ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Řídicí aplikace pro reflektometr

Bakalářská práce

David Hodač

Praha, květen 2014

Vedoucí bakalářské práce: Ing. Rudolf Bayer

Studijní program: Aplikovaná elektrotechnika

Katedra elektroenergetiky

Poděkování

Velké poděkování patří Ing. Rudolfovi Bayerovi za ochotu, trpělivost a za podnětné připomínky k mojí práci. Děkuji mu i za čas, který mi věnoval při objasňování problematiky ovládání přístroje a programovacího jazyka Java.

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů [autorský zákon].

V Praze 23. května 2014 ……..

David Hodač

Anotace

Cílem této bakalářské práce je naprogramovat řídicí aplikaci, díky níž bude umožněno ovládat měřicí přístroj reflektometr. Projekt bude vytvořen ve vývojovém prostředí NetBeans IDE pomocí programovacího jazyka Java. Jedná se o jednoduchou aplikaci, kterou je možno nainstalovat na běžný počítač s operačním systémem Windows nebo Linux. Komunikace s reflektometrem bude probíhat přes USB rozhraní.

Abstract

The goal of this bachelor thesis is to create a control application. This application will control a measuring instrument which is called Reflectometer. The Application will be programmed in the development environment NetBeans IDE using the Java programming language. The program can be installed on the computer with Windows or Linux operating system and the communication with Reflectometer will be conducted by USB interface.

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra elektroenergetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **David Hodač**

Studijní program: Elektrotechnika, energetika a management
Obor: Aplikovaná elektrotechnika

Název tématu: **Řídicí aplikace pro reflektometr**

Pokyny pro vypracování:

1. Seznámení se s komunikačním rozhraním přístroje.
2. Návrh a vytvoření aplikace pro komunikaci s přístrojem a zpracování naměřených dat.
3. Provedení zkušebního měření a zhodnocení výsledků.

Seznam odborné literatury:

[1] HABEL, Jiří. Světlo a osvětlování, Praha: FCC Public, 2013, 622 s. ISBN 978-80-86534-21-3.
[2] HEROUT, Pavel. Učebnice jazyka Java 5., rozš. vyd. České Budějovice: Kopp, 2010, 386 s. ISBN 978-80-7232-398-2.
[3] HEROUT, Pavel. Java grafické uživatelské prostředí a čeština 5., rozš. vyd. České Budějovice: KOPP, c2001, 316 s. ISBN 80-723-2150-1.

Vedoucí: Ing. Rudolf Bayer

Platnost zadání: do konce letního semestru 2014/2015

prof. Ing. Josef Tlustý CSc.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 12. 2. 2014

Obsah

1 Úvod

V dnešním moderním světě je naprosto zbytečné provádět jakoukoliv činnost, která by měla zabrat dlouhou dobu, bez sofistikovaného automatizovaného systému. Nynější technická společnost je vybavena nevyčíslitelnou sumou možností, jak si ulehčit svoji práci a hlavně si ušetřit drahocenný čas. Proč se tedy nepustit do úkolu, který bude mít za výsledek nejen úsporu vzácného času, ale také nahlédnutí do počítačové praxe, dnes tolik využívaného programování.

Cílem této bakalářské práce je vytvořit program pro ovládání měřicího přístroje reflektometru. Aplikace je naprogramována ve vývojovém prostředí NetBeans IDE, které umožňuje vytvářet aplikace v programovacím jazyce Java. Zhotovená aplikace bude sloužit katedře elektroenergetiky pro laboratorní měření.

V první části práce bude zmíněna podstatná teorie o základních pojmech a veličinách světelné techniky. Objasněny budou potřebné definice, které se tohoto projektu týkají. Dále pak bude popsáno, jaký programovací jazyk byl použit a důvod k jeho aplikování. Probrán bude nejen zmiňovaný programovací jazyk, nýbrž také vývojové prostředí, na základě kterého bude řídicí aplikaci vytvořena. V poslední řadě bude objasněna funkčnost programu a komunikace s reflektometrem.

