$ a dokopy budú dáta zaberať $1\,\text{MB} * 250000 = 250\,\text{GB}$. Takéto obrovské množstvo dát by bolo v reálnych aplikáciách nemysliteľné. Z tohto dôvodu existujú rôzne optimalizácie (hardvérové alebo softvérové), ktoré dokážu pamäťové nároky značne znížiť, ale na úkor presnosti a kvality merania.

Z uvedených príkladov vyplýva, že výsledná aplikácia musí zvládať zachytiť a následne spracovať veľké množstvo dát a zároveň zachytené dáta vhodným spôso-

[1] Andor iDus 416: `https://www.andor.com/pdfs/specifications/Andor_iDus_416_Specifications.pdf`

bom vykresliť užívateľovi. Všetky tieto úkony musia byť zrealizované v reálnom čase bez toho, aby užívateľ pociťoval spomalenie aplikácie. Zároveň musí užívateľské rozhranie reagovať na vstupné požiadavky užívateľa (napr. zmena nastavení detektora, listovanie medzi zachytenými spektrami a pod.).

1.4 Existujúce softvérové riešenia

Každý výrobca spektrometrov poskytuje k svojim zariadeniam aj príslušný softvér, pomocou ktorého je možné nastavovať parametre spektroskopu a detektora a taktiež vizualizovať a ukladať zachytené dáta.

Ako bolo spomenuté v úvode, cieľom tejto práce je návrh a tvorba univerzálneho rozhrania pre rôzne typy spektrometrov. Výsledná aplikácia by mala obsahovať všetky potrebné ovládacie a vizualizačné prvky a zároveň by mala byť pre koncového užívateľa intuitívna a jednoduchá na ovládanie. Z tohto dôvodu je veľmi vhodné preštudovať a vyskúšať si už existujúce aplikácie od známych výrobcov spektrometrov. Nápomocné by taktiež mohli byť skúsenosti užívateľov, ktorí tieto programy využívajú každodenne. Ich spätná väzba môže odhaliť nedostatky, ktoré nemusia byť na prvý pohľad viditeľné.

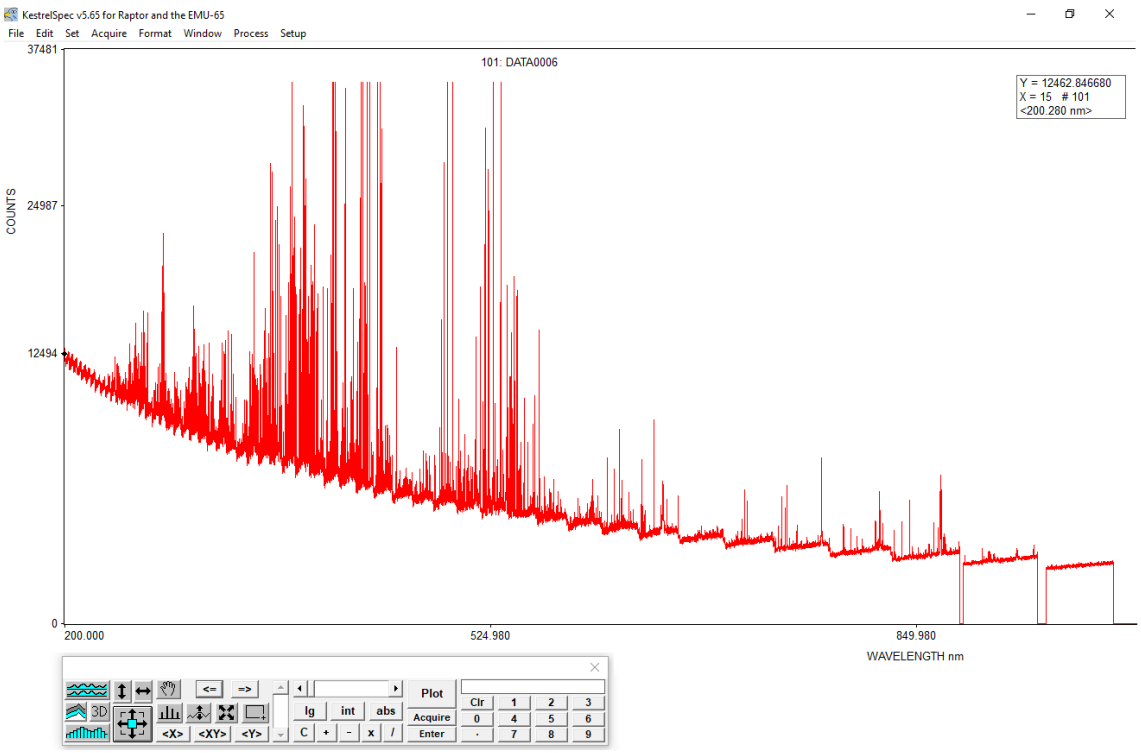
1.4.1 KestrelSpec

KestrelSpec[2] je spektroskopický softvér na zachytávanie a analýzu spektier určený pre operačný systém Windows. Program poskytuje možnosť ovládania detektorov a spektroskopov. Umožňuje vizualizáciu spektier vo forme 2D a 3D grafov, s možnosťou detekcie špičiek v grafe a pod. Ďalšou funkciou je export nameraných dát do ASCII [3] textového formátu.

Výhodou v prípade tohto programu je, že podporuje zariadenia (spektrometre a detektory) od viacerých výrobcov. Ďalšou výhodou je jeho jednoduchosť. Napríklad kliknutím do grafu sa označí najvyšší vrchol spektra a zobrazí sa jeho vlnová dĺžka a intenzita. Taktiež proces kalibrácie je ralizovaný iba jediným tlačidlom. Ako je vidno na obrázku 1.6, základné užívateľské rozhranie obsahuje len minimum tlačidiel a jedno okno s grafom, kde je zobrazené posledné zachytené spektrum. KestrelSpec je teda vhodný pre užívateľov, ktorým stačí jednoduché rozhranie na základné nastavenie spektrometra a detektora a na jednoduchú vizualizáciu dát. Ak však užívateľ potrebuje využívať aj pokročilé funkcie a možnosti, ktoré táto aplikácia poskytuje, vyžaduje si to istú prax a vedomosti. To je hlavným nedostatkom

[2] Špecifikácia dostupná na: `http://www.catalinasci.com/WEBPDF/Kestrel_Brochure.pdf`

[3] ASCII (American Standard Code for Information Interchange) je spôsob kódovania znakov anglickej abecedy v informatike



Obr. 1.6: Ukážka užívateľského rozhrania KestrelSpec

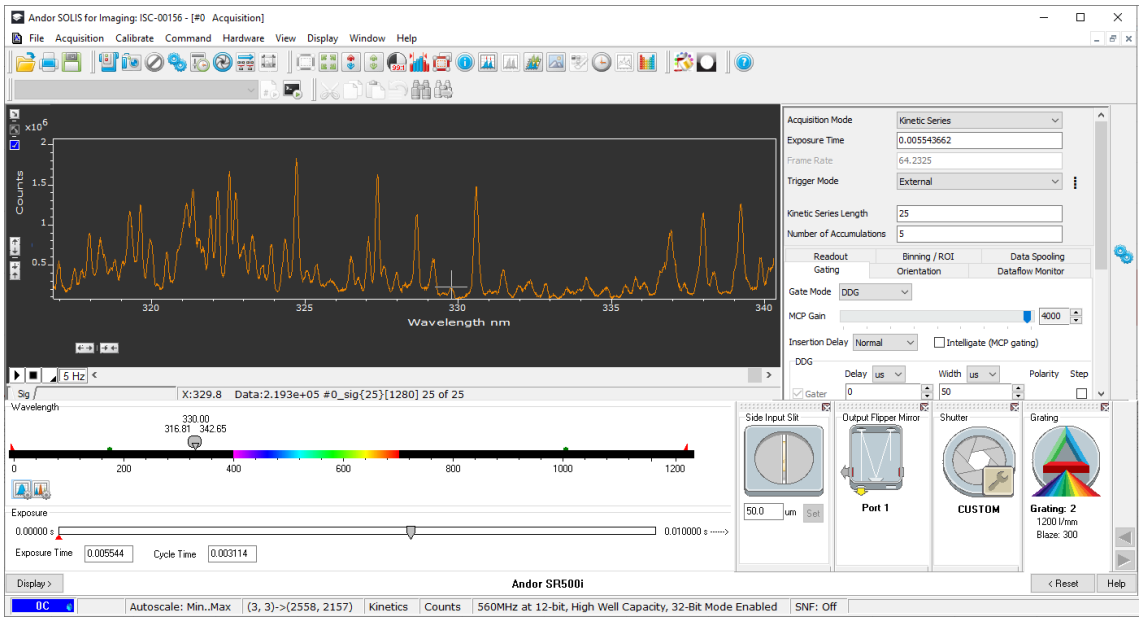
tohto programu, pretože ovládanie týchto pokročilých funkcií je značne neintuitívne. Navyše, užívateľské rozhranie obsahuje množstvo skrytých chýb, ktoré dokážu znehodnotiť celé meranie. Za nevýhodu môže byť považovaná aj skutočnosť, že je táto aplikácia určená iba pre operačné systémy Windows.

1.4.2 Andor Solis

Druhým príkladom na ovládanie a analýzu spektroskopických dát je aplikácia Andor Solis.[4] Jedná sa o program, ktorý je vhodný na rôzne spektroskopické aplikácie a je určený iba pre platformy s operačným systémom Windows. Poskytuje plnú kontrolu nad spektroskopom a detektorom. Samozrejmosťou sú rôzne spôsoby zobrazenia spektier (2D, 3D, prekrývanie, zmena farieb grafu a pod.) a taktiež možnosť exportovania dát do rôznych typov výstupných formátov. Na obrázku 1.7 je zobrazené užívateľské rozhranie počas zachytávania dát zo spektrometra.

Medzi hlavné výhody tejto aplikácie patria hlavne rozmanité možnosti nastavovania parametrov spektroskopu a detektora a možnosť nekonečného snímania spektier.

[4]Špecifikácia dostupná na: `https://andor.oxinst.com/assets/uploads/products/andor/documents/Solis-Brochure.pdf`



Obr. 1.7: Ukážka užívateľského rozhrania Andor Solis

Ďalej sú to možnosti zobrazovania a práce so zachytenými spektrami a solídne grafické spracovanie, ktoré je na rozdiel od programu KestrelSpec 1.4.1 na oveľa vyššej úrovni. Na druhej strane, toto grafické spracovanie má aj isté nevýhody a to hlavne neprehľadnosť, pretože užívateľské rozhranie obsahuje množstvo obrázkových ikon, pri ktorých nemusí byť na prvý pohľad jasné, na aký účel slúžia. Aplikácia obsahuje zopár chýb ako napríklad, že pri ukladaní veľkého množstva spektier dochádza k poruchám a strate dát. Ďalšou nevýhodou tejto aplikácie je, že je určená iba pre systémy s operačným systémom Windows a podporuje zariadenia iba od firmy Andor.

Tieto informácie potvrdzujú aj skúsenosti užívateľov, ktorí pravidelne pracujú s oboma uvedenými programami.

2 Použité technológie

Aby bolo možné vytvoriť návrh riešenia a následne implementovať výslednú aplikáciu, je nevyhnutné, si dobre premyslieť, aké implementačné technológie zvoliť a použiť. Výberom vhodných technológií sa práca výrazne zjednoduší a predíde sa aj množstvu problémov, ktoré by mohli vzniknúť ich nesprávnou voľbou. Napríklad voľba nesprávneho programovacieho jazyka môže mať za následok spomalenú reakciu aplikácie na vstupné dáta (čo môže byť v tomto prípade kritické) a podobne.

Na riešenie tejto práce bol zvolený jazyk C++ (podkapitola 2.1), knižnica Qt (podkapitola 2.2) a protokol softvérovo definovaného zariadenia (podkapitola 2.3).

2.1 Jazyk C++

Ako bolo spomenuté vyššie, na implementáciu tejto práce bol vybraný programovací jazyk C++. Hlavnou výhodou tohto jazyka je kombinácia rýchlosti a objektovo orientovaného prístupu. Práve z tohto dôvodu je veľmi vhodnou voľbou pre danú prácu, pretože rýchlosť výsledného programu je v tomto prípade kľúčová (podkapitola 1.3). Jazyk C++ je jedným z najrozšírenejších programovacích jazykov na svete a je veľmi dobre zdokumentovaný. Na internete existuje množstvo príkladov a riešení problémov, ktoré sú často nápomocné pri samotnom programovaní. Ďalšou výhodou je, že pre tento jazyk existuje množstvo knižníc, ktoré dokážu značne urýchliť a zjednodušiť prácu programátora.

2.2 Knižnica Qt

Jednou z knižníc, ktoré uľahčujú prácu programátora v jazyku C++ je knižnica *Qt*. Táto knižnica umožňuje tvorbu grafického rozhrania a je kompatibilná medzi viacerými platformami ako sú Windows, Linux a pod. Jej veľkou výhodou je kvalitne a podrobne spracovaná online dokumentácia. [1] Na prácu s touto knižnicou je určené vývojové prostredie *Qt Creator*, ktoré je taktiež na veľmi vysokej úrovni, čo sa týka celkového dizajnu, ale aj funkčnosti. Na preklad zdrojového kódu je podporovaných viac typov kompilátorov a na automatizáciu generovania tzv. *makefile*[2] je možné využiť nástroj *qmake*.

Grafické rozhranie je možné vytvárať pomocou tzv. *widgetov* (všeobecné označenie pre ovládací prvok). Aplikácie, ktoré sú postavené na widgetoch majú väčšinou rovnaký dizajn (štýl, font písma, farby a pod.) a využívajú štandardné ovládacie

[1] Dokumentácia je dostupná na: `https://doc.qt.io/`

[2] Makefile určuje postup a závislosti medzi zdrojovými súbormi pri preklade programu

prvky ako napríklad rozbaľovacie okno, textové pole atď. Widgetové aplikácie sa vytvárajú v prostredí Qt Creator veľmi jednoducho. Programátor si vytvorí projekt s hlavným oknom, do ktorého myšou vkladá štandardné ovládacie prvky. Výhodou je, že programátor vidí už počas návrhu, ako daná aplikácia vyzerá.

Druhou možnosťou tvorby grafického rozhrania je jazyk QML (angl. Qt Modeling Language). Jedná sa o deklaratívny jazyk podobný zápisu JSON (angl. JavaScript Object Notation). Poskytuje rozmanitejšie možnosti návrhu grafického rozhrania oproti predchádzajúcemu spôsobu pomocou widgetov. QML je vhodné použiť na grafické rozhrania, ktoré nemajú štandardný vzhľad a taktiež pre zariadenia s dotykovým ovládaním. [2] [6] [9]

2.3 Softvérovo definované zariadenie

Softvérovo definované zariadenie alebo skrátene SDD (angl. Software Defined Device) je komunikačný protokol založený na protokole HTTP[3]. Jedná sa o nový spôsob, ako definovať zariadenia, ktoré môžu medzi sebou jednoducho komunikovať. Protokol softvérovo definovaných zariadení bol vyvinutý v technologickom inštitúte CEITEC VUT. Architektúra tohto protokolu je modulárna a tzv. *fail-safe*. To znamená, že v prípade ak niektoré zariadenie zlyhá, zvyšok systému môže fungovať ďalej.

Fyzické prepojenie zariadení bude v tomto prípade realizované pomocou ethernetového[4] rozhrania. Dáta, ktoré si zariadenia medzi sebou posielajú, sú enkapsulované do štruktúry JSON. Zariadenia je možné zapojiť do stromovej štruktúry, čo vlastne vychádza z definície ethernetu. Príklad zapojenia SDD zariadení do topológie je znázornený na obrázku 2.1. Logická štruktúra tohto protokolu môže, ale aj nemusí byť totožná s fyzickou štruktúrou. To znamená, že napríklad jedno fyzické zariadenie sa môže skladať z viacerých logických.

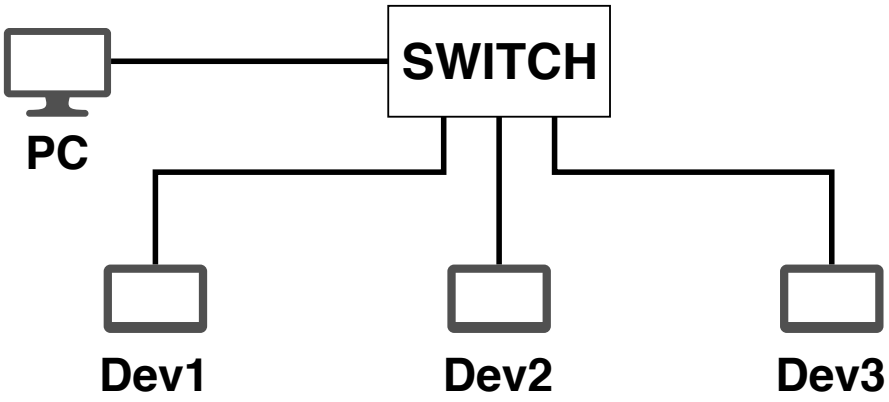
Tento protokol poskytuje dva základné prvky:

- **Device** alebo zariadenie môže reprezentovať buď konkrétne fyzické zariadenie (napr. spektrometre, lasery atď.), alebo logické (abstraktné) zariadenie (napr. zóna, ktorá združuje skupinu teplomerov)
- **Remote device** alebo vzdialené zariadenie väčšinou reprezentuje softvér, ktorý dané zariadenie ovláda

Po štarte systému si každé vzdialené zariadenie (PC) preverí, koľko ďalších zariadení je dostupných v danej topológii. Preverovanie siete je realizované pomocou tzv.