1.1 Koncepce řídicí aplikace pro reflektometr

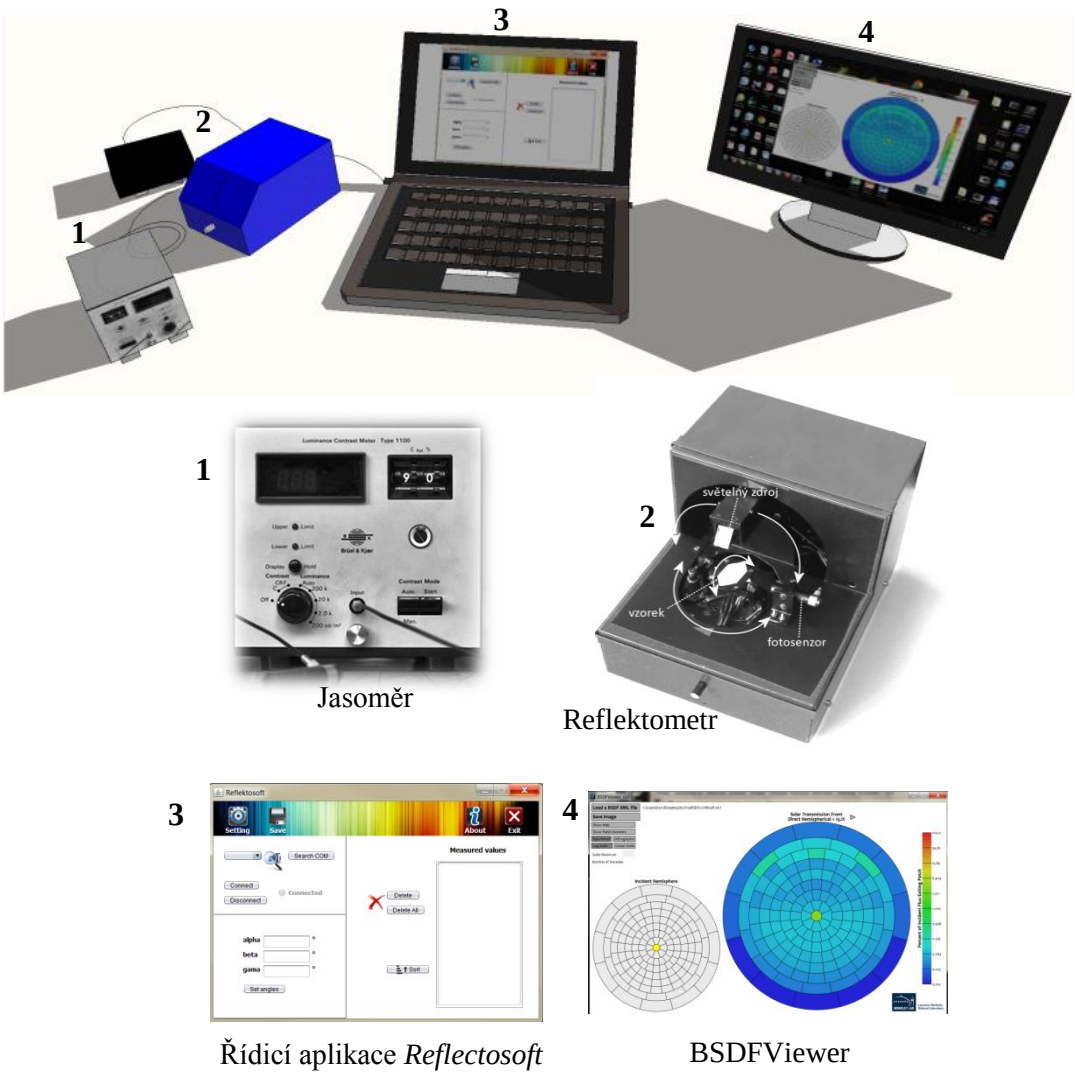
Řídicí aplikace je koncipována na základě jednoduchého programu, ve kterém bude možno ovládat měřicí přístroj pomocí rozhraní. Uživatel si v určených mezích bude moci nastavit úhly, pod nimiž chce měřit odrazy světla. Komunikace programu s měřicím přístrojem je naplánována přes USB kabel. Pro toto spojení bude v hlavním okně programu tlačítko k možnosti vyhledání portu. Nebude chybět ani tlačítko pro připojení k nalezenému portu.

V aplikaci si uživatel bude moci naměřená data seřadit podle velikosti od nejnižší hodnoty po nejvyšší. Dále bude zprostředkováno uložení naměřených hodnot, jak ve formátu csv[1], tak i v xml[2] pro další zpracování dat.

V ideálním případě bude pracoviště rozmístěné podle obrázku 1. Pod očíslovanými prvky se skrývají části celého pracoviště. Čísla jedna a dva zobrazují jasoměr s reflektometrem. V reflektometru je nasazena sonda jasoměru k měření odrazů světla. Pod čísly 3 a 4 je umístěn počítač s již funkční řídicí aplikací a obrazovka se s

[1] Formát CSV znamená Comma- Separated Value, kde hodnoty jsou ukládány do textového souboru oddělené čárkou.
[2] Formát XML je Extensible Markup Language, co v překladu znamená rozšiřitelný značkový jazyk.

puštěným programem nazývaným *BSDFViewer*. Tento nástroj slouží jako prohlížeč, jenž poskytuje interaktivní průzkum BSDF[3] datových souborů s koncovkou *.xml* [4].