[3]HTTP (HyperText Transfer Protocol) je protokol založený na modeli klient-server a je určený na prenos hypertextových (štruktúrovaných) dokumentov

[4]Ethernet – štandard IEEE 802.3 je technológia pre lokálne počítačové siete



Obr. 2.1: Príklad SDD topológie

discovery paketu na všesmerovú IPv4 adresu. Všetky zariadenia prijímajú discovery pakety na UDP porte 16549. Každý takýto paket obsahuje špecifický reťazec znakov, za ktorým nasledujú informácie o odosielateľovi, ako napr. jeho IP adresa s TCP portom, na ktorom čaká odpoveď, meno zariadenia a verzia firmvéru. Zariadenie, ktoré príjme discovery paket, si v prvom rade overí úvodný špecifický reťazec. V prípade, že je správny, odpovie vzdialenému zariadeniu svojimi informáciami o konfigurácií na IP adresu a TCP port, ktorý bol uvedený v tele discovery paketu. Celý tento popísaný mechanizmus umožňuje zariadeniu vstup do systému v ľubovoľnom čase a to periodickým zasielaním discovery paketov. Priebeh komunikácie je znázornený na obrázku 2.2 [4]

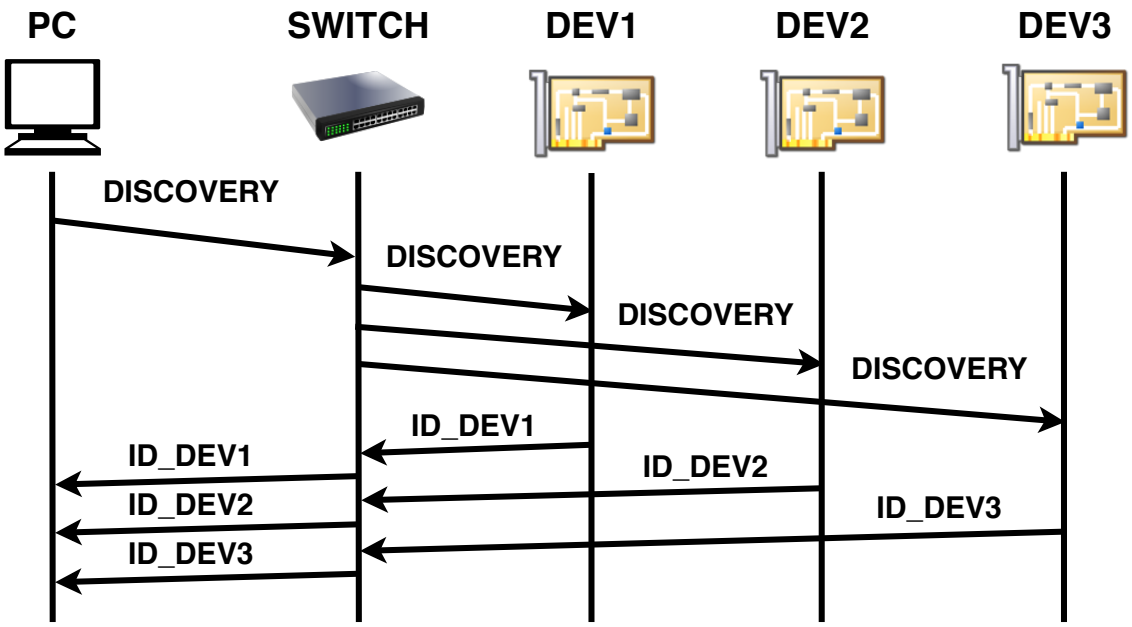
V každom zariadení je možné vytvárať ľubovoľný počet nasledovných atribútov:

- **Property** – udržuje jeden zvolený dátový typ (napr. int, double a pod.), pozri oddiel 2.3.1
- **Slot** – slúži ako signál, ktorý môže odštartovať nejakú udalosť (oddiel 2.3.2)
- **File** – reprezentuje binárny súbor a používa sa na prenos binárnych dát medzi device a remote device (oddiel 2.3.3)

2.3.1 Premenná hodnota (property)

Atribút *property* slúži na reprezentáciu parametrov daného zariadenia. Napríklad ak je potrebné definovať jednoduchý ventilátor, bude potrebné vytvoriť minimálne tri atribúty *property*, ktoré budú reprezentovať:

- **Aktuálne otáčky** - počet otáčok za minútu.
- **Nastavený výkon** - napr. v percentách.
- **Stav** - vypnutý/zapnutý.



Obr. 2.2: Priebeh komunikácie pri hľadaní zariadení v sieti

Z príkladu je vidno, že každé zariadenie zvyčajne obsahuje viacero atribútov *property* a ako už bolo spomenuté, môžu obsahovať jeden zo základných dátových typov. Pri definícii *property* sa uvedie unikátny názov, a rovnako tak možnosť modifikovať obsah (tzv. *read-only* parameter). Navyše, pre každú *property* je potrebné nastaviť, aký rozsah hodnôt, bude považovaný za platný pomocou tzv. validátora. Hodnotu *property* môže meniť zariadenie aj vzdialené zariadenie, ale iba v prípade, ak nebol definovaný parameter read-only. V prípade zmeny hodnoty *property*, je zariadenie automaticky informované o tejto zmene. To však neplatí pre vzdialené zariadenie. Aby bolo vzdialené zariadenie taktiež automaticky informované o zmene hodnoty *property*, musí požiadať o odoberanie zmien danej *property*. Ak však vzdialené zariadenie neodoberá zmeny danej *property*, môže jednorazovo požiadať o zaslanie jej aktuálnej hodnoty.

2.3.2 Signál (slot)

V prípade, že je potrebné odoslať signál, ktorý odštartuje nejakú udalosť, je vhodné použiť atribút *slot*. Slot je možné použiť v oboch smeroch (t. j. zo zariadenia na vzdialené zariadenie a opačne). Napríklad ak chceme odštartovať alebo zastaviť činnosť snímača v spektrometri, použije sa na to práve atribút slot.

2.3.3 Súbor (file)

Na prenos veľkého množstva binárnych dát pomocou protokolu SDD slúži atribút *file* alebo súbor. Prenos dát je navrhnutý tak, aby pracoval iba smerom zo zariadenia, ktoré získava dáta (napr. spektrá zo spektrometra) do vzdialeného zariadenia, ktoré tieto dáta ďalej spracováva. Binárny prenos pozostáva z prenosu riadiacich informácií o zmenách v súbore a z prenosu samotných dát.

Informácie o zmenách v súbore

Tento mechanizmus sa používa na informovanie vzdialeného zariadenia o všetkých zmenách v súborovej štruktúre. Existujú tri typy správ, ktoré nesú informáciu:

- **Nový súbor** - vytvorenie nového súboru.
- **Nové dáta** - dáta v súbore boli zmenené.
- **Vymazaný súbor** - daný súbor bol vymazaný.

Každá správa obsahuje názov súboru, ktorého sa táto správa týka. Ak chce vzdialené zariadenie automaticky dostávať informácie o zmenách v súbore, musí zariadeniu poslať žiadosť o odoberanie zmien v súbore.

Binárny prenos

Ak chce zariadenie odoslať súbor, musí najprv nejaký vytvoriť, pričom sa musí definovať názov nového súboru a jeho obsah. Následne je možné kedykoľvek meniť obsah tohto súboru novými dátami. V prípade, že už daný súbor nie je ďalej potrebný, je možné ho vymazať.

Na opačnej strane, ak chce vzdialené zariadenie automaticky prijímať nový obsah súboru, musí požiadať o odoberanie daného súboru. V prípade, že vzdialené zariadenie neodoberá tento súbor, stále môže poslať jednorazovú požiadavku na získanie aktuálneho obsahu daného súboru.

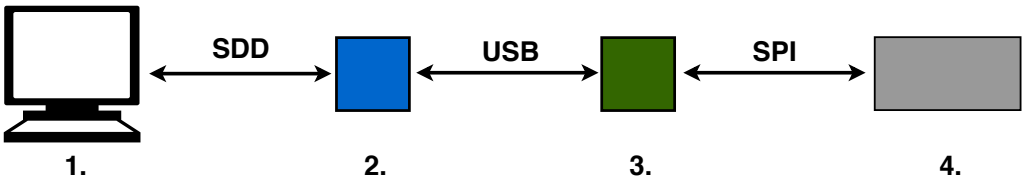
3 Návrh riešenia

Návrh riešenia tejto práce sa bude odvíjať od troch hlavných častí. Prvým bodom bude vyriešenie komunikácie medzi spektrometrom a počítačom (podkapitola 3.1). Keď budú dáta zo spektrometra prenesené do počítača s užívateľským rozhraním, je nutné ich ďalej spracovať. V prvom rade sa dáta musia vykresliť užívateľovi formou grafu (podkapitola 3.2) a následne je potrebné ich uložiť vhodným spôsobom na pevný disk (podkapitola 3.3). Posledným bodom je návrh samotného grafického rozhrania (podkapitola 3.4).

3.1 Zaistenie komunikácie

Zaistenie komunikácie sa skladá z dvoch častí. Schéma komunikácie je zobrazená na obrázku 3.1. V prvej časti sa využíva komunikačný protokol softvérovo definovaných zariadení (pozri podkapitolu 2.3). Vzdialené zariadenie bude v tomto prípade počítač s grafickým užívateľským rozhraním. Na opačnej strane sa bude nachádzať SDD zariadenie, ktoré môže byť vo forme dedikovaného zariadenia s RTOS[1] operačným systémom alebo to môže byť pokojne rovnaký počítač, na ktorom je spustené užívateľské rozhranie (tzv. *localhost*[2] komunikácia).

V druhej časti je potrebné zabezpečiť komunikáciu medzi SDD zariadením a spektrometrom. Na riešenie a testovanie funkčnosti tejto práce bol použitý spektrometer typu *FCU-109*[3] od firmy Ibsen s integrovaným CMOS snímačom (pozri oddiel 1.1.2).



Obr. 3.1: Schéma komunikácie medzi zariadeniami. 1) Počítač s užívateľským rozhraním (vzdialené SDD zariadenie), 2) SDD zariadenie, 3) Prevodník USB-SPI, 4) Spektrometer FCU-109

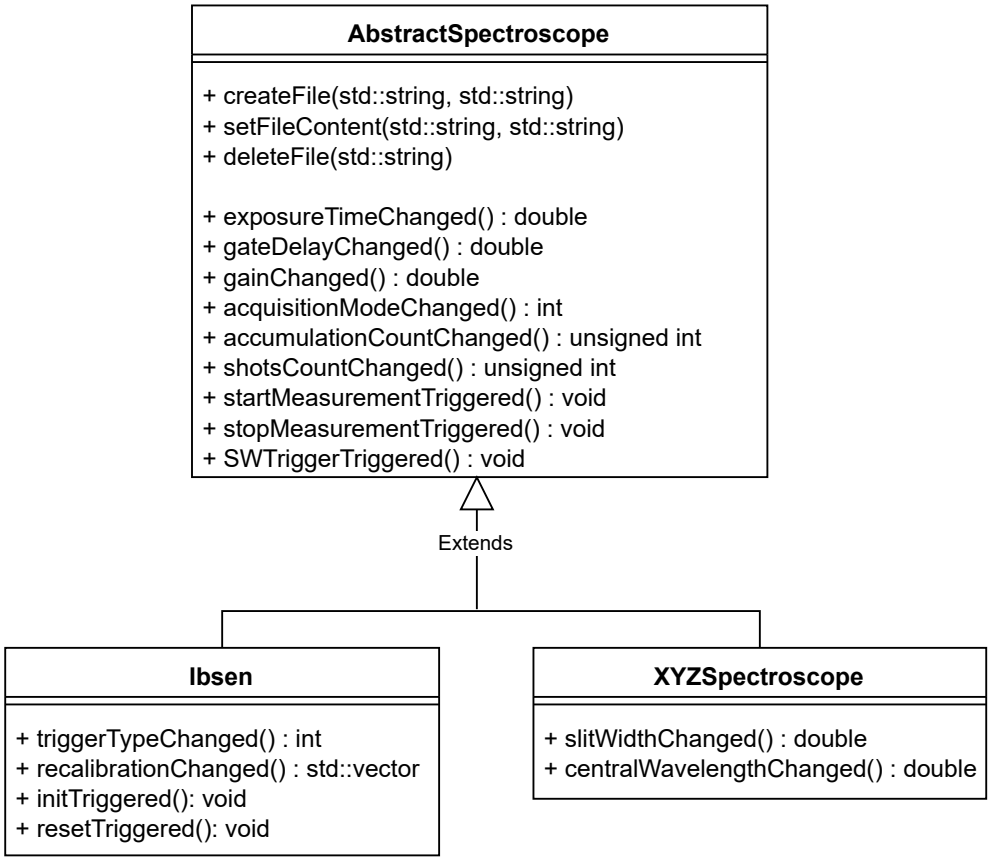
[1]RTOS (Real-Time Operating System) je typ operačného systému, ktorý sa používa vo vstavaných zariadeniach, robotike, telekomunikáciách a pod.

[2]localhost je odkaz na práve používaný počítač, pričom sa využíva v prípade IPv4 špeciálna adresa 127.0.0.1

[3]Bližšia špecifikácia spektrometra Ibsen typu Freedom FCU-109 je dostupná na: `https://ibsen.com/wp-content/uploads/Ibsen-Product-Sheets-FREEDOM-C-UV.pdf`

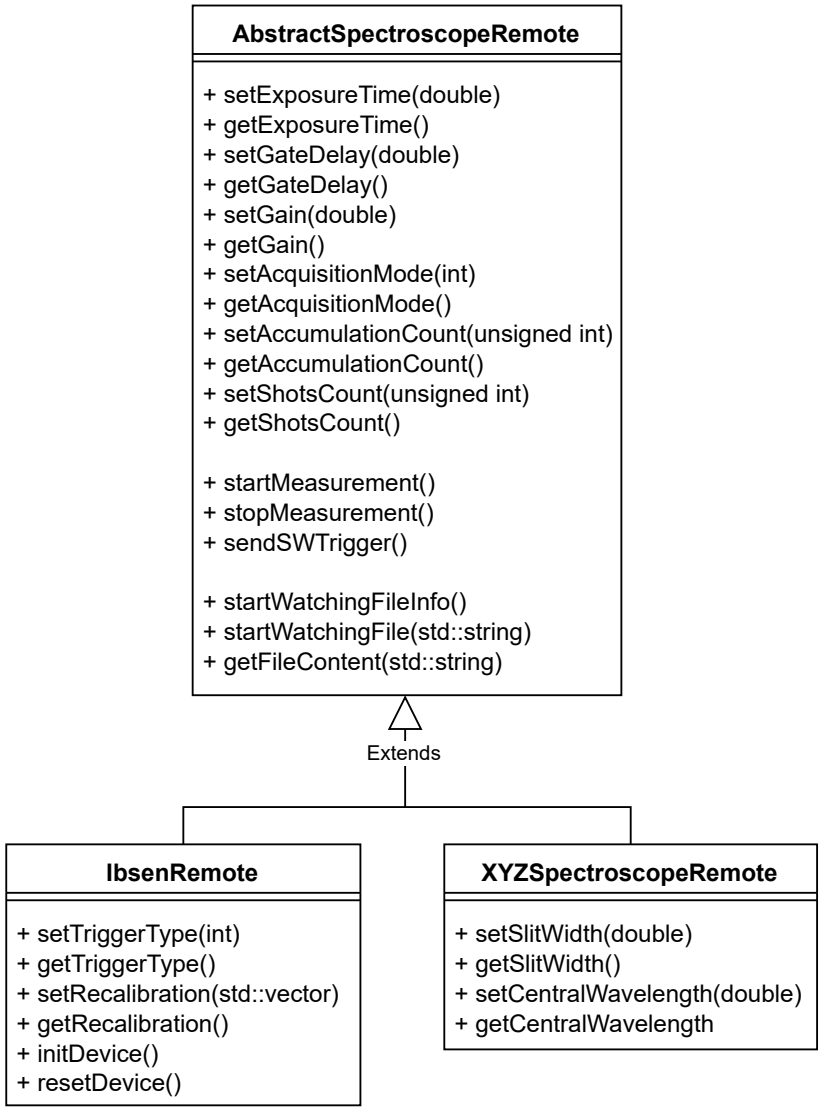
3.1.1 Komunikácia medzi užívateľským rozhraním a SDD zariadením

Na komunikáciu medzi užívateľským rozhraním a SDD zariadením je použitý protokol softvérovo definovaných zariadení. Na implementáciu sa využíva rovnomenná knižnica, ktorá SDD protokol implementuje. Aby mohla vzniknúť komunikácia, v prvom rade sa musia do knižnice SDD pridať dve triedy. Tieto triedy reprezentujú SDD zariadenie a vzdialené SDD zariadenie. Každý spektrometer môže mať špecifické parametre, ktoré nie sú rovnaké pri všetkých typoch spektrometrov. Z tohto dôvodu je nutné vytvárať pre každý typ spektrometra dvojicu nových SDD tried. Na druhej strane existuje množstvo parametrov, ktoré majú všetky spektrometre rovnaké. Vzhľadom na to, že cieľom tejto práce je vytvoriť univerzálne rozhranie, je veľmi výhodné tieto rovnaké parametre zlúčiť do spoločnej triedy. Konkrétne sa bude jednať o dve triedy, z ktorých sa pomocou dedičnosti môžu vytvárať nové triedy pre konkrétny typ spektrometra. Pre lepšiu predstavu štruktúry tried je na obrázku 3.2 zobrazený UML diagram SDD zariadenia a na obrázku 3.3 UML diagram vzdialeného SDD zariadenia.



Obr. 3.2: Čiastočný UML diagram SDD zariadenia

Na obrázku 3.2 je uvedený UML diagram SDD zariadenia. Trieda `AbstractSpectroscope` obsahuje funkcie na prácu s binárnym súborom (pozri oddiel 2.3.3), ktorý sa používa na prenos zachytených spektier. Táto trieda taktiež obsahuje tzv. *callback*[4] funkcie, ktoré sú volané zakaždým, keď dôjde k zmene niektorého parametra zo strany vzdialeného SDD zariadenia. Trieda `Ibsen` dedí všetky vlastnosti z triedy abstraktného zariadenia a navyše implementuje nové callback funkcie, ktoré sú pre tento konkrétny typ spektrometra špecifické. Ako príklad je uvedená trieda `XYZSpectroscope`, ktorá ukazuje možnosť pridania nového typu spektrometra.