Obrázek 1: Koncepce rozmístění pracoviště k ovládání reflektometru a zpracování naměřených dat, obrázky jasoměru a reflektometru jsou převzaty z [10]

1.1.1 Měřicí přístroj reflektometr

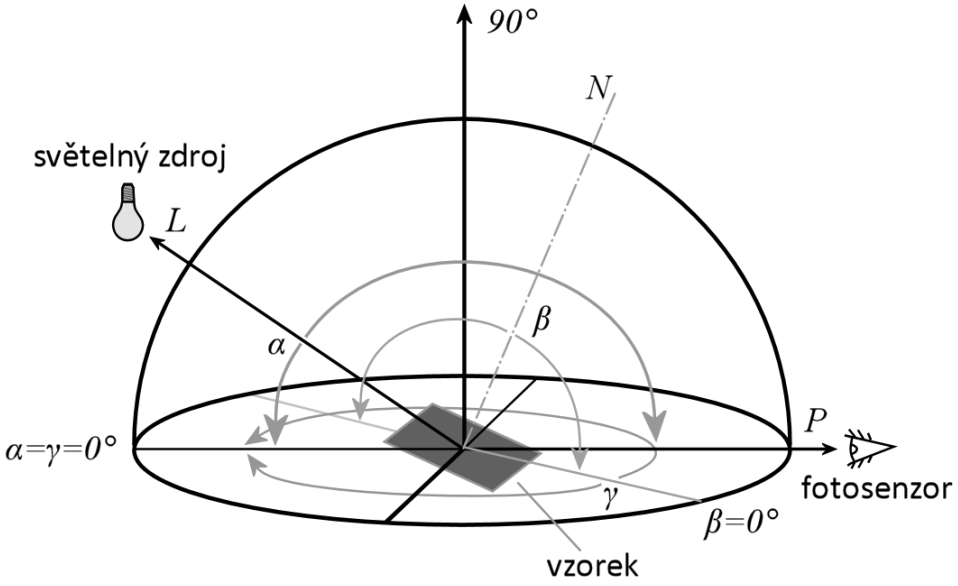
Měřicí přístroj reflektometr bude sloužit k měření prostorového rozložení jasu odrazných povrchů různých materiálů. Obsahuje tři motory a každý z těchto pohonů bude mít vlastní ovládací mechanismus. V přístroji, do pěvně dané polohy, je umístěn

[3] BSDF (Bidirectional scattering distribution function) dvousměrná funkce distribuce rozptylu.

fotosenzor jasoměru. Na obrázku 2 je názorně ukázán pohyb všech pohonů měřicího přístroje, umístění světelného zdroje a situování fotosenzoru.

Pod úhlem alfa je zobrazen pohyb prvního pohonu. Jde o systém, který natáčí světelný zdroj. Úhly beta a gama ukazují směr natáčení měřeného vzorku druhým a třetím pohonem ve dvou rovinách. Měřený vzorek je povrch určitého tělesa pro zjištění odrazných vlastností materiálu.

Po celé vnitřní části reflektometru je matný povrch černé barvy k vyvarování nechtěného odrazu. Přístroj pro měření je zcela uzavíratelný, tak aby výsledky měření nebyly ovlivňovány vnějším vlivem světla.



Obrázek 2: Ukázka pohybu světelného zdroje a vzorku v přístroji, převzato z [10]

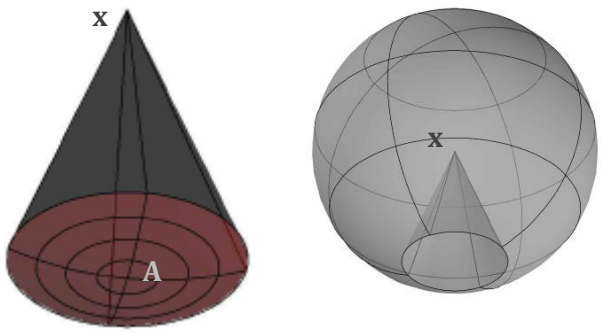
2 Základní pojmy a veličiny světelné techniky

Lidským zrakem není možno vnímat souhrnné působení záření za určitý čas, a tedy veličiny světelné techniky se nevyhodnocují pomocí radiometrických veličin (například zářivý tok nebo zářivost apod.), ale fotometrickými veličinami, jež zohledňují subjektivní citlivost oka, která není stálá k záření různých vlnových délek [1].

Pro určování těchto základních pojmů a veličin je nutné objasnit, o jaký se jedná zdroj. Pokud se jedná o bodový zdroj světla nebo o prostorový světelný zdroj. Bodový zdroj světla je definován jako světelný zdroj, jehož vzdálenost od pozorovacího místa je několikanásobně větší než rozměr světelného zdroje, proto jeho velikost od osvětlovací plochy je prakticky bezvýznamná. Z tohoto důvodu je označován jako bodový. Přejdeme-li ke složitějšímu prostorovému zdroji světla, tak ten je na opačnou stranu brán, jako těleso s určitým prostorovým tvarem. Rozměr tohoto zdroje nelze zanedbat, respektive je nutné počítat nejen s plochou zdroje, ale také s úhlem, pod kterým se nachází pozorovací místo [1].

2.1. Prostorový úhel

Mezi podstatnou světelně technickou geometrickou veličinu se řadí prostorový úhel. Jedná se o velikost výseče vyťaté kuželovou plochou z povrchu jednotkové koule. Označení tohoto úhlu je Ω a jednotkou je *steradián (sr)*. Prostorový úhel se určuje poměrem plochy vytyčenou na povrchu koule ke kvadrátu poloměru koule. Pro názornou ukázku slouží obrázek 3, kde je zobrazena zvětšená kuželová plocha vyjmutá z prostoru koule o poloměru *r*. Ve vrcholu zobrazené kuželové plochy je vyznačen bod *x*, který vychází ze středu koule. [1].



Obrázek 3: Ukázka prostorového úhlu

2.2 Světelný tok

Jedná se o množství světelné energie, která odpovídá zářivému toku vyhodnoceným zrakem v souladu se spektrální citlivostí oka. Značí se Φ a jednotka je 1 *lumen (lm)* [1].

2.3 Svítivost

Svítivost patří mezi základní jednotky soustavy SI. Svítivost se značí I a jednotkou je *kandela (cd)*, znamenající v překladu svíčka. Tím je míněno, že jedna kandela je přibližné množství svítivosti jako při jedné zapálené svíčce [1].

2.4 Osvětlenost

Značkou je E a jednotka se nazývá *lux (lx)*. Osvětlenost je definovaná jako světelný tok Φ dopadající na určitou plochu A [1].

Osvětlení je velice sledovaná veličina pro zajištění co nejvyšší bezpečnosti při zrakovému úkonu.

K měření osvětlenosti se obecně používá přístroj zvaný luxmetr. Ten zpracovává signál získaný prostřednictvím čidla, které vyhodnocuje dopadající světlo [1]. Měření osvětlenosti bývá nejčastěji prováděné měření, protože je například nutné udržovat v určitých prostorách rovnoměrnou hladinu osvětlenosti podle blíže specifických požadavků.

2.5 Jas

Jas je pro tuto práci nejdůležitější světelně technická veličina, protože díky jasu je možné měřit hodnoty odrážející se od povrchu rozličných objektů. Jas má jednotku $cd.m^{-2}$ a značí se L. Jde o veličinu, která je definována jako prostorová a plošná hustota světelného toku přenášena svazky paprsků [1].

V přístroji, který je za úkol zprovoznit, se nachází měřicí přístroj zvaný jasoměr. K měření jasů se mezi hlavní přístroje používají vizuální jasoměry. Vizuální jasoměr funguje na základě porovnávání dvou hodnot jasu v zorném poli[4]. Jedna hodnota jasu je neznámá a ta je porovnávána s uměle vytvořeným jasem, který je produkován přímo v měřicím přístroji. Přístroj se skládá z fotočlánku, vyhodnocovacího systému a z tubusu, který má uvnitř černý povrch a clony sloužící k vymezení dopadajících paprsků směrem k fotočlánku [1].

Podle velikosti clony, určující zorné pole, jsou jasoměry dále rozlišovány na bodové a integrační. [1].

[4] Zorné pole je takový prostor, který je měřicí přístroj či oko schopné zachytit.

3 Odraz světla

Přibližně z 2. století před naším letopočtem existuje příběh o známém matematikovi a fyzikovi Archimedovi, který při obléhání jeho města Syrakus na Sicílii, použil odraz slunečních záření od měděných štítů, jež měli bránící vojáci proti římským útokům. Vojáci nastavili štíty do takového úhlu, aby odražené paprsky dopadaly do jednoho bodu. Tímto způsobem byly plachty atakujících římských lodí zapalovány na dálku. Tento nápad sice nepomohl k zastavení napadení města Syrakus, ale byla to jistě, na tu dobu, obdivuhodná myšlenka [9].