Obr. 3.3: Čiastočný UML diagram vzdialeného SDD zariadenia

Z UML diagramu na obrázku 3.3 je vidno, že trieda abstraktného vzdialeného

[4]Callback funkcia je funkcia, ktorá je predaná inej funkcii napr. A() ako argument. Po vykonaní tela funkcie A() sa automaticky zavolá aj callback funkcia.

SDD zariadenia `AbstractSpectroscopeRemote` implementuje množstvo funkcií, ktoré sú pri všetkých typoch spektrometrov rovnaké. V prípade triedy `IbsenRemote` dochádza opäť k dedeniu všetkých vlastností z triedy abstraktného zariadenia. Táto trieda navyše implementuje vlastné funkcie, ktoré sú pre tento konkrétny typ spektrometra špecifické. Trieda `XYZSpectroscopeRemote` je znovu uvedená ako príklad pridania nového typu spektrometra.

3.1.2 Komunikácia medzi SDD zariadením a spektrometrom

Zvolený spektrometer FCU-109 dokáže komunikovať iba pomocou sériového periférneho rozhrania SPI[5]. Zariadenie SDD má v tomto prípade k dispozícii iba zbernicu USB (angl. Universal Serial Bus). Aby bolo možné zabezpečiť komunikáciu medzi SDD zariadením a spektrometrom, použije sa špeciálne zariadenie (pozri obrázok 3.1, zariadenie č.3) na prevod medzi USB a SPI rozhraním. Toto zariadenie bolo navrhnuté a vyrobené v technologickom inštitúte CEITEC. Na prevod medzi rozhraniami sa využíva mikrokontrolér od firmy FTDI typu *FT2232*[6].

Použitím USB-SPI prevodníka teraz môže SDD zariadenie jednoducho komunikovať so spektrometrom. Spektrometer je možné ovládať zápisom špecifických hodnôt do jeho registrov. Adresy registrov s popisom funkcie, tzv. registrová mapa, je uvedená v manuáli, ktorý bol súčasťou balenia so spektrometrom.

3.2 Vizualizácia dát

Po zachytení a prenesení nameraných dát do počítača s užívateľským rozhraním, je potrebné tieto dáta vizualizovať. V laserovej spektroskopii sa na zobrazenie spektier využívajú najčastejšie čiarové grafy.

Na tvorbu čiarového grafu bola použitá knižnica Qt, s využitím jazyka QML (pozri podkapitolu 2.2). Graf musí podporovať vykresľovanie viacerých spektier, pričom každá čiara musí mať špecifikovaný svoj názov, farbu a Y súradnice. Na dosiahnutie týchto požiadaviek je potrebné vytvoriť štruktúru, ktorá tieto vlastnosti udržuje. Príklad takejto štruktúry je uvedený vo výpise 3.1. Vodorovná os X je pre všetky spektrá rovnaká, a preto nemusí byť súčasťou každej čiary. Rovnaká je z toho dôvodu, že každé spektrum, ktoré bude zachytené jedným spektrometrom, musí mať os X, nazvanú aj kalibrácia, vždy rovnakú.

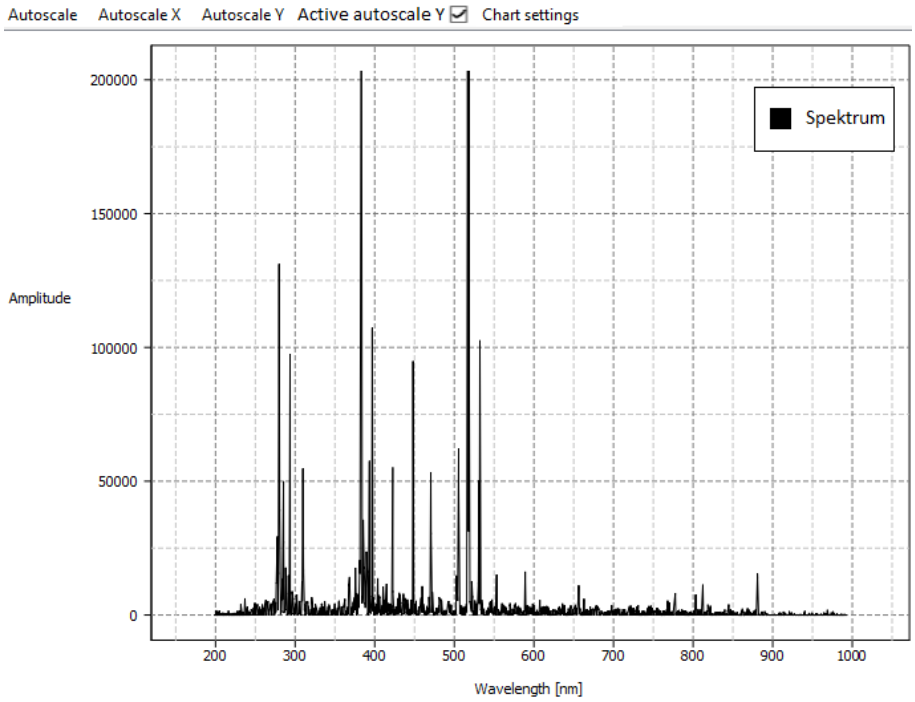
[5]SPI (Serial Peripheral Interface) je synchrónne sériové komunikačné rozhranie, ktoré sa využíva prevažne vo vstavaných zariadeniach na komunikáciu medzi mikroprocesormi a integrovanými obvodmi

[6]Bližšia špecifikácia USB-SPI prevodníka FTDI FT2232 je dostupná na: `https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT2232D.pdf`

```
1 struct PlotLine {
2     string lineName;  //Reprezentuje názov danej čiary
3     color  lineColor; //Reprezentuje farbu danej čiary
4     vector data;      //Reprezentuje Y hodnoty danej čiary
5 };
```

Výpis 3.1: Príklad štruktúry na reprezentáciu jednej čiary grafu

Výsledný návrh grafu je zobrazený na obrázku 3.4. Graf musí zobrazovať popis oboch osí a taktiež by mal obsahovať legendu, pomocou ktorej môže užívateľ lepšie porozumieť zobrazeným dátam. Na uľahčenie vyčítania údajov z grafu je vhodné zobrazenie mriežky.



Obr. 3.4: Návrh vykresľovania grafu

Aby bola práca s grafom efektívna, musí byť umožnené nad grafom vykonávať základné operácie, ako sú:

- **posun**
- **priblíženie/oddialenie**
- **zobrazenie zvoleného úseku**
- **automatické nastavenie rozsahu**
- **automatické popisy osí**

Posun grafu funguje ako takmer pri všetkých grafových aplikáciách. Pri kliknutí ľavým tlačidlom na myši sa aktivuje posun a kým bude toto tlačidlo stlačené, je možné grafom posúvať do všetkých smerov.

Priblíženie a odďaľovanie grafu reaguje na otáčanie kolieskom na myši. Podľa smeru otáčania kolieska sa určí, či sa má graf priblížiť alebo oddialiť. Potrebné je taktiež brať do úvahy pozíciu kurzora myši a k tomuto miestu prispôsobiť približovanie resp. odďaľovanie grafu.

Ďalšou funkciou, ktorá patrí medzi operácie nad grafom, je zobrazenie alebo priblíženie zvoleného úseku. Stlačením a podržaním kolieska na myši sa na pozícii kurzora myši aktivuje obdĺžnikový selektor. Zmenou pozície myši sa vytvorí obdĺžnik medzi miestom kliknutia a uvoľnením kolieska myši. Tento obdĺžnik reprezentuje miesto, ktoré bude prispôsobené (priblížené) na aktuálne rozmery okna grafu.

Užitočnou funkciou je tiež automatické nastavenie rozsahu zobrazenia. Kombináciou už spomenutých funkcií dokáže užívateľ rozladiť zobrazenie grafu. Funkcia automatického nastavenia rozsahu umožňuje obnoviť ideálne zobrazenie grafu. Táto funkcia je navyše rozdelená na automatické nastavenie oboch alebo iba jednej z osí. Automatické nastavenie iba jednej osi môže byť užitočné v prípade, keď sa užívateľ potrebuje zamerať na určitý interval niektorej osi a druhú os potrebuje prispôsobiť rozsahu.

Nevyhnutnou súčasťou každého grafu je popis hodnôt jednotlivých osí. V tomto konkrétnom prípade sa jedná o hodnoty vlnových dĺžok na horizontálnej osi a hodnoty intenzít jednotlivých zložiek nasnímaného svetla na osi vertikálnej. Pri vytváraní takéhoto grafu počítačom, je veľmi vhodné popisy osí generovať automaticky a intervaly medzi jednotlivými číslami voliť "rozumne". To znamená, že zobrazené čísla budú napr. násobky 1, 2, 5 alebo 10. V takto zvolených popisoch osí sa vie užívateľ oveľa lepšie orientovať ako keby bol rozsah a interval číslovania ľubovoľný. Ak máme napríklad rozsah dát 204 až 982, vhodným rozsahom na zobrazenie bude 200 až 1000 s intervalom 100 medzi jednotlivými číslami (pozri obrázok 3.4). Pre človeka je relatívne jednoduché určiť takýto interval, ale v prípade počítača je potrebné zvoliť vhodný algoritmus, ktorý dokáže zakaždým vygenerovať požadované rozsahy správne [7].

3.3 Ukladanie dát

Ďalším bodom návrhu je spôsob ukladania zachytených dát. Vzhľadom na objem dát, ktorý môžu vzniknúť (pozri podkapitolu 1.3), by mal výstupný súbor zaberať čo najmenej miesta na disku. Zároveň, by práca s veľkým objemom dát nemala výrazne spomaľovať prácu s aplikáciou.

Zachytené dáta sa ukladajú do dvoch súborov:

- **libsmetadata**
- **libsdata**

V súbore *Libsmetadata* sú uložené informácie o jednotlivých spektrách. Tieto informácie sa tiež nazývajú metadáta a slúžia na bližšiu špecifikáciu konkrétneho spektra. Metadáta môžu napríklad obsahovať súradnice, na ktorých bolo dané spektrum získané a pod. Okrem metadát bude tento súbor obsahovať aj počet vlnových dĺžok v jednom spektre a počet spektier. Ukladanie do tohto súboru bude realizované v zápise JSON.

V druhom súbore *Libsdata* sú uložené samotné dáta. V prvom riadku sú uložené vlnové dĺžky spektrometra ktoré sa nazývajú kalibrácia. Kalibrácia v grafe reprezentuje vodorovnú os. Za kalibráciou potom nasledujú jednotlivé intenzity nameraných spektier (v grafe sa jedná o zvislú os). Z uvedeného vyplýva, že každé namerané spektrum, musí mať rovnaký počet intenzít. V opačnom prípade nebude možné s dátami ďalej pracovať. Hodnoty sa do súboru ukladajú pomocou mapovania súborov do pamäte. Mapovanie súborov do pamäte (angl. Memory-mapped file) je spôsob práce so súbormi. Súbor je z pevného disku mapovaný do operačnej pamäte jadrom operačného systému. To má za následok efektívnejšiu prácu s veľkými súbormi z hľadiska programátora.

V prípade, že si chce užívateľ neskôr zobraziť namerané hodnoty, systém automaticky najprv načíta súbor s metadátami. Z tohto súboru vyčíta počet spektier a vlnových dĺžok, čím zistí veľkosť bloku, ktorý môže následne načítať z dátového súboru.

3.4 Návrh užívateľského rozhrania

Dôležitou časťou tejto práce je užívateľské rozhranie. Je to hlavne kvôli tomu, že každý užívateľ, ktorý príde do kontaktu s touto aplikáciou, bude pracovať práve s užívateľským rozhraním. Z tohto dôvodu je kladený veľký dôraz na jednoduchosť a intuitívnosť ovládania. Docieliť jednoduché a intuitívne ovládanie je často veľmi zložité, a preto je vhodné si dobre premyslieť celý návrh. Je potrebné poznamenať, že text užívateľského rozhrania by mal byť v anglickom jazyku, čo zvýši univerzálnosť použitia medzi rôzne hovoriacimi užívateľmi. Výsledný návrh grafického užívateľského rozhrania je zobrazený na obrázku 3.5.

Základné okno užívateľského rozhrania obsahuje iba komponenty, ktoré sú nevyhnutné k nastaveniu a následnému priebehu merania. Predovšetkým sa jedná o nasledovné komponenty:

- **Menu panel** – zoskupuje všetky podrobnejšie nastavenia a možnosti.
- **Nastavenie počtu meraní** – nastaví počet výstrelov.

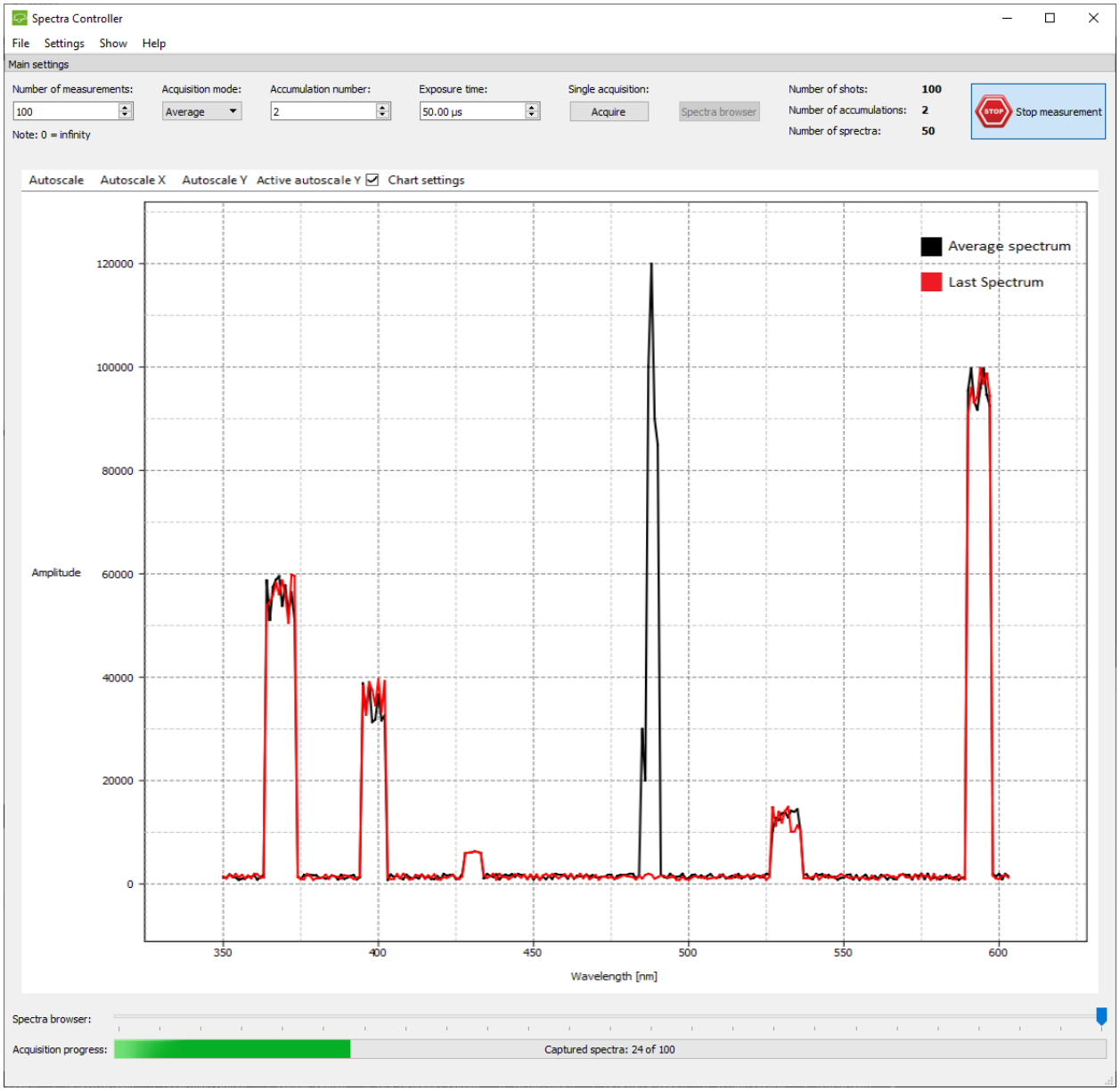
- **Výber spôsobu akumulácie spektier** – voľba buď bez akumulácie, alebo priemerovanie spektier.
- **Nastavenie počtu akumulácií** – ak je zvolené priemerovanie spektier, nastaví počet, z koľkých spektier sa bude robiť priemer.
- **Nastavenie expozičnej doby detektora** – nastaví expozičnú dobu spektrometra.
- **Tlačidlo okamžitého zachytenia jedného spektra** – zosníma jedno spektrum a zobrazí v grafe.
- **Tlačidlo prehliadania zachytených spektier** – je aktívne až po skončení merania.
- **Informačná tabuľka s nastavenými parametrami merania** – informuje užívateľa o počte výstrelov, nastavenom počte akumulácií a výslednom počte zobrazených/uložených spektier.
- **Štart/stop tlačidlo** – odštartuje, resp. zastaví meranie.
- **Graf spektier** – vykreslí posledné zachytené a priemerné spektrum spolu s legendou.
- **Posuvný jazdec na listovanie medzi spektrami** – umožní listovať medzi zachytenými spektrami.
- **Ukazovateľ priebehu merania** – informuje užívateľa, koľko výstrelov ešte ostáva do konca.

Menu panel poskytuje rozšírené možnosti a nastavenia tak, ako je to aj pri ostatných aplikáciách. Patrí sem hlavne výber použitého typu spektrometra, pokročilé nastavenia detektora a spektrometra, zobrazenie logovacích informácií a spôsob exportovania dát. Každá položka z menu panela (s výnimkou logovacích informácií) má svoje vlastné, tzv. modálne dialógové okno[7].