V reálném světě odrazy světla, ať už je to od povrchu zemského nebo různých objektů, nemají bezvýznamný vliv na naše zrakové vnímání okolí. Objekty, a především jejich povrchy, kterými jsme obklopeni, nejsou primárními zdroji světla. Jde o sekundární zdroj světla, tedy o odražené světlo. Za zmínku stojí odraz světla od Měsíce v nočních hodinách při jasné obloze. Například při úplňku je poměrně dobrá viditelnost v porovnání s novoluní[5] fází Měsíce nebo v zimním období, kdy se od zasněžené krajiny odráží slunečné záření. Jedná se především o vyšší nadmořské oblasti a bývá to velmi nepříjemné pro lyžaře či turisty.

Odrazy světla jsou hojně využívané jako bezpečnostní prvky. Především pro silniční dopravu, kde si pracovníci v silničním stavebnictví, cyklisté, policisté, ale i účastníci nehod oblékají reflexní oděvy pro vyšší viditelnost. Nejedná se pouze o silniční dopravu nebo stavby, nýbrž i v některých sportech se používají reflexní oblečení.

Na rozdíl od záměrných světelných odrazů jsou také četně používané antireflexní úpravy materiálů, především čirá dioptrická skla. Tyto skla jsou k nalezení v dražších modernějších a kvalitnějších brýlích, které majiteli umožňují příjemnější pohled v slunných dnech.

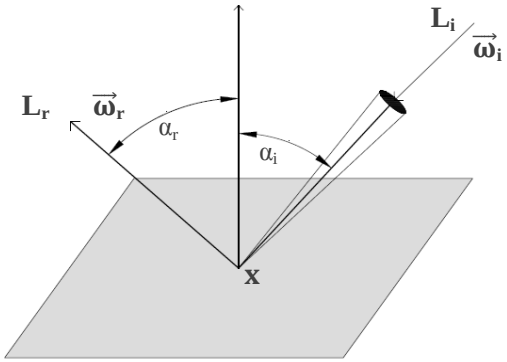
Povrchy různých těles jsou rozlišovány podle odlišných odrazných vlastností. Ideální případy odrazů jsou difúzní a zrcadlové. Ve skutečném světě se jedná o odraz něco mezi difúzním odrazem a zrcadlovým. V každém případě je nutné podotknout, že část světelné energie je vždy pohlcena materiálem a přeměněna na teplo.

3.1 Dvousměrová odrazová distribuční funkce – BRDF

Je jasné, že světlo, které vnímáme je z největší části světlo odražené od různých objektů a také s tím poměrně souvisí i fakt o barvě předmětů.

[5] Novoluní fáze Měsíce je stav, kdy Měsíc je k zemi natočen svojí neosvětlenou částí.

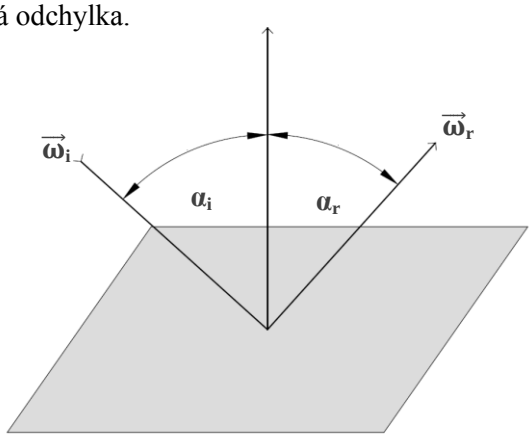
BRDF (Bidirectial Reflectance Distribution Function) znamená v překladu dvousměrová odrazová distribuční funkce a je používaná obzvláště v moderní počítačové grafice. BRDF popisuje odrazové schopnosti povrchu objektu v jednom bodě. Podle obrázku 4 je tímto bodem *x*. Světlo dopadá ze směru $\vec{\omega}_i$ a je odraženo ve směru $\vec{\omega}_r$. Funkce BRDF je definována podílem odraženého světla v bodě *x* a dopadajícího světla, neboli poměr odraženého jasu a vstupní diferenciální jas promítnutý na kolmou plochu [2].



Obrázek 4: BRDF, dvousměrová odrazová distribuční funkce

3.2 Zrcadlový odraz

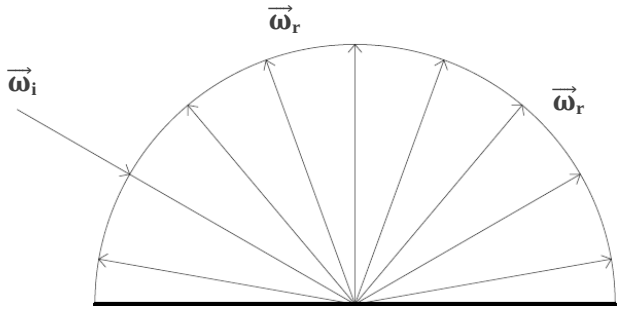
Nejzákladnější typ odrazu dopadajícího světelného paprsku je zrcadlový odraz. Zrcadlový odraz má poměrně předvídatelný způsob chování. Na obrázku 5 je vidět intuitivní průběh světelného paprsku, dopadajícího a odraženého od ideálně lesklého a hladkého povrchu, kde přesně to co dopadne, to se i odrazí. To znamená, že úhel dopadu se rovná úhlu odrazu. Tato definice platí, pokud se jedná o ideálně lesklý a hladký povrch, ovšem v praxi je taková věc nemožná, i když u některých materiálů mezi úhlem dopadu a odrazu není příliš velká odchylka.