Tlačidlo prehliadania zachytených spektier (Spectra browser) je aktívne až po skončení merania. Po kliknutí na toto tlačidlo sa zobrazí modálne dialógové okno, v ktorom je zobrazený graf, zoznam zachytených spektier a nasledovné ovládacie prvky:

- **Zobraziť/skryť jednotlivé spektrá** – umožňuje zobraziť viac zvolených spektier do jedného grafu.
- **Skryť všetky spektrá** – skryje všetky spektrá z grafu.
- **Vymazať jedno spektrum** – vymaže jedno vybrané spektrum.
- **Vymazať zvolené spektrá** – vymaže vybrané spektrá.
- **Vymazať všetky spektrá** – vymaže všetky zachytené spektrá.

[7]Modálne okno zabraňuje práci s hlavným oknom, kým nie je toto modálne okno obslúžené.



Obr. 3.5: Návrh grafického užívateľského rozhrania

4 Implementácia a riešenie problému

Hlavným bodom tejto práce je riešenie zadaného problému a implementácia samotnej aplikácie. Riešenie je postavené na základe návrhu z kapitoly 3 a s využitím technológií, ktoré sú uvedené v kapitole 2.

4.1 Riešenie komunikácie

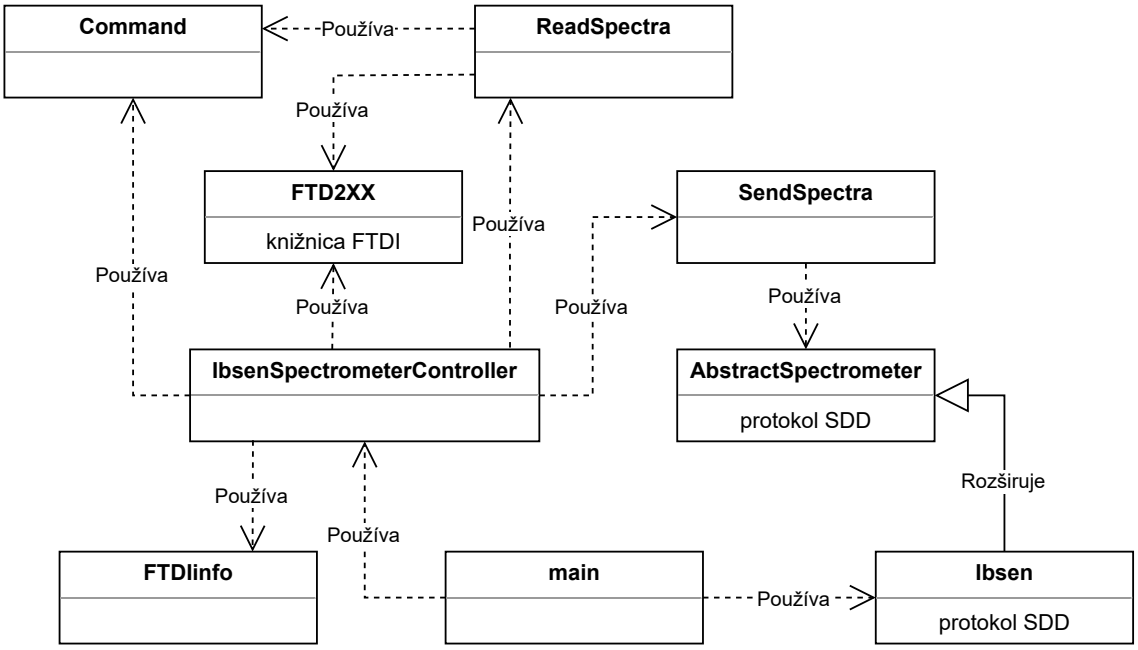
Podľa návrhu z podkapitoly 3.1 sa riešenie komunikácie skladá z dvoch častí. Prvou časťou je ovládač spektrometra, ktorý rieši komunikáciu medzi spektrometrom a SDD zariadením. Druhou časťou je komunikácia medzi SDD zariadením a užívateľským rozhraním.

4.1.1 Ovládač spektrometra

Ovládač spektrometra je implementovaný ako samostatná konzolová aplikácia. Toto riešenie umožňuje spustiť ovládač spektrometra nezávisle na užívateľskom rozhraní. Ovládač pozostáva z dvoch základných objektov. Prvým je SDD zariadenie, ktoré poskytuje komunikáciu s užívateľským rozhraním. Druhý objekt je určený na komunikáciu so samotným hardvérom, resp. spektrometrom.

Pri spustení tejto aplikácie sa v prvom rade zistí IP adresa rozhrania, na ktorom sa bude komunikovať. Následne je možné vytvoriť SDD zariadenie, ktoré bude komunikovať práve na tejto IP adrese. Ďalej je potrebné preskenovať všetky dostupné USB porty a hľadať na nich mikrokontrolér *FT2232*. V prípade, že je dostupných viac takýchto zariadení, prioritne sa kontroluje konfiguračný súbor. Ak konfiguračný súbor obsahuje kód zariadenia a rovnaký kód je aktuálne dostupný, program sa pripojí na toto zariadenie. V opačnom prípade sa aplikácia pripája k prvému dostupnému zariadeniu zo zoznamu a toto zariadenie sa uloží do konfiguračného súboru. Po pripojení sa k zariadeniu, dochádza znovu ku kontrole konfiguračného súboru na obsah kalibračných koeficientov. V prípade, že nie sú prítomné, budú vyčítané zo zariadenia a uložené do konfiguračného súboru. V ďalšom kroku sa pomocou protokolu SDD odošle do užívateľského rozhrania informácia o počte pixelov detektora. Po potvrdení prijatia tejto informácie užívateľským rozhraním sa pokračuje odoslaním kalibrácie, ktorá je vypočítaná zo spomínaných kalibračných koeficientov a počtu pixelov. Kalibrácia je odosielaná pomocou binárneho súboru (pozri oddiel 2.3.3).

Na obrázku 4.1 je zobrazený UML diagram tried, reprezentujúci ovládač spektrometra Ibsen. Jedná sa o zjednodušený UML diagram, ktorý neobsahuje atribúty ani metódy tried. Uvedený diagram slúži hlavne pre lepšie pochopenie štruktúry tried a ich prepojenia medzi sebou.



Obr. 4.1: Zjednodušený UML diagram tried ovládača spektrometra Ibsen

Komunikácia so spektrometrom (`IbsenSpectrometerController`)

Komunikácia so spektrometrom je implementovaná v triede `IbsenSpectrometer-Controller`. Na komunikáciu s mikrokontrolérom *FT2232* je využitá knižnica `ftd2xx`, ktorá sa inštaluje automaticky pripojením zariadenia do USB portu. Hlavnými funkciami, ktoré sa z tejto knižnice využívajú k zápisu a čítaniu, sú `FT_Write()` a `FT_Read()`. V prípade odosielania, je potrebné funkcii `FT_Write()` odovzdať tzv. rukoväť (angl. handle), ktorá reprezentuje zariadenie, s ktorým sa má komunikovať. Ďalšími argumentami funkcie sú samotné dáta, ktoré sa majú poslať, ich veľkosť a ukazovateľ na premennú, do ktorej bude uložená veľkosť (v bytoch), ktorá sa skutočne odoslala. V prípade funkcie `FT_Read()` sa funkcii odovzdáva znovu rukoväť, ukazovateľ na premennú, kam sa uložia prijaté dáta. Ďalej je to očakávaná veľkosť prijatých dát a ukazovateľ na premennú, do ktorej bude uložená veľkosť dát, ktoré sa skutočne prijali.

Každý príkaz, či už na zápis, alebo čítanie musí obsahovať okrem adresy registra v spektrometri aj príkazy pre mikrokontrolér na nastavenie parametrov odosielania. Na tento účel slúži trieda `Command`, ktorá poskytuje tzv. *method chaining*[1]. Vďaka tomuto spôsobu práce s funkciami je možné vytvárať jeden zložený príkaz z viacerých atomických príkazov. Výhodou tohto spôsobu tvorby príkazov je, že je možné naraz poslať viac príkazov pomocou jediného volania funkcie `FT_Write()`. Tým sa ušetrí

[1]method chaining - je spôsob volania funkcie priamo na návratovú hodnotu inej funkcie

réžia mikrokontroléra spojená s komunikáciou cez USB rozhranie.

Na vyčítanie zachytených spektier zo spektrometra slúži trieda `ReadSpectra`. Na vyčítanie jedného spektra je potrebné vytvoriť toľko príkazov, z koľkých pixelov sa skladá detektor spektrometra. Informácia o tom, že zachytené spektrum je pripravené na vyčítanie, je indikované nastavením portu mikrokontroléra *GPIOL1* na úroveň logickej jednotky spektrometrom. Po zmene logickej úrovne signálu z nuly na jednotku, mikrokontrolér spracuje ďalšiu inštrukciu, ktorou je vyčítanie spektra. Dotazovanie sa na príchod nového spektra je možné pomocou funkcie `FT_GetQueueStatus()`, ktorá indikuje nové dáta vo fronte mikrokontroléra. Problémom pri implementácii bola skutočnosť, že nebolo možné vytvoriť príkaz na vyčítanie všetkých pixelov detektora naraz, kvôli obmedzenej veľkosti fronty. Z tohto dôvodu sa poslal jeden príkaz na vyčítanie jedného pixela a po potvrdení dostupnosti dát sa mohol poslať druhý príkaz na vyčítanie zvyšných pixelov. Po obdržaní hodnôt pixelov, je možné tieto dáta uložiť do premennej. V tomto prípade sa jedná o frontu. Vzhľadom na to, že je v triede `ReadSpectra` použité aktívne čakanie na prijaté dáta, musí byť táto trieda implementovaná ako samostatné vlákno tzv. viacvláknové programovanie (angl. *multithreading*[2]). V prípade, že by táto trieda nebola implementovaná ako samostatné vlákno, aplikácia by vyťažovala procesor na maximum. Zároveň by nebolo možné s aplikáciou komunikovať, čo by malo za následok vypnutie aplikácie operačným systémom. Použitím viacvláknového programovania, bolo nutné zabezpečiť zdieľanú premennú medzi vláknami. Konkrétne sa jedná o frontu, ktorú reprezentuje dátový typ `QQueue<float>()`. Tento dátový typ poskytuje použitá knižnica Qt. Aby bol prístup k fronte bezpečný z hľadiska viacvláknového programovania, bol využitý objekt `QMutex()`. Tento objekt ochraňuje zdieľanú frontu pred súčasným prístupom viacerých vlákien v rovnakom čase.

Prenos dát pomocou protokolu SDD

Zachytené dáta je potrebné ďalej preniesť pomocou protokolu SDD do grafického užívateľského rozhrania. Následne budú tieto dáta ďalej spracované, vizualizované a prípadne uložené na pevný disk. Odosielanie zachytených spektier sa vykonáva v triede `SendSpectra`. Trieda je rovnako, ako to bolo v prípade triedy `ReadSpectra`, implementovaná formou samostatného vlákna.

Odosielanie dát je implementované cyklickým vyčítaním zdieľanej fronty. Každých sto milisekúnd dochádza k vyprázdneniu celej fronty do dočasnej premennej. Ak sú parametre merania nastavené na režim akumulácie, resp. priemerovania spektier, požadovaný počet spektier sa sčíta (resp. spriemeruje). Výsledné dáta sú následne pripravené na odoslanie pomocou protokolu SDD. Odosielanie je uskutočnené

[2]multithreading je schopnosť procesora poskytovať viacero vlákien na vykonávanie výpočtov

prostredníctvom binárneho súboru. Binárny súbor, ako to už z názvu vyplýva, je určený na prenos binárnych dát. Zachytené spektrá, ktoré chceme odoslať, je preto potrebné vhodným spôsobom pretransformovať na binárnu formu. Spektrum je uložené v dátovej štruktúre typu vektor, ktorý obsahuje prvky typu float. Transformácia spektra na binárne dáta je pomerne jednoduchá pomocou typovej konverzie. Spôsob konverzie je uvedený vo výpise 4.1.

```
1 SDD::Data binaryData(reinterpret_cast<char*>(spektrum.data()),
2                      spektrum.size() * sizeof(float));
```

Výpis 4.1: Konverzia spektra na binárne dáta

Jeden odosielaný binárny súbor dát však môže obsahovať niekoľko spektier. Navyše, je potrebné súčasne preniesť metadáta, ktoré sú súčasťou každého spektra. Aby bolo možné binárny súbor spätne konvertovať na spektrá, je nutné pridať informáciu o počte spektier v danom súbore. Taktiež sa musia k súboru pripojiť aj metadáta. Z uvedených dôvodov bol na tvorbu binárneho súboru použitý formát zobrazený vo výpise 4.2.

```
1 POČET_SPEKTIER:SPEKTRUM_1...SPEKTRUM_X:METADATA_1/.../METADATA_X
```

Výpis 4.2: Formát odosielaného binárneho súboru

Ako je možno vidieť vo výpise 4.2, informácia o počte spektier a metadáta sú od samotných spektier oddelené znakom `":"`. Jednotlivé metadáta sú medzi sebou oddelené znakom `"/"`. Použitie týchto vybraných rozdeľovacích znakov umožní na strane príjemcu správne rozdelenie binárneho súboru na jednotlivé spektrá a k ním prislúchajúce metadáta.

4.1.2 Implementácia komunikácie medzi užívateľským rozhraním a SDD zariadením

Komunikácia s SDD zariadením je na strane užívateľského rozhrania implementovaná v triede `SpectraControllerMaster`. Vyhľadávanie SDD zariadení rieši trieda `Controller`, v ktorej sa periodicky dotazuje na stav siete. V prípade, že sa v sieti objaví nové dostupné SDD zariadenie, je toto zariadenie propagované do triedy `SpectraControllerMaster`.

Trieda `SpectraControllerMaster` si udržuje všetky dostupné SDD zariadenia vo svojej premennej. Na základe zvoleného zariadenia, ktoré chce užívateľ používať, sa v tejto premennej požadované zariadenie vyhľadá a pripojí sa k nemu. Samozrejme, požadované zariadenie sa musí nachádzať v premennej dostupných zariadení. Pripojenie spočíva vo vytvorení objektu vzdialeného SDD zariadenia.

Konkrétne v tomto prípade je to vytvorenie objektu triedy `IbsenRemote`. V prípade, že v budúcnosti bude implementovaná podpora ďalších spektrometrov, vytvorí sa objekt triedy podľa voľby požadovaného spektrometra. Okrem vytvoria objektu triedy `IbsenRemote` (resp. triedy iného spektrometra) sa vytvorí aj objekt triedy `AbstractSpectrometerRemote`. Táto trieda implementuje všetky vlastnosti, ktoré sú medzi rôznymi typmi spektrometrov rovnaké. Napriek tomu, že sa trieda `AbstractSpectrometerRemote` nazýva abstraktná, nejedná sa o abstraktnú triedu podľa definície objektovo orientovaného programovania. Práve z tohto dôvodu je možné vytvoriť objekt z tejto triedy. Používanie objektu abstraktného spektrometra značne zjednoduší a zredukuje množstvo zdrojového kódu.

Po vytvorení objektov vzdialených zariadení je možné pomocou nich ovládať zvolený typ spektrometra a prijímať zachytené spektrá. Ovládanie, resp. nastavovanie parametrov spektrometra je veľmi jednoduché pomocou príslušných funkcií vytvorených objektov vzdialeného zariadenia. Prijímanie zachytených dát je implementované pomocou tzv. *callback* funkcie (pozri oddiel 3.1.1) `fileNewSpectraChanged`. Po prijatí nových dát sa automaticky vyvolá táto funkcia, ktorá ďalej odovzdá prijaté binárne dáta funkcii `processNewSpectra()`. Úlohou tejto funkcie je transformovať binárne dáta na jednotlivé spektrá a k ním prislúchajúce metadáta. Binárne dáta sú transformované na základe formátu uvedeného vo výpise 4.2. Spracovanie prijatých binárnych dát je uvedené vo výpise 4.3.

```
1  void SpectraControllerMaster::processNewSpectra(SDD::Data spectra) {
2    // Parse data to get: number of spectra, spectra & metadata
3    std::size_t indexOfLength   = spectra.find_first_of(":");
4    std::size_t indexOfMetadata = spectra.find_last_of(":");
5
6    QString metadataStr = QString::fromStdString(
7                          spectra.substr(indexOfMetadata + 1));
8
9    // Get metadata
10   QStringList metadata = metadataStr.split('/');
11
12   // Remove last empty
13   metadata.removeLast();
14
15   // Get number of spectra
16   int numberOfSpectra = std::stoi(spectra.substr(0, indexOfLength);
17
18   // Get spectra block
19   spectra = spectra.substr(indexOfLength + 1, indexOfMetadata);
```

```
20
21    if (metadata.size() != numberOfSpectra)
22        emit newLogMessage("error", "Missing element in metadata.");
23
24    // Convert binary data to floats
25    const float* spectrum = reinterpret_cast<const float*>(
26                            spectra.c_str());
27    for (int i = 0, offset = 0; i < numberOfSpectra; i++) {
28        // Push float[] into vector
29        m_dataset->setRows(m_dataset->rows() + 1);
30        float* dataBlock = m_dataset->getDataBlock(
31                           0, m_dataset->rows() - 1);
32        for (unsigned int i = 0; i < m_pixelsCount; i++) {
33            dataBlock[i]             = spectrum[offset + i];
34            m_accumulatedSpectra[i] = m_accumulatedSpectra[i] +
35                                      static_cast<uint64_t>(spectrum[i]);
36        }
37
38        m_dataset->ungetDataBlock(dataBlock);
39        m_dataset->metadataSet(m_dataset->rows() - 1,
40                   "Index of measurement", metadata.at(i));
41        m_capturedSpectra++;
42        offset += m_pixelsCount;
43    }
44 }
```

Výpis 4.3: Konverzia binárnych dát na spektrá a metadáta

4.2 Vykresľovanie grafu

Na implementáciu čiarového grafu bola využitá knižnica Qt a programovacie jazyky QML a C++. V jazyku QML boli implementované všetky grafické elementy, ktoré po spojení tvoria graf. Jazyk C++ bol využitý na všetky potrebné výpočty. Prepojenie atribútov medzi QML a C++ je realizované pomocou tzv. *Q_PROPERTY*.[3] Využitie tohto spôsobu prepojenia umožňuje vykonávať rôzne výpočty ako napr. mapovanie dát na pixely alebo generovanie popisu osí v jazyku C++. Výsledky výpočtov je následne možné využiť v jazyku QML na prácu s grafickými elementami. V opačnom smere je možné napr. pozíciu kurzora zachytenú v QML elemente využiť v triede jazyka C++.

Graf je realizovaný ako modul vo forme dynamicky spojenej knižnice (skrátene DLL). Implementácia formou modulu bola zvolená hlavne z dôvodu, aby graf mo-

[3] Q_PROPERTY je makro, ktoré slúži na definíciu premenných, ktoré sa správajú ako členské premenné danej triedy, ale majú ďalšie vlastnosti.

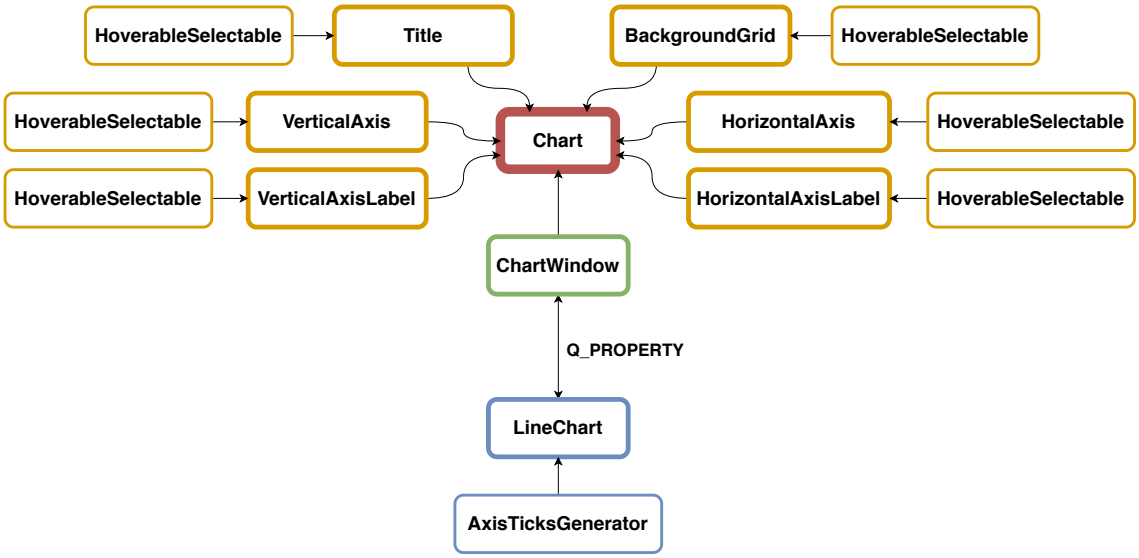
hol byť v budúcnosti využitý aj v iných programoch. V každej aplikácii, kde bude potrebné využiť graf, stačí jednoducho priložiť vygenerovaný DLL súbor spolu s hlavičkovým súborom.

Hlavným QML elementom je **`Chart.qml`**, ktorý spája všetky ostatné elementy, ktorými sú:

- **`Title.qml`** - názov grafu
- **`BackgroundGrid.qml`** - mriežka grafu
- **`VerticalAxis.qml`** - vertikálna os
- **`VerticalAxisLabel.qml`** - názov vertikálnej osi
- **`HorizontalAxis.qml`** - horizontálna os
- **`HorizontalAxisLabel.qml`** - názov horizontálnej osi
- **`ChartWindow.qml`** - okno grafu

Všetky elementy, okrem `BackgroundGrid.qml` a `ChartWindow.qml`, obsahujú ďalší sub-element **`HoverableSelectable.qml`**. Jeho významom je zvýraznenie daného elementu pri presune kurzora do jeho oblasti. V prípade zvýraznenia a následného dvojitého kliknutia na tento element sa otvorí dialógové okno s nastavením fontu písma.

Kompletná architektúra QML elementov a C++ tried je znázornená na obrázku 4.2.



Obr. 4.2: Architektúra QML elementov a C++ tried

4.2.1 Okno grafu (`ChartWindow.qml`)

Okno grafu slúži na vykresľovanie samotného grafu a umožňuje vykonávať operácie nad grafom (pozri podkapitolu 3.2). Tento element je prepojený s C++ triedou `LineChart`, v ktorej sú realizované všetky potrebné výpočty a aj samotné kreslenie jednotlivých čiar grafu. Dôležitou časťou okna grafu je tzv. `MouseArea`. Tento sub-element slúži na zachytávanie a obsluhu všetkých akcií vyvolaných užívateľom pomocou myši. Napríklad, ak si chce užívateľ priblížiť isté miesto v grafe, presunie kurzor na toto miesto a otočením kolieska myši vyvolá signál `onWheel`. Tento signál je ďalej vypropagovaný do C++ triedy `LineChart` spolu so súradnicami kurzora a uhlom, o ktorý sa pohlo koliesko myši.

4.2.2 Výpočty v triede C++ (`LineChart`)

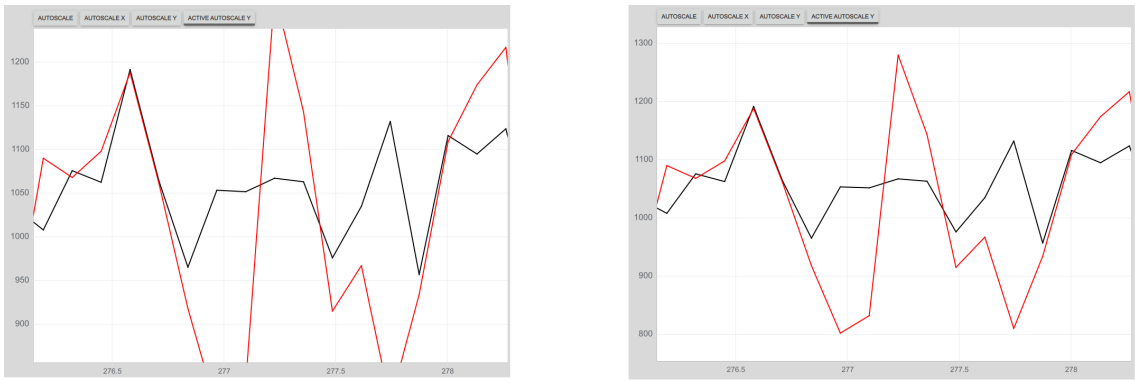
Všetky výpočty, potrebné na správne vykresľovanie a fungovanie grafu, sú realizované v C++ triede `LineChart`, ktorá dedí vlastnosti z triedy `QQuickPaintedItem`[4]. Trieda `QQuickPaintedItem` umožňuje kresliť pomocou triedy `QPainter` do QML scény.

Hlavnou funkciou triedy `LineChart` je funkcia `paint()`. Jedná sa o funkciu, ktorá prekrýva (angl. override) virtuálnu funkciu triedy `QQuickPaintedItem`. Vo funkcii `paint()` dochádza k vykresleniu úsečiek, ktoré sú definované dvojicou súradníc X a Y. Súradnice X a Y reprezentujú určitý bod, resp. pixel v QML grafickej scéne. Skupina vykreslených úsečiek zobrazuje priebeh grafu, v tomto prípade zachytené spektrum. Zároveň sa v tejto funkcii vygenerujú nové popisy oboch osí pomocou triedy `AxisTicksGenerator`, ktorá je podrobnejšie popísaná v oddieli 4.2.3. K prekresleniu scény, resp. k volaniu tejto funkcie dochádza vždy, pri akejkoľvek zmene v grafe (napr. zmena dát, priblíženie alebo posunutie zobrazenia grafu).

Ako už bolo spomenuté, približovanie a odďaľovanie grafu je možné ovládať kolieskom myši. Táto funkcia je implementovaná pod názvom `zoom()`. Vstupnými parametrami tejto funkcie sú súradnice kurzora myši, os alebo osi, na ktoré sa má aplikovať funkcia `zoom()` a faktor, čo je vlastne informácia o tom, či sa bude približovať, alebo odďaľovať. Súradnice kurzora myši určujú bod, ku ktorému sa bude graf približovať (resp. od ktorého sa bude odďaľovať). Vynásobením faktora s rozdielom maxima a minima aktuálneho zobrazenia oboch osí dostávame jednotku *delta*. Ďalej je potrebné vypočítať tzv. *percentáž*, ktorá zohľadňuje polohu kurzora myši a ovplyvňuje tým pozíciu, ku ktorej sa má graf priblížiť (resp. oddialiť). Pomocou získaných hodnôt je možné určiť nové minimum a maximum zobrazenia a prekresliť grafickú scénu.

[4]Podrobnejšie informácie o triede QQuickPaintedItem je možné nájsť v dokumentácii ku knižnici Qt dostupnej na: `https://doc.qt.io/qt-5/qquickpainteditem.html`

Posúvanie grafu je vyriešené vo funkcii `drag()`. Táto funkcia počíta rozdiel medzi pozíciou kurzora v momente stlačenia ľavého tlačidla myši a pozíciou kurzora v momente uvoľnenia tohto tlačidla. Pomocou vypočítaného rozdielu sa určí nové minimum a maximum zobrazenia. Nakoniec sa zavolá funkcia na prekreslenie grafickej scény.



Obr. 4.3: Aplikovanie funkcie automatického nastavenia osi Y. 1) pred aplikovaním funkcie `autoscale(Y)`; 2) po aplikovaní funkcie `autoscale(Y)`

Funkcia automatického nastavenia rozsahu osí je implementovaná pod menom `autoscale()`. Jediným vstupným parametrom je informácia, pre ktoré osi sa má vykonať automatické nastavenie rozsahu. V prípade, že bola zadaná os X alebo obe, dochádza k určeniu minimálnej a maximálnej hodnoty zobrazenia na osi X podľa prvej a poslednej hodnoty kalibrácie (pozri podkapitolu 3.2). V prípade, že bola zvolená os Y alebo obe, dochádza k vyhľadaniu minimálnej a maximálnej hodnoty intenzity vo všetkých zobrazených spektrách. Vyhľadávanie spomínaných hodnôt musí prebiehať iba v rozsahu aktuálneho zobrazenia grafu a nie z celého rozsahu spektier. V opačnom prípade by nastalo nesprávne nastavenie rozsahu osi Y. Na záver tejto procedúry sa pridá tzv. hranica, ktorá ma za účel zlepšiť prehľadnosť novonastaveného zobrazenia a grafická scéna sa prekreslí. Na lepšie pochopenie funkcionality slúži obrázok 4.3.

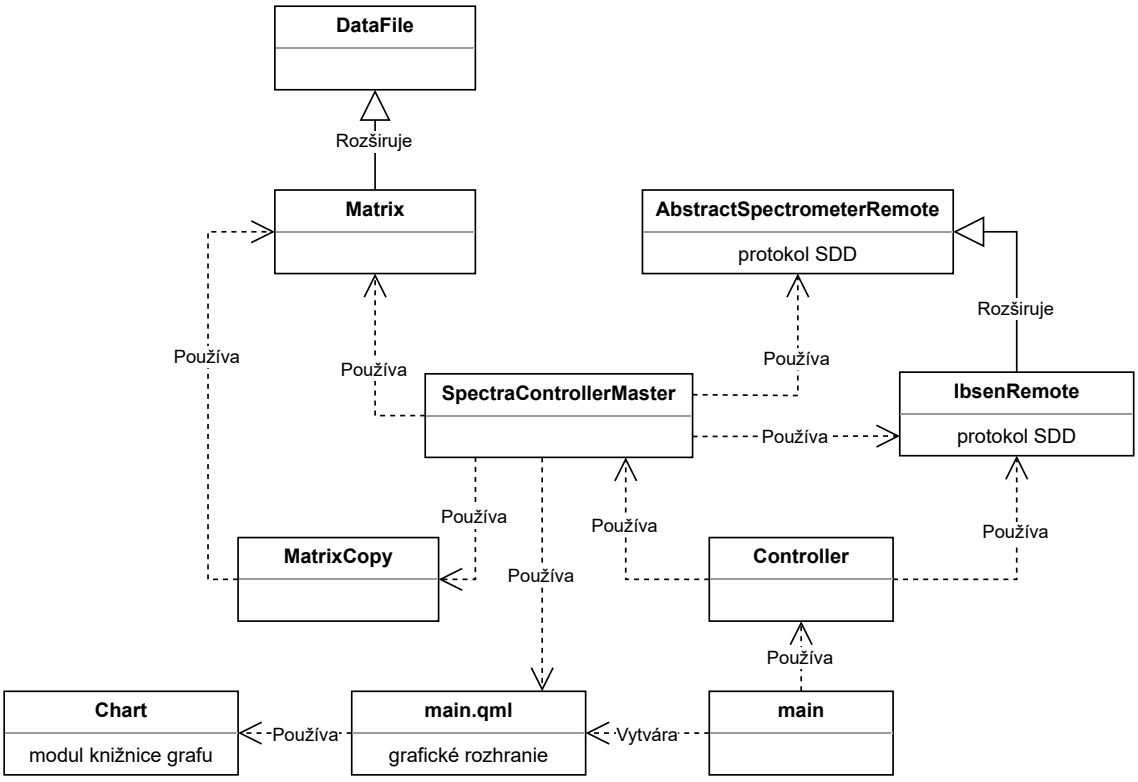
Aby bolo možné graf vykresliť a taktiež ním posúvať, približovať a pod., je potrebné zabezpečiť mapovanie X a Y súradníc medzi dátami (spektrami) a pixelmi v grafickej scéne. Na tento účel slúžia nasledovné štyri funkcie:

- `mapToX()` - prevod súradnice pixela na hodnotu spektra v osi X
- `mapFromX()` - prevod hodnoty spektra na súradnicu pixela v osi X
- `mapToY()` - prevod súradnice pixela na hodnotu spektra v osi Y
- `mapFromY()` - prevod hodnoty spektra na súradnicu pixela v osi Y

4.2.3 Automatické generovanie popisov osí (`AxisTicksGenerator`)

Ako už bolo spomenuté v podkapitole 3.2, je veľmi dôležité popísať osi grafu vhodnými hodnotami. Keďže sa zobrazenie grafu môže meniť, či už zásahom užívateľa (priblíženie, posun v grafe atď.), alebo vykreslením nového spektra, je dobré tieto popisy osí generovať automaticky. Na tento účel existuje trieda `AxisTicksGenerator` v jazyku C++. Vstupnými parametrami sú najmenší a najväčší (resp. prvý a posledný) bod osi. Na spracovanie hodnôt je potrebné použiť funkciu `processData()`. Následne je možné získať výstupné hodnoty generátora, ktorými sú nový prvý a posledný bod osi a medzera medzi jednotlivými dielikmi (angl. ticks) popisov osí. Na základe výstupov je možné popísať os grafu. Popis osi je možné vykresliť na základe dvoch vektorov typu `double`. Prvým z vektorov je `m_xAxisTicks`, v ktorom sú uložené číselné hodnoty jednotlivých dielikov osi. Druhý vektor má názov `m_xAxisTicksPosition` a určuje pozíciu (v pixeloch) jednotlivých dielikov v grafickej scéne.

4.3 Výsledné užívateľské rozhranie



Obr. 4.4: Zjednodušený UML diagram tried užívateľského rozhrania

Užívateľské rozhranie bolo implementované s využitím knižnice Qt a programovacích jazykov QML a C++. Práve implementácia grafického rozhrania v jazyku QML priniesla oveľa modernejší vzhľad oproti navrhnutému použitiu widgetov knižnice Qt (pozri obrázok 3.5). Funkcionalita užívateľského rozhrania však ostala rovnaká, ako bola navrhnutá (pozri podkapitolu 3.4) a je podrobnejšie popísaná v nasledujúcich oddieloch.

Štruktúra jednotlivých tried užívateľského rozhrania je zobrazená na zjednodušenom UML diagrame na obrázku 4.4.