Obrázek 5: Ideální zrcadlový odraz, kde úhel α_i se rovná úhlu α_r

3.3 Difúzní odraz

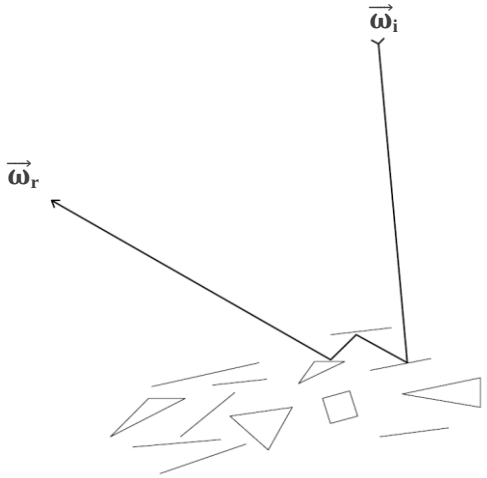
Difúzní odraz rozptyluje vstupní světelné paprsky rovnoměrně do všech směrů v prostoru, respektive jas se rozloží do prostoru všude stejně. Jedná se opět o ideální stav materiálu. Tento případ je na obrázku 6, kde je patrné, že odraz světla je nezávislý na úhlu dopadajícího světelného záření. Mezi materiály, jež mají nejlepší difúzní vlastnosti, patří zejména křída [2].



Obrázek 6: Ideální difúzní odraz světla

3.4 Lesklý odraz

Lesklý odraz je zapříčiněn na základě povrchu složeného z mikroplošek. Tyto mikroplošky si vzájemně stíní a umožňují světlu se dostat pod povrch materiálu. Na směr odrazu má zásadní vliv podpovrchové stínění a rozptyl světla. Tento odraz není zcela určitelný, proto je nutná aproximace při jeho analyzování. Na obrázku 7 je přiblížení situace při lesklém odrazu [2].



Obrázek 7: Lesklý odraz od povrchu složeného z mikroplošek

3.5 Činitel odrazu některých látek

Světelný tok, který dopadá na povrch materiálu je rozdělen do tří charakteristických částí. První část je integrální činitel odrazu, druhou je integrální činitel prostupu a třetí integrální činitel pohlcení. Nejvyšší prioritu v této práci má integrální činitel odrazu, proto v následující tabulce 1 jsou především vyznačeny povrchy materiálů s přibližnou hodnotou činitele odrazu. Každé těleso má jiný integrální činitel odrazu, který je určen podílem odrážejícího světelného toku a celkového dopadajícího světelného toku. Hodnoty se udávají v procentech. V tabulce jsou zobrazeny materiály seřazené sestupně na základě odrazivosti jejich povrchu [1].

Povrch materiálu	**Činitel odrazu v %**	**Činitel prostupu v %**	**Činitel pohlcení v %**
Leštěné stříbro	85 – 94		
Bílý smalt	85 – 90		
Bílý papír	přibližně 80		
Bílá sádra	přibližně 80		
Bílá malba	76 – 88		
Leštěné zlato	70		
Leštěná platina	62		
Javorové dřevo	40 – 50		
Dubové dřevo	30 – 49		
Bílé hedvábí	28 – 38	61 - 71	přibližně 1
Červené cihly	25		
Ořechové dřevo	15 – 20		
Tmavě zelená malba	15 – 35		
Čiré sklo (2 - 4 mm)	6 – 8	90 - 92	2 - 4
Černá malba	2 – 4		

Tabulka 1, ukazující odrazivost popřípadě prostup a pohlcení vybraných povrchů. Data jsou převzata z [1]

4 Programování

Počítačová technika ovládá dnešní světovou ekonomiku a díky ní je možné provádět operace, které by bez výpočetní techniky bylo absolutně nemožné uskutečnit. Software je nedílnou součástí k ovládání hardwaru.

Podíváme-li se do současného moderního světa techniky a především do chytrých mobilních přístrojů, kde si vlastníci mohou stahovat různé aplikace, je třeba si uvědomit, že každé toto programové vybavení muselo být vytvořeno programátorem v konkrétním programovacím jazyce. Vyskytuje se neuvěřitelné množství programů sloužící k zábavě, jako pomocník při práci nebo k ovládání elektroniky.