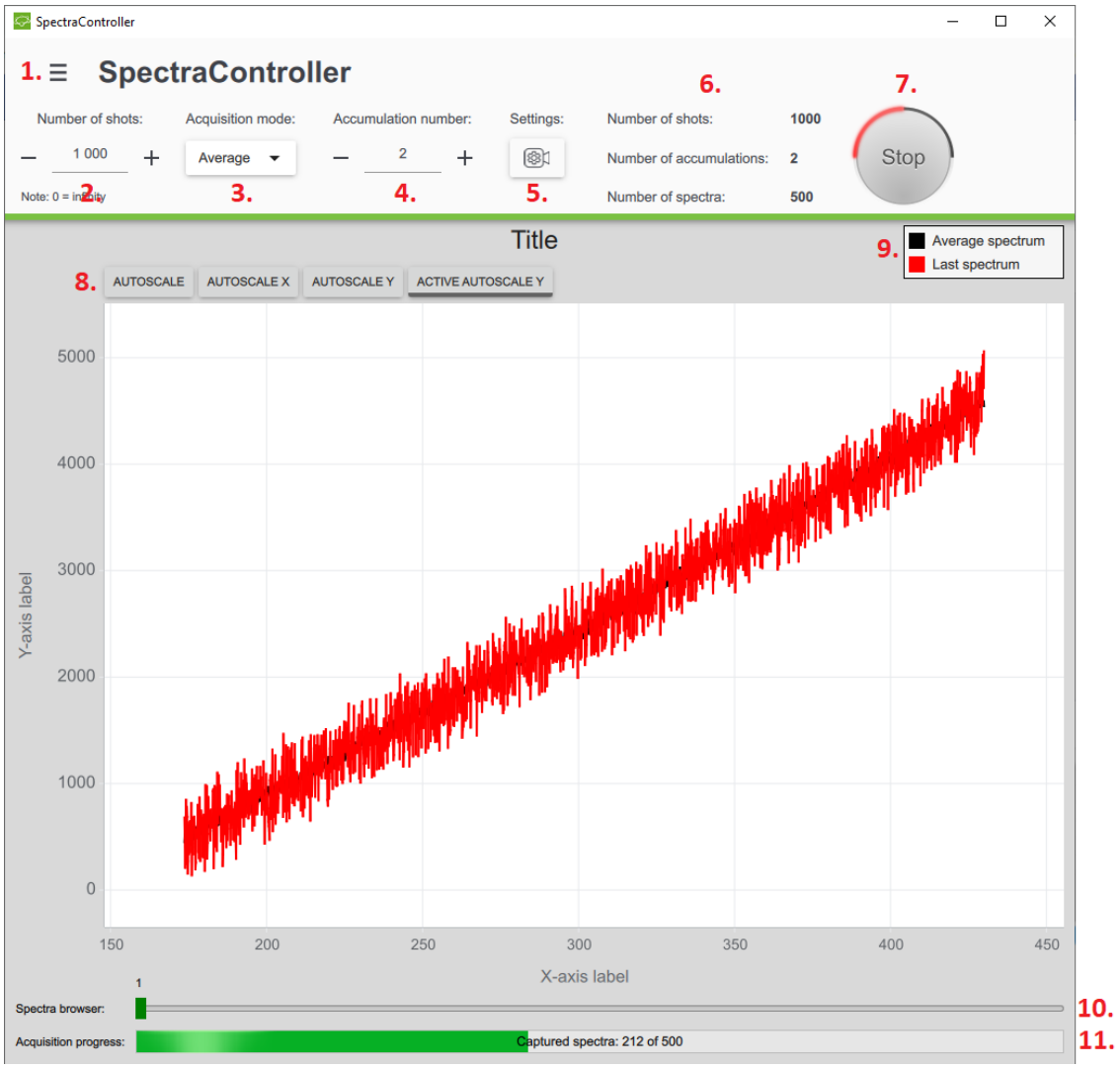
4.3.1 Grafická časť užívateľského rozhrania

Grafická časť užívateľského rozhrania je implementovaná v súbore s hlavným oknom `main.qml`. Hlavné okno slúži na vytvorenie všetkých ovládacích prvkov a taktiež na načítanie a zobrazenie knižnice grafu (pozri podkapitolu 4.2). Výsledné užívateľské rozhranie je zobrazené na obrázku 4.5.

Grafické užívateľské rozhranie je rozdelené na dve časti, ktoré sú medzi sebou oddelené vodorovnou zelenou linkou. Horná časť užívateľského rozhrania slúži na nastavovanie parametrov spektrometra a merania. Spodná časť okna slúži na vizualizáciu nameraných spektier a prípadné listovanie v nich. Význam a funkcionalita jednotlivých komponentov grafického rozhrania z obrázka 4.5 je nasledovná:

1. Tlačidlo zobrazenia hlavného menu.
2. Vstupné pole na zadávanie počtu výstrelov.
3. Rozbaľovací zoznam na výber módu merania.
4. Vstupné pole na zadávanie počtu akumulácií.
5. Tlačidlo na zobrazenie dialógového okna s nastavením detektora.
6. Prehľadová tabuľka merania podľa nastavených parametrov.
7. Tlačidlo na spustenie/zastavenie merania.
8. Tlačidlá automatického nastavenia rozsahu osí grafu.
9. Posuvná legenda grafu.
10. Posuvný jazdec (angl. slider) na voľbu zobrazenia spektra.
11. Ukazovateľ postupu (angl. progress bar) merania.

Aby nevznikli problémy s nepochopením funkcionality a významu jednotlivých ovládacích komponentov užívateľmi, boli použité nápovedy. Nápovedy sa v prostredí Qt a jazyka QML nazývajú *tooltips* a slúžia na vysvetlenie funkcionality daného ovládacieho prvku. Ak si užívateľ nie je istý významom niektorého prvku, stačí kurzorom myši vojsť do oblasti tohto komponentu a nápoveda sa po určitom čase sama zobrazí. Použitie nápovied má aj bezpečnostný charakter, pretože je možné predísť nepocho-



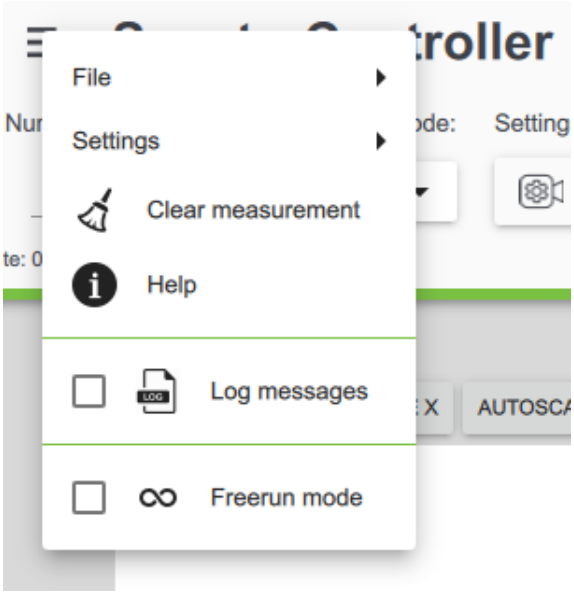
Obr. 4.5: Výsledné užívateľské rozhranie

peniu niektorej z funkcionalít aplikácie. Tým pádom sa môže predísť aj možnému poškodeniu prístrojov.

Hlavné menu aplikácie

Súčasťou grafického užívateľského rozhrania je menu, ktoré užívateľovi ponúka ďalšie voľby a možnosti pre prácu s aplikáciou. Po kliknutí na tlačidlo menu sa na tomto mieste rozbalí ponuka, ktorá je zobrazená na obrázku 4.6. Jednotlivé položky menu majú okrem textových štítkov aj ikony. Použitie ikon značne spríjemnilo výzor menu aplikácie a zjednodušilo prácu s ním.

Zaškrtávacia možnosť *Freerun mode* umožňuje aktivovať režim voľného merania. Tento režim nahrádza navrhovanú funkciu okamžitého zachytenia jedného spektra



Obr. 4.6: Hlavná ponuka menu

(pozri podkapitolu 3.4). Voľné meranie sa zvyčajne používa pri nastavovaní a kalibrovaní meracej zostavy. V režime voľného merania nedochádza k ukladaniu nameraných dát a taktiež sa neberú do úvahy žiadne nastavenia parametrov merania.

Zobrazenie informačných správ je aktívne po kliknutí na zaškrtávaciu možnosť *Log messages*. Informačné správy je vhodné využiť v prípadoch, keď dôjde k nejakým problémom s komunikáciou a pod. Vypisovanie správ je navyše farebne rozdelené podľa dôležitosti správ.

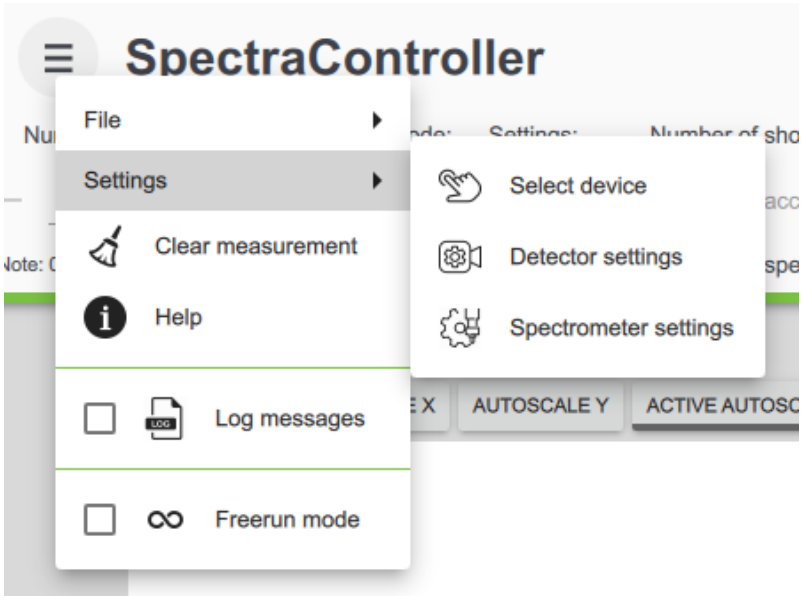
Na zobrazenie informácií týkajúcich sa aplikácie ako napríklad verzia alebo autor slúži tlačidlo *Help*.

V prípade, že užívateľ chce vymazať svoje posledné meranie, môže použiť tlačidlo *Clear measurement*. Pri použití tohto tlačidla, je užívateľ vyzvaný k potvrdeniu svojej voľby a je upozornený, že príde o svoje dáta.

Uloženie nameraných dát je dostupné v podmenu *File*, kliknutím na tlačidlo *Save*. Po kliknutí na toto tlačidlo sa zobrazí dialógové okno s výberom názvu súboru a cesty, kam sa namerané dáta majú uložiť.

Podmenu *Settings* zoskupuje možnosti, ktoré sa týkajú nastavení používaného zariadenia. Možnosti podmenu nastavení sú zobrazené na obrázku 4.7.

Voľba spektrometra, ktorý sa bude používať, je dostupná pod tlačidlom *Select device*. Po stlačení tlačidla sa zobrazí dialógové okno, ktoré je zobrazené na obrázku 4.8. Požadovaný typ spektrometra sa jednoducho vyberie pomocou rozbaľovacieho zoznamu. Po zvolení niektorého typu spektrometra sa zobrazí aj jeho fotografia. Užívateľ si tak môže byť úplne istý, že zvolil správny typ zariadenia, ktoré



Obr. 4.7: Podmenu nastavení

chce používať. Po odsúhlasení voľby zariadenia, aplikácia automaticky začne proces pripájania sa k zariadeniu prostredníctvom protokolu SDD (pozri oddiel 4.1.2).

Ďalšou položkou v podmenu nastavení sú nastavenia detektora, ktoré sú dostupné pod názvom *Detector settings*. Dialógové okno nastavení detektora je generované podľa typu zvoleného spektrometra. V tomto prípade sa jedná o dialógové okno nastavení detektora spektrometra Ibsen, ktoré je zobrazené na obrázku 4.9. V nastaveniach detektora je možné nastavovať napríklad dobu expozície, zosilnenie, typ spúšte atď.

Posledným elementom podmenu nastavení sú nastavenia spektrometra alebo *Spectrometer settings*. Nastavenia spektrometra slúžia na nastavenie parametrov ako napríklad zmena šírky vstupnej štrbiny, nastavenie pozadia a pod.

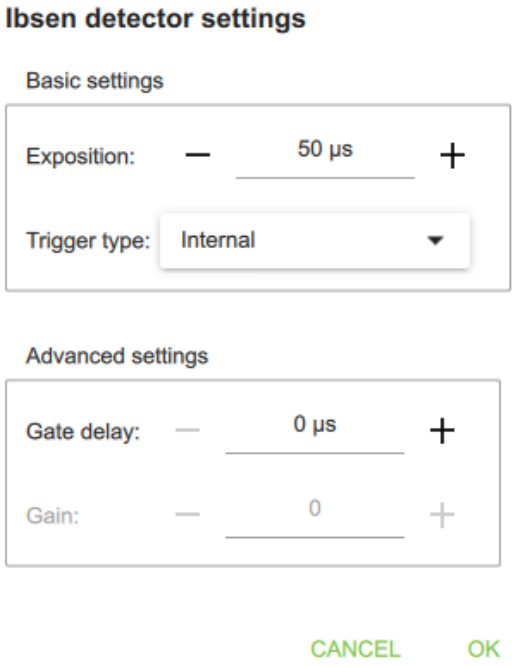
4.3.2 Funkčná časť užívateľského rozhrania

Prepojenie grafickej časti užívateľského rozhrania s poskytovaním funkcionality je realizované v triede `SpectraControllerMaster`. Ako už bolo spomenuté v oddieli 3.1.1, táto trieda slúži aj na komunikáciu s SDD zariadením. Všetky užívateľské vstupy, ktoré súvisia s ovládaním práce spektrometra sú práve v tejto triede spracované. Ak užívateľ v grafickom rozhraní zmení niektorú hodnotu, je táto hodnota prenesená do triedy `SpectraControllerMaster` pomocou už spomínaných *Q_PROPERTY* (pozri podkapitolu 4.2). Po prenesení novej hodnoty, je táto hodnota odoslaná SDD zariadeniu pomocou príslušnej funkcie daného parametra.

Obr. 4.8: Dialógové okno na voľbu spektrometra

V opačnom smere je potrebné zobraziť prijaté spektrum. Spracovaniu prijatých binárnych dát sa venuje oddiel 4.1.2. Spracované spektrá sú následne uložené do dočasného dátového súboru. Princíp a formát tohto súboru je uvedený v podkapitole 3.3. Ukladanie do dočasných súborov bolo zvolené hlavne z dôvodu, že je žiadúca možnosť prezerania zachytených spektier. Ukladanie dát do súboru je jednoduché. Prvým krokom je pridanie nového riadka, ktorý reprezentuje jedno spektrum. Následne je možné získať ukazovateľ na tento riadok pomocou funkcie `getDataBlock()`. Získaný ukazovateľ ukazuje na pole dátových typov *float*. Nové spektrum je možné do súboru uložiť nakopírovaním všetkých hodnôt spektra do tohto poľa. Po skončení práce s riadkom je potrebné zavolať funkciu `ungetDataBlock()`, ktorá uvoľní zabraný riadok súboru a uloží ho do pamäte. Okrem uloženia samotného spektra do súboru je potrebné uložiť aj metadáta. Uloženie metadát je možné pomocou funkcie `metadataSet()`. Zapisovanie nového prijatého spektra a metadát je uvedené vo výpise 4.3 funkcie `processNewSpectra()`, počnúc riadkom číslo 29 a končiac riadkom číslo 40.

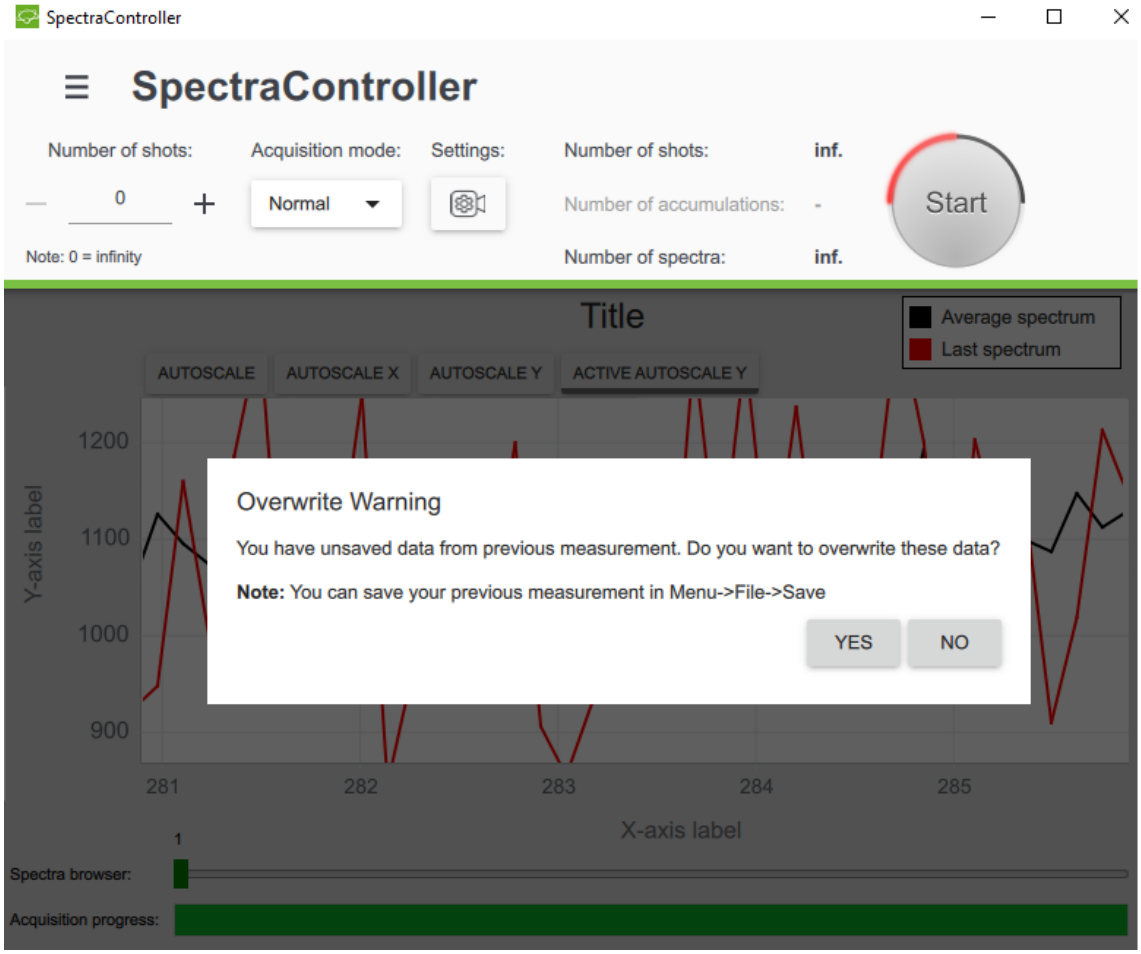
Každým spustením nového merania sa dočasný súbor s uloženými spektrami vynuluje. V prípade, že užívateľ namerané dáta potrebuje, môže si ich uložiť na zvolené miesto na disku, ako bolo spomenuté v oddieli 4.3.1. Môže ale nastať situácia, že si



Obr. 4.9: Dialógové okno nastavení detektora

užívateľ zabudne uložiť posledné meranie a o dáta v dočasnom súbore príde. Z tohto dôvodu bol implementovaný mechanizmus, ktorý upozorní užívateľa na neuložené dáta, ak hrozí ich strata. Princíp ochranného mechanizmu je nasledovný. Akonáhle sa spustí meranie, program si uloží informáciu, že existujú neuložené dáta. Ak si užívateľ po skončení merania svoje dáta uloží, táto informácia sa vymaže. V prípade, že si ale užívateľ svoje dáta neuloží a pokúsi sa spustiť nové meranie, zatvoriť aplikáciu alebo vyčistiť meranie, zobrazí sa dialógové okno s upozornením. Dialógové okno upozorní užívateľa, že môže prísť o svoje neuložené dáta. V poznámke je tiež uvedené, akým spôsobom je možné si svoje dáta uložiť. Dialógové okno s upozornením je zobrazené na obrázku 4.10.

Vzhľadom na to, že užívateľské rozhranie a ovládač spektrometra sú dva oddelené programy, je potrebné riešiť situáciu, ak sa aplikácia užívateľského rozhrania nečakane vypne. Je to dôležité hlavne z dôvodu, že k ukončeniu užívateľského rozhrania môže dôjsť aj počas spusteného merania. Ovládač spektrometra sa musí nejakým spôsobom dozvedieť o tom, že už neexistuje žiadny program, ktorý ho ovláda. Spoliehať sa na odoslanie signálu v deštruktore užívateľského rozhrania nie je rozumné. Aplikácia sa môže v prípade jej pádu ukončiť aj bez zavolania deštruktora. Z tohto dôvodu bol implementovaný spôsob na sledovanie, či je aplikácia užívateľského rozhrania aktívna. Tento spôsob dohľadu sa označuje aj pod anglickým pojmom *watchdog timer*. Funkcionalita spočíva v periodickom zasielaní signálov pomocou SDD



Obr. 4.10: Dialógové okno s upozornením na neuložené dáta

protokolu. Každým prijatím tohto signálu sa vynuluje časovač na strane ovládača. V prípade, že dôjde k vypršaniu časovača v ovládači, je užívateľské rozhranie považované za neaktívne a ovládač riadne ukončí svoje aktivity a vypne sa.

Užívateľom nemusí vždy vyhovovať forma dvoch oddelených aplikácií. Môže nastať prípad, že užívateľ chce vykonávať meranie na rovnakom zariadení, ku ktorému je aj pripojený spektrometer. V tomto prípade je možné použiť verziu aplikácie, ktorá automaticky spustí ovládač spektrometra pri voľbe zariadenia. Spustenie programu v rámci inej aplikácie je možné pomocou objektu triedy `QProcess`. Spôsob implementácie je uvedený vo výpise 4.4.

```
1 // Run driver & start checking for remote
2 QProcess* driver = new QProcess();
3 driver->start("Driver/Ibsen/Driver.exe", QStringList() << "");
```

Výpis 4.4: Spúšťanie programu vo vnútri iného programu

Aby bolo možné spustiť ovládač, musí sa nachádzať spustiteľný súbor ovládača v zložke programu užívateľského rozhrania. Konkrétne musí byť umiestnený v zložke *Driver/typ_spektrometra*, kde *typ_spektrometra* je typ konkrétneho spektrometra. Aplikácia, ktorá je v prílohe tejto práce, je tiež vo verzii, kde sa ovládač spektrometra spúšťa automaticky.

Vzhľadom na to, že na spustenie ovládača je potrebné mať pripojený spektrometer, je v prílohe tejto práce mierne pozmenená verzia aplikácie. Táto verzia umožňuje spustenie ovládača aj bez pripojeného spektrometra. Navyše, po spustení merania sú generované náhodné dáta frekvenciou $100\,Hz$. Pozmenená verzia aplikácie slúži na demonštráciu funkcionality implementovaného systému bez nutnosti vlastniť, resp. použiť fyzické zariadenie.

5 Testovanie

Záverečným bodom tejto práce je otestovanie výsledného softvéru. Priebeh testovania sa dá rozdeliť na dve základné časti. Prvou časťou je testovanie jednotlivých častí aplikácie počas implementácie. Účelom takéhoto testovania je odhaliť chyby a problémy v implementácií. Odhalené chyby sa v tomto štádiu dajú jednoducho opraviť alebo vyladiť. V prípade, že by sa chyby odhalili až v procese testovania hotovej aplikácie, bolo by omnoho ťažšie takéto chyby opraviť. Po overení správnosti programu je následne potrebné vykonať testovanie v reálnych podmienkach. Počas testovania programu na skutočných zariadeniach treba overiť, či hotová aplikácia spĺňa kritériá, ktoré boli stanovené v zadaní práce.

5.1 Testovanie počas implementácie

Testovanie jednotlivých častí aplikácie prebiehalo priebežne počas celej doby implementácie. Týmto spôsobom testovania sa podarilo odhaliť množstvo problémov, ktoré by mohli znižovať kvalitu výslednej aplikácie. Priebeh testovania jednotlivých častí je popísaný v nasledujúcich oddieloch.

5.1.1 Testovanie ovládača spektrometra

V tejto časti testovania bolo nutné zamerať sa na overenie správnosti komunikácie so spektrometrom. Kľúčovou vlastnosťou, ktorú bolo potrebné otestovať, bola rýchlosť vyčítania spektier. Princíp testovania spočíval v postupnom zvyšovaní frekvencie pulzov, ktoré reprezentovali výstrely lasera. Vyčítanie prebiehalo bezchybne približne do frekvencie $50\,Hz$. Zvyšovaním frekvencie nad $50\,Hz$ už dochádzalo k stratám niektorých spektier. Počas testovania bolo zistené, že limitujúca je veľkosť fronty mikrokontroléra *FT2232*. Ako bolo uvedené v oddieli 4.1.1, nie je možné naraz vyčítať všetky hodnoty pixelov detektora. Analýzou USB prenosu bolo zistené, že vytvorenie každého príkazu na vyčítanie pixelov zo spektrometra má veľkú réžiu, ktorá celý proces spomaľuje. Aby sa rýchlosť vyčítania mohla zvyšovať, bolo by nutné posielať požiadavku na vyčítanie viacerých spektier v rámci jediného príkazu mikrokontroléru. Toto však nie je možné vzhľadom na spomínanú obmedzenú veľkosť fronty mikrokontroléra. Jedná sa teda o problém v návrhu dodaného zariadenia na prevod medzi USB a SPI rozhraním. Riešením by mohlo byť použiť iný typ mikrokontroléra na prevod medzi rozhraniami.

5.1.2 Testovanie prenosu nameraných dát protokolom SDD

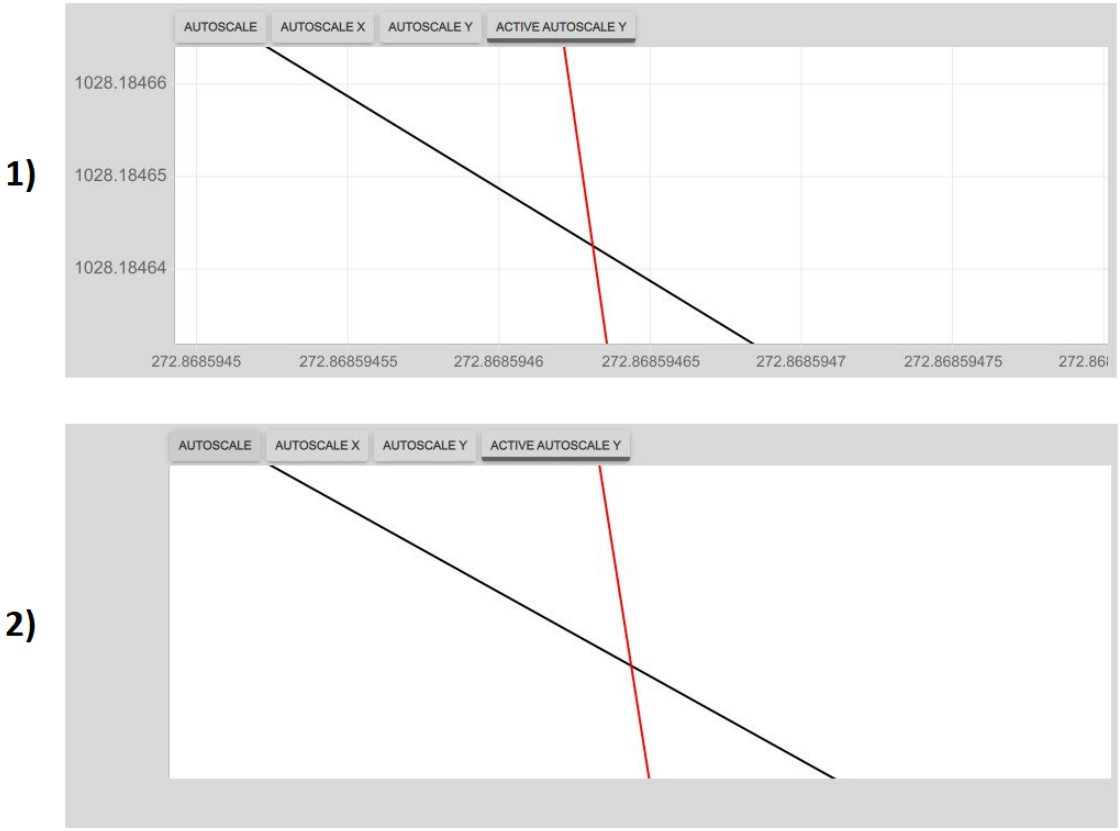
S maximálnou frekvenciou zachytávania spektier súvisí, okrem rýchlosti vyčítania dát zo spektrometra, aj prenos binárnych dát protokolom SDD (pozri oddiel 2.3.3). Ako bolo uvedené v oddieli 4.1.1, vyčítanie zdieľanej fronty a následné odoslanie dát prebieha raz za sto milisekúnd. Pôvodne bolo odosielanie implementované takým spôsobom, že získané spektrum bolo ihneď konvertované na binárne dáta a odoslané. Počas testovania odosielania však bolo zistené, že už pri frekvencii generovania dát $20\,Hz$ dochádzalo k výpadkom niektorých spektier. Analýzou sieťového toku bolo odhalené, že pri takomto veľkom množstve generovaných paketov dochádza k ich zahadzovaniu operačným systémom. Práve z tohto dôvodu bol zvolený spomínaný spôsob odosielania dát v časových intervaloch a za pomoci zdieľanej fronty. Časový interval odosielania bol zvolený postupným zvyšovaním tejto hodnoty, až do doby $80\,ms$, kedy už nedochádzalo k zahadzovaniu paketov. Pre istotu bol tento časový interval ešte navýšený o hodnotu $20\,ms$. Výsledný časový interval medzi odosielaním dát je teda $100\,ms$. Vzhľadom na to, že $100\,ms$ predstavuje frekvenciu $10\,Hz$, je táto hodnota prijateľná aj pre užívateľa. Vyššiu frekvenciu obnovovania spektier v grafe by užívateľ aj tak nepostrehol.

Po aplikovaní spomínaného spôsobu odosielania dát v časových intervaloch bol vykonaný test na určenie maximálnej prípustnej frekvencie generovania dát. Vzhľadom na to, že ovládač spektrometra zvládal frekvenciu iba do $50\,Hz$, boli generované náhodné dáta. Generovaním náhodných dát bola simulovaná funkcia spektrometra. Testovanie prebiehalo postupným zvyšovaním frekvencie generovania dát a sledovaním správnosti prijatých dát v užívateľskom rozhraní. Náhodne generovaná hodnota bola z rozsahu 0 až 5000 a počet pixelov bol 2048. Výsledok testovania je, že prenos náhodných dát fungoval bezchybne do frekvencie generovania $1000\,Hz$. Pri takto vysokej frekvencii však už bolo možné pozorovať drobné spomalenia, resp. zasekávania sa užívateľského rozhrania. Na druhej strane hodnota $1000\,Hz$ je ďaleko nad požadovanou hodnotu opakovanej frekvencie $50\,Hz$. Na základe získaných výsledkov z testovania, možno usúdiť, že prenos spektier protokolom SDD je pripravený zvládnuť vysoké opakovacie frekvencie.

5.1.3 Testovanie zobrazovania grafu

V prípade zobrazovania grafu, bol testovaním odhalený problém s automatickým generovaním popisov osí. Problém spočíval v nepresnostiach pri generovaní desatinných čísiel do popisov osí. V prípade, že si užívateľ príliš priblížil isté miesto v grafe, dochádzalo k tvorbe popisov osí z desatinných čísiel. Ak sa toto číslo skladalo zo štyroch a viac desatinných miest, dochádzalo k nepresnostiam pri počítaní s takýmito číslami. Následkom nepresností vo výpočtoch bol neočakávaný pád aplikácie. Z tohto

dôvodu bolo pridané zaokrúhľovanie výsledkov pri počítaní s dvoma desatinnými číslami. Pridaním zaokrúhľovania výsledkov sa odstránil problém s neočakávaným pádom aplikácie. Maximálny limit počtu desatinných miest v generovanom popise osí sa zvýšil na sedem desatinných miest. Zároveň, po prekročení tohto limitu sa už ďalej negenerujú nové popisy osí, ale graf a celá aplikácia ostáva naďalej funkčná. Popísané ošetrenie daného problému je vyhovujúce, vzhľadom na to, že užívateľ nepotrebuje väčšiu presnosť zobrazenia. Ošetrenie limitu automatického generovania popisov osí znázorňuje obrázok 5.1.



Obr. 5.1: Ošetrenie limitu automatického generovania popisov osí grafu. 1) Okno grafu s hraničnými hodnotami popisov osi; 2) Okno grafu bez popisov osí

5.1.4 Testovanie užívateľského rozhrania

Overovanie správnej funkcionality užívateľského rozhrania spočívalo vo vyskúšaní všetkých ovládacích prvkov. Počas testovania sa odhalilo množstvo drobných problémov, ktoré mohli nastať nepozornosťou pri implementácií. Po overení funkcionality a opravení prípadných nedokonalostí, bolo užívateľské rozhranie testované osobami,

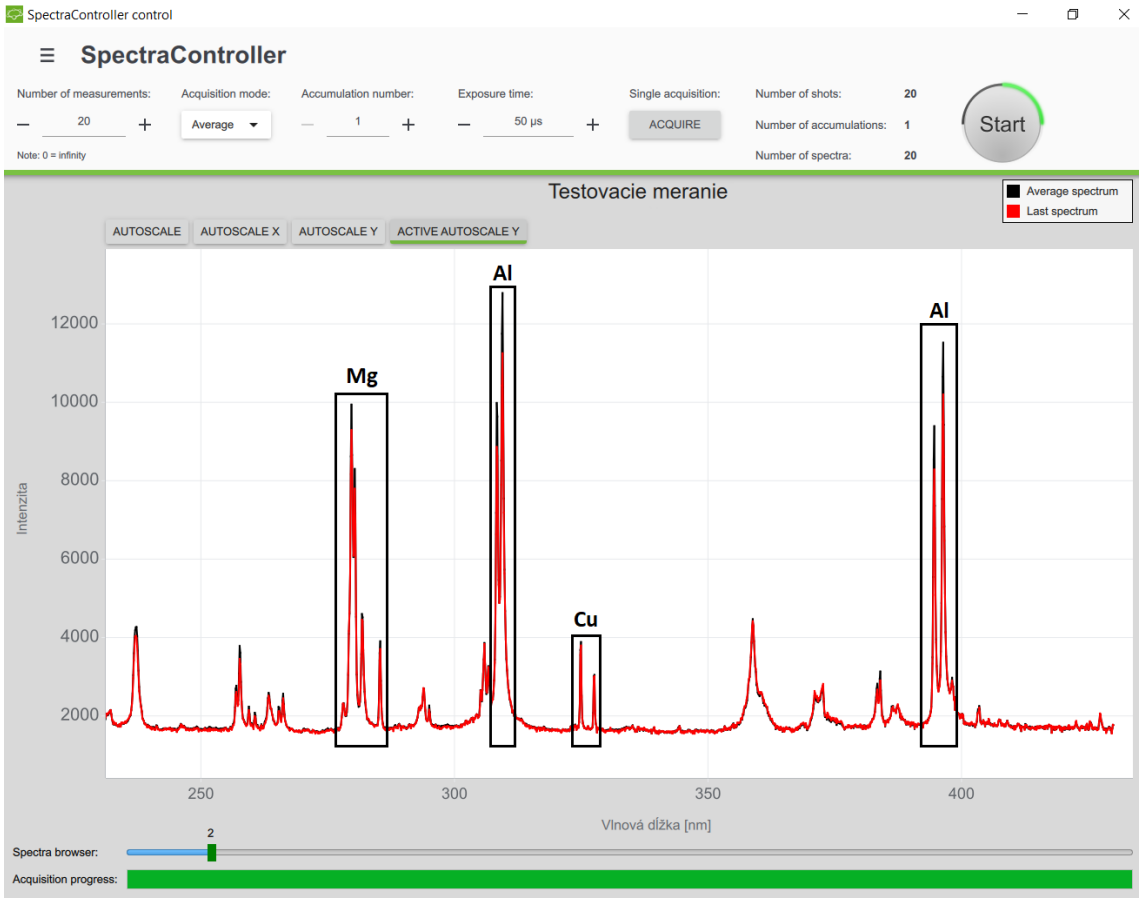
ktoré ho v budúcnosti budú používať. Konkrétne sa jednalo o kolegov vedcov a laborantov, ktorí nemusia mať znalosti z oblasti programovania softvéru. Na druhej strane majú skúsenosti s používaním rôznych iných programov podobného zamerania a ich spätná väzba k tejto aplikácií môže byť veľmi hodnotná. Hodnotenia testovacích užívateľov boli veľmi pozitívne. Za najväčšiu výhodu bola označená jednoduchosť používania a minimalistický vzhľad grafického rozhrania. Pozitívne ohlasy mal aj moderný dizajn aplikácie. Nedostatky v aplikácii neboli podľa testovacích užívateľov takmer žiadne až na malé drobnosti. Drobné nedostatky boli hneď opravené. Týkali sa hlavne pozície niektorých ovládacích prvkov, ich veľkosti a popisu. Skupina skúsenejších užívateľov by ocenila pridanie ďalších funkcií ako napríklad vyhľadávanie maxima v spektre alebo export nameraných dát do textového formátu. Implementáciu uvedených prídavných funkcií by bolo určite vhodné zvážiť v budúcnosti vývoja tejto aplikácie.