Existuje několik programovacích jazyků, ale já zmíním jen Javu, kterou jsem použil.

4.1 Java

Zaprvé bych chtěl v několika větách vysvětlit, proč jsem se rozhodl tento úkol vytvářet v programovacím jazyce Java a ne v jiném jazyce, například v C.

Začnu-li od nejjednoduššího, tak zkrátka v jazyce Java jsem měl již zkušenosti oproti jazyku C. Za další Java je vyšší programovací jazyk, ve kterém je možno využít objektové programování, dovoluje vytvářet programy, aniž by bylo nutné je kompilovat na ostatní počítače. Java má svojí specifickou a promyšlenou kompilaci, při níž se zdrojový kód převede do bajtkódu[6] a je uložen do souboru s koncovkou *.class* [6]. A závěr mé odpovědi je, že práce v Javě, oproti zmiňovanému jazyku C je přehledná, „čistá“, jednodušší, s vysokou úrovní zabezpečení, což zajišťuje ochranu po spuštění programu. Java se svým zabezpečením významným způsobem předčila ostatní „konkurenty“ vyšších programovacích jazyků. Patří mezi opravdovou špičku v programovacích jazycích a je nedílnou součástí dnešního softwarového světa.

V současné době je Java také řazena mezi nejoblíbenější programovací jazyky, jak mezi začínajícími programátory, tak i mezi profesionály.

Java je:

- **objektově orientovaná:** jazyk, který je snadno rozšiřitelný, právě proto, že je založen na objektivně orientovaném modelu.

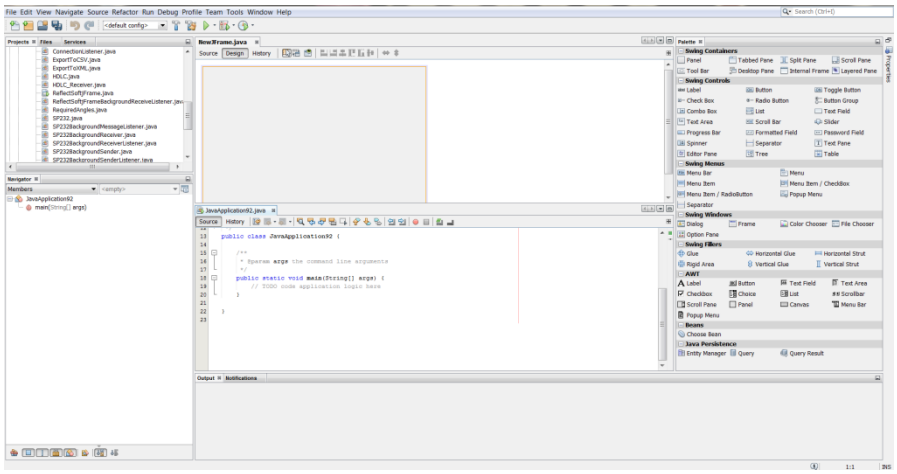
[6] Bajtkód je označení pro snadnou přenositelnost aplikace vytvořené v programovacím jazyce Java

- **jednoduchá:** tento jazyk je navržen pro jednoduché a rychlé pochopení. Už jenom to, že metody se nazývají celými slovy, třeba metoda *isGreater* nebo *toString* a podobně.
- **bezpečná:** oproti jazyku C, ve kterém je možno si „sáhnout“ na téměř vše, co se nachází v paměti systému, tak naopak v Javě je na vše odkazováno přes určitou adresu, nazvanou referencí, respektive referenční proměnou.
- **více vláknová:** běh programu může být zajištěn nejen hlavním vláknem, ale chod aplikace existuje i s více vlákny najednou. Tato užitečná vlastnost dovoluje činnost vláken na pozadí, kde se běh programu významným způsobem urychluje.
- **multiplatformní:** v praxi to znamená, že napsané programy jsou zcela přenositelné mezi operačními systémy bez kompilace. Samozřejmě musí na to být takový operační systém, umožňující spuštění Java platformy [7].

4.2 Vývojové prostředí Netbeans IDE

Pro programování rozsáhlejších aplikací je dobré používat sofistikovanější prostředky, které rapidně ušetří drahocenný čas a občas i nervy. Ovšem jako je to se všemi věcmi, jež mají ušetřit a zefektivnit práci, tak nejprve je potřeba naučit se s nimi pracovat, jinak hrozí, že je naopak práce ztížena. Vždy je nepříjemné se seznámit s novým softwarem, a zpravidla to vývojářovi nějakou dobu trvá, než se s novým vývojovým prostředím sžije.