5.2 Konečné testovanie v reálnych podmienkach

Úplne posledným krokom tejto práce bolo otestovanie implementovaného programu v skutočných podmienkach. Testovanie prebiehalo v laboratóriu laserovej spektroskopie na Fakulte strojního inženírství VUT v Brne. Zostava na testovanie pozostávala z tzv. *Q-SWITCH* lasera, digitálneho generátora pulzov, spektrometra Ibsen a počítača na ovládanie jednotlivých zariadení. Vzhľadom na to, že maximálna frekvencia merania je limitovaná nesprávnym návrhom USB-SPI prevodníka (pozri oddiel 5.1.1) bola frekvencia pulzov nastavená na hodnotu $50\,Hz$. Energia lasera bola nastavená na hodnotu $10\,mJ$ a expozičná doba detektora bola $50\,\mu s$. Na meranie bola použitá vzorka s certifikovaným zložením *BAM-311 ($AlCuMg_2$)* zobrazená na obrázku 5.2.

Obr. 5.2: Vzorka s certifikovaným zložením

Meraním vzorky štandardu je možné porovnať namerané spektrá s hodnotami, ktoré sú uvedené v dokumentácii použitého štandardu. Na obrázku 5.3 je zachytené užívateľské rozhranie počas merania hliníkového štandardu.



Obr. 5.3: Ukážka užívateľského rozhrania z reálneho merania

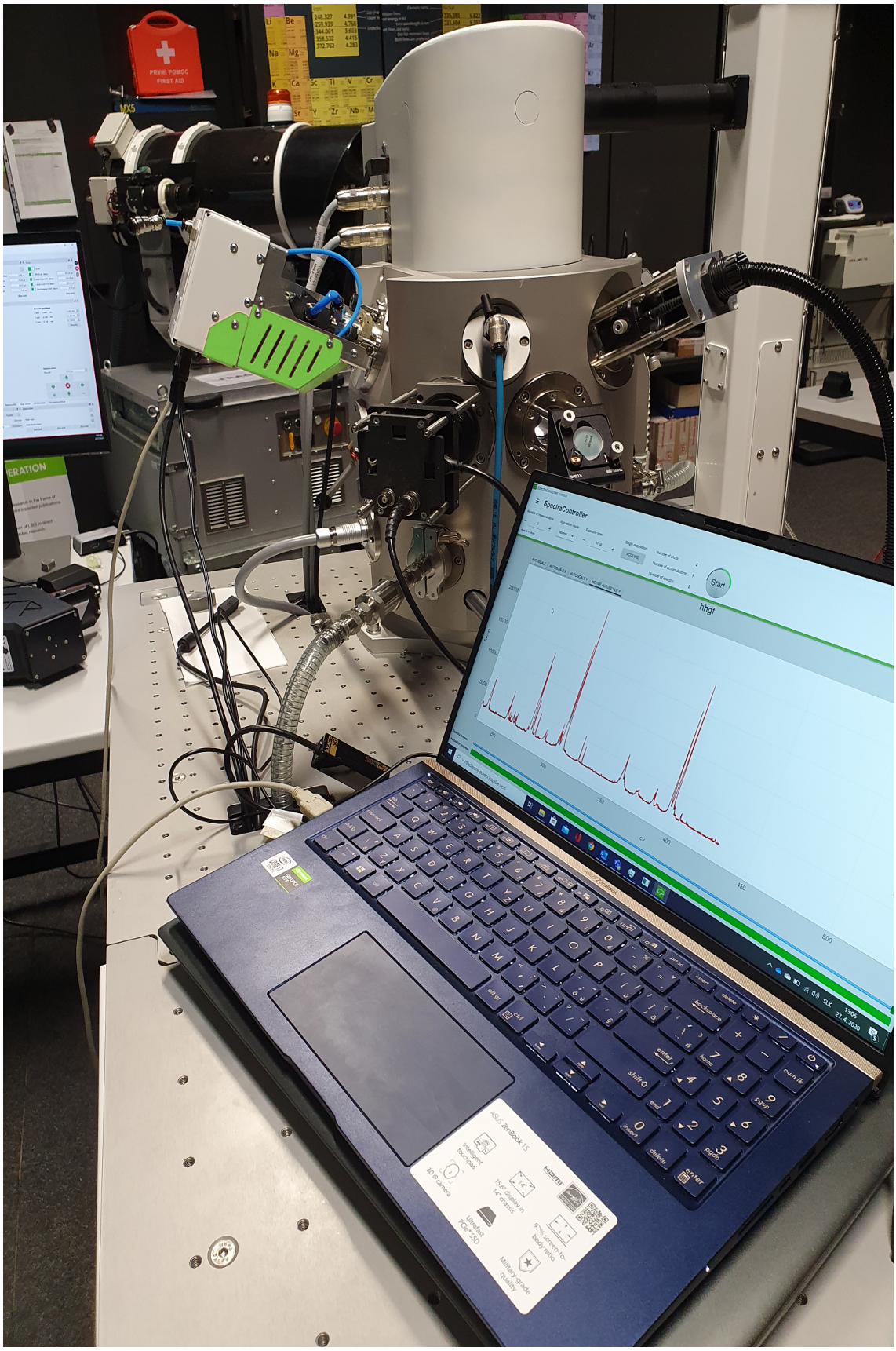
Použitý štandard sa skladá z troch základných chemických prvkov a to z hliníka, medi a horčíka. V špeciálnej periodickej tabuľke prvkov pre laserovú spektroskopiu je možné vyhľadať vlnové dĺžky vyžarovaného svetla pre jednotlivé prvky. Pre hliník je to približne 308 nm až 309 nm a 394 nm až 396 nm. V prípade medi je to 324 nm až 327 nm. Emisivita horčíka je v rozsahu 279 až 280. Porovnaním uvedených hodnôt s hodnotami vyznačenými na obrázku 5.3 je možné skonštatovať, že implementovaná aplikácia funguje správne.

Okrem správnej reprezentácie odmeraných hodnôt sa ďalej testovalo, či program zvláda maximálnu frekvenciu, ktorú umožňuje USB-SPI prevodník. Opakovaciu frekvenciu 50 Hz aplikácia zvládala bez akýchkoľvek problémov.

Ďalším bodom testovania bolo odskúšanie funkcionality všetkých ovládacích prvkov, ako napríklad nastavenie expozičnej doby, doby oneskorenia atď. Po skúške ovládacích elementov je znovu možné skonštatovať bezchybnú funkčnosť aplikácie.

Posledným krokom overovania funkčnosti bolo uloženie nameraných dát na pevný disk. Kontrolou správnosti bolo použitie vhodného softvéru, ktorý dokáže pracovať s rovnakým typom súborov s príponou *.libsdata* a *.libsmetadata* (pozri oddiel 3.3). Načítanie uložených dát prebehlo úspešne a zobrazené spektrá boli totožné so spektrami z merania.

Na obrázku 5.4 je zachytené pracovisko s inštrumentáciou počas testovania hotovej aplikácie.



Obr. 5.4: Ukážka pracoviska počas reálneho merania a testovania aplikácie

Záver

Cieľom tejto diplomovej práce bolo preštudovať problematiku inštrumentácie používanej v spektroskopii laserom budenej plazmy. S tým súviselo spracovanie problematiky najnovších trendov, ktoré sú žiadané v tomto odbore. Na základe týchto poznatkov bolo možné zvoliť vhodné technológie, ktoré budú použité na výslednú realizáciu práce. Zvolené technológie bolo nutné taktiež vhodne naštudovať, aby sa s nimi dalo efektívne pracovať. Po spracovaní všetkých informácií bolo možné prejsť k návrhu riešenia práce.

V prvom kroku bolo preštudovaných množstvo vedeckých článkov a kníh týkajúcich sa laserovej spektroskopie. Na základe týchto informácií bolo zistené, že medzi trendy v oblasti laserovej spektroskopie patrí zvyšovanie opakovacej frekvencie výstrelov. Zvyšovanie opakovacej frekvencie však prináša množstvo problémov, na základe ktorých boli definované potreby na implementáciu (podkapitola 1.3).

V druhej fáze tejto práce sa na základe získaných poznatkov zvolili technológie, ktoré sú použité na implementáciu výsledného softvérového rozhrania. V prvom rade bola preštudovaná literatúra, ktorá sa týkala knižnice Qt (podkapitola 2.2). Následne bol preskúmaný protokol Softvérovo definovaných zariadení (podkapitola 2.3). Keďže sa nejedná o štandardizovaný protokol, bolo kľúčové si podrobne prejsť a vyskúšať všetky možnosti, ktoré tento protokol poskytuje. Pomocou nadobudnutých informácií sa už mohol začať formovať návrh riešenia.

V ďalšom kroku bolo navrhnuté riešenie výslednej aplikácie s ohľadom na všetky aspekty, ktoré vyplývajú z predchádzajúcich bodov. Jednou z hlavných častí bol návrh výsledného grafického rozhrania. Nemenej dôležitými súčasťami boli návrh vykresľovania grafu odmeraných spektier a návrh spôsobu, akým sa budú ukladať zachytené dáta na pevný disk.

Hlavným bodom práce bola implementácia aplikácie podľa zadania. Riešením tejto práce je spojenie teoretických poznatkov a návrhu riešenia pomocou zvolených implementačných technológií. Počas implementácie došlo k istým zmenám oproti návrhu. Hlavnou modifikáciou bola zmena grafického vzhľadu užívateľského rozhrania použitím jazyka QML.

Poslednou časťou diplomovej práce bolo testovanie aplikácie. Prvou fázou bolo testovanie počas implementácie, v rámci ktorej bolo odhalených množstvo problémov. Väčšina týchto problémov bola odstránená resp. opravená. Jediným nedostatkom, ktorý nebolo možné vyriešiť, je problém s opakovacou frekvenciou spektrometra. V tomto prípade sa však jedná o nesprávny návrh USB-SPI prevodníka.

Záverečnou fázou bolo otestovanie implementovaného riešenia v reálnych podmienkach. Testovanie prebiehalo v laboratóriu laserovej spektroskopie na Fakulte strojního inženýrství VUT v Brne. Na testovanie bola použitá vzorka štandardu,

podľa ktorého sa dala overiť korektnosť zobrazených dát aplikáciou.

Ciele práce, ktoré boli dopredu stanovené, sa podarilo splniť. Výsledkom práce je analýza, podrobný návrh riešenia, implementácia a otestovanie funkčnosti univerzálneho softvérového rozhrania pre detekčné jednotky používané v laserovej spektroskopii. Konečná aplikácia je schopná pracovať v požadovaných opakovacích frekvenciách a poskytuje intuitívne ovládanie, ktoré ocenia budúci užívatelia tohto programu.

Do budúcnosti by bolo určite vhodné zamyslieť sa nad možnosťou pridania ďalších funkcií podľa potrieb koncových užívateľov. Taktiež bude v budúcnosti nevyhnutné pridanie podpory ďalších typov spektrometrov. Pridanie nových typov zariadení by však nemal byť problém vzhľadom na návrh aplikácie, ktorý s touto možnosťou počítal.

Literatúra

[1] Anabitarte, F.; Cobo, A.; López-Higuera, J.: Laser-Induced Breakdown Spectroscopy: Fundamentals, Applications, and Challenges. *ISRN Spectroscopy*, 10 2012, doi:10.5402/2012/285240.

[2] Belica, M.: *Analyzátor signálu založený na DVB-T USB tuneru*. Bakalářská práce, Vysoké učení technické v Brně. Fakulta informačních technologií. Ústav počítačové grafiky a multimédií, Brno, 2017.
URL `http://hdl.handle.net/11012/69853`

[3] Bette, H.; Noll, R.; Müller, G.; aj.: High-speed scanning laser-induced breakdown spectroscopy at 1000 Hz with single pulse evaluation for the detection of inclusions in steel. *Journal of laser applications*, ročník 17, č. 3, 2005: s. 183–190.

[4] Cebo, P.: *Zvýšení odolnosti komunikační architektury pro dálkovou spektroskopii*. Bakalářská práce, Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací, Brno, 2018.
URL `http://hdl.handle.net/11012/82369`

[5] Cremers, D. A.; Radziemski, L. J.: History and fundamentals of LIBS. *Laser Induced Breakdown Spectroscopy: Fundamentals and Applications*, 2006: s. 9–16.

[6] Eng, L.: *Qt5 C++ GUI Programming Cookbook*. Packt Publishing, 2016, ISBN 9781783280278.

[7] Glassner, A.: *Graphics Gems*. Graphics Gems - IBM, Elsevier Science, 1990, ISBN 9780122861666.
URL `https://books.google.de/books?id=fvA7zLEFWZgC`

[8] Hang, Y.: *Time-Delay-Integration CMOS Image Sensor Design For Space Applications*. Ph.d. dissertation, Dept. Elect. Eng., Nanyang Technological Univ., Singapore, 2016.

[9] Lazar, G.; Penea, R.: *Mastering Qt 5*. Packt Publishing, 2016, ISBN 9781786467126.

[10] Litwiller, D.: Ccd vs. cmos. *Photonics spectra*, ročník 35, č. 1, 2001: s. 154–158.

[11] Miziolek, A. W.; Palleschi, V.; Schechter, I.: *Laser induced breakdown spectroscopy*. Cambridge university press, 2006.

[12] Noll, R.: *Laser-Induced Breakdown Spectroscopy*. Springer-Verlag Berlin Heidelberg, 01 2012, doi:10.1007/978-3-642-20668-9.

[13] Novotný, J.: *Dálkově řízená laserová spektroskopie (LIBS)*. Disertační práce, Vysoké učení technické v Brně. Fakulta strojního inženýrství. Ústav fyzikálního inženýrství, Brno, 2012.
URL `http://hdl.handle.net/11012/18045`

[14] Singh, J.; Thakur, S.: *Laser-Induced Breakdown Spectroscopy*. Elsevier Science, 2007, ISBN 9780080551012.
URL `https://books.google.cz/books?id=miDNbby3BPsC`

Zoznam skratiek

A/D	Analógovo-digitálny – Analog to Digital
ASCII	Americký štandardný kód na výmenu informácií – American Standard Code for Information Interchange
AES	Atómová emisná spektroskopia – Atomic Emission Spectroscopy
CCD	Nábojovo viazaná štruktúra – Charge-coupled device
CEITEC	Stredoeurópsky technologický inštitút – Central European Institute of Technology
CMOS	Komplementárny kov-oxid-polovodič – Complementary Metal Oxide Semiconductor
DLL	Dynamicky spojená knižnica – Dynamic Link Library
HTTP	Hypertextový prenosový protokol – HyperText Transfer Protocol
IP	Internetový protokol – Internet Protocol
JSON	JavaScript Object Notation
LIBS	Spektroskopia laserom budenej plazmy – Laser Induced Breakdown Spectroscopy
PDA	Pole fotodiód – Photo Diode Array
PMT	Fotonásobič – Photo Multiplier Tube
RTOS	Operačný systém reálneho času – Real-Time Operating System
SDD	Softvérovo definované zariadenie – Software Defined Device
SPI	Sériové periférne rozhranie – Serial Peripheral Interface
TCP	Protokol riadenia prenosu – Transmission Control Protocol
UDP	Používateľský datagramový protokol – User Datagram Protocol
UML	Jednotný modelovací jazyk – Unified Modeling Language
USB	Univerzálna sériová zbernica – Universal Serial Bus
QML	Qt modelovací jazyk – Qt Modeling Language

Zoznam príloh

A Obsah priloženého pamäťového média

K tejto práci je priložené CD, ktorého obsah je rozdelený do nasledujúcich adresárov:

- `CD`
  - `App` ..... adresár s aplikáciou
    - `Source` ..... zdrojové súbory a knižnice potrebné na kompiláciu
      - `Driver` ..... zdrojové súbory ovládača
      - `GUI` ..... zdrojové súbory grafického užívateľského rozhrania
      - `SDDceitec` ..... knižnica zariadení protokolu SDD
      - `SDDcommon` ..... knižnica zariadení protokolu SDD
      - `SDDcore` ..... knižnica jadra protokolu SDD
      - `Tools` ..... zdrojové súbory podporných nástrojov pre SDD
    - `Executable` ..... obsahuje spustiteľnú aplikáciu
  - `Latex` ..... zdrojové súbory textu tejto práce v jazyku LaTeX
  - `Text` ..... text tejto práce vo formáte PDF

B Kompilácia a spustenie

B.1 Kompilácia zdrojových súborov

Na kompiláciu zdrojových súborov je najvhodnejšie mať nainštalovaný program *Qt Creator*. Jedná sa o bezplatné vývojové prostredie určené pre prácu s knižnicou *Qt*. Táto práca bola vyvíjaná pod verziou *Qt 5.14.1*, na operačnom systéme Windows a pomocou kompilátora *MSVC2017 64bit*.

Ovládač a grafické užívateľské rozhranie sú vo forme oddelených projektov. Každý projekt obsahuje tzv. projektový súbor (prípona .pro). Na importovanie projektu do prostredia *Qt Creator* stačí otvoriť projektový súbor a zvoliť verziu knižnice *Qt* (príp. kompilátora), ktorá sa má používať. Po úspešnom importovaní projektu je možné ho skompilovať.

B.2 Spustenie aplikácie

Na spustenie priloženej aplikácie je potrebné používať operačný systém MS Windows a 64-bitovú architektúru. Zároveň je potrebné mať nainštalovaný balík *Microsoft Visual C++ Redistributable for Visual Studio 2015* a ovládač pre mikročip *FTDI*. Najľahší spôsob ako získať ovládač pre prevodník *FTDI*, je pripojiť zariadenie s týmto mikročipom do USB zbernice počítača. Operačný systém si potom sám nainštaluje potrebný ovládač.

Po splnení všetkých požiadaviek je možné spustiť aplikáciu *SpectraController.exe*. Ako už bolo spomenuté v oddieli 4.3.2 priložená verzia aplikácie si po zvolení typu spektrometra sama spustí jeho ovládač. Na správne fungovanie tejto verzie aplikácie nie je potrebné mať pripojený žiadny spektrometer.