Jak již bylo poznamenáno v úvodu této práce, tak vývojové prostředí Netbeans stojí na základě programovacího jazyka Java. Je to nástroj k usnadnění vývoje Java aplikací, obsahující všechny nezbytné části k vytvoření programu, jako je textový editor zdrojového kódu, kompilátor, debbuger apod. Velkou výhodou tohoto vývojového prostředí je grafické uživatelské rozhraní, pro grafické zpracování aplikace.



Obrázek 8: Ukázka vývojového prostředí Netbeans IDE

4.2.1 Grafické uživatelské rozhraní

Java nabízí dvě grafická uživatelská prostředí. Jedním je starší AWT[7] a novější JFC Swing[8]. V obou případech se jedná o grafické zázemí, zatímco AWT patří mezi zastaralé, tak JFC Swing je v dnešní době používanější. Je to zapříčiněno hlavně v nabízených možnostech, kdy JFC Swing má větší množství, ovšem je nutné dodat, že JFC Swing vychází z AWT. V mém případě jsem zvolil JFC Swing [3], [6].

Pomocí grafického uživatelského rozhraní, jež je obsaženo ve vývojovém prostředí Netbeans, může uživatel interagovat s ovládacími prvky a tak vytvořit ovladatelnou aplikaci pomocí klávesnice a myši. Pod výběrem různých oken, panelů, ovladačů či ukazatelů si vývojář může sestavit program, jaký chce. Představa v nastavování tlačítek, oken, různých panelů, popisků nebo přidávání obrázků prostřednictvím uživatelem psaných kódů manuálně je pro mě dosti trpká. Netbeans nabízí palety, ve kterých si vývojář vybere komponentu a přemístí přetažením myši na zvolené místo. Této metodě se říká drag and drop (táhnout a pustit). Každá komponenta má definovatelné vlastnosti pro jednoduché přizpůsobení.

Někomu se může jevit uživatelské prostředí, že se už nejedná o programování, nýbrž o jakési kreslení nebo skládání. Měl by svým způsobem pravdu, ale podle mého názoru je to užitečný nástroj, který programátorovi ušetří opravdu spoustu stráveného času u počítače.

4.2.2 Výhody a nevýhody vývojové prostředí Netbeans IDE

Mezi hlavní výhody patří fakt, že vývojové prostředí Netbeans IDE je zcela zdarma a volně dostupné na webu od vývojářské společnosti Oracle Corporation. Následujícím obrovským přínosem je multiplatformní užití pro různé operační systémy. To znamená, že tento software je uživatel schopný zprovoznit, jak na nejpoužívanějším operačním systému Windows, tak i na systému Linux nebo Mac OS. V neposlední řadě musím podotknout, že aplikace kontroluje průběh vývoje aplikace, upozorňuje na vzniklé chyby, napsané nepozorným programátorem. Jednoduše je toto prostředí použitelné i pro začínající programátory.

Co se týká nevýhod, tak já nejsem dost kritický v této věci a za sebe sem musím napsat pouze o uživatelském rozhraní, ve kterém by vzhled panelů a tlačítek mohl být více vzhledově propracovanější, zkrátka modernější.

Předpokládám, že zmíněných kladů či záporů by mohla být celá řada, ovšem chtěl jsem poznamenat jen ty významné. Navíc zmiňované vývojové prostředí nabízí opravdu

[7] AWT (Abstract Window Toolkit)
[8] JFC Swing (Java Foundation Classes)

profesionální zázemí pro zkušené uživatele. Já zdaleka neprozkoumal většinu funkcí, kterých je nabízeno pro práci s tímto prostředím.

5 Seznámení s postupem vytvoření aplikace

Aplikace je vytvořena ve vývojovém prostředí Netbeans IDE, které je mimo jiné určené především pro vývoj aplikací v jazyce Java, jak již bylo zmíněno. Program je naprogramován objektově, pomocí tříd.

Je známo, že objekt je instancí třídy. Kdybych měl přiblížit, co je skryto pod pojetím třída a objekt, zabrousím-li do stavebnictví, kde při stavbě rodinného domu nejdříve potřebuji mít nakreslený plán celkové stavby. Tento plán, respektive projekt domu odpovídá struktuře, jak se má budova postavit a tou je v terminologické oblasti programování – třída. V opačném případě, objekt je již postavené stavení, které bylo vybudováno dle navrženého projektu, respektive je to něco, co se reálně použije na rozdíl od plánu jakožto třídy. To znamená, že skutečný, už postavený dům je instancí naplánovaného projektu rodinného domu.

V průběhu spuštěného programu se objekty jak dynamicky tvoří tak zanikají. Je možné založit libovolné množství objektů pomocí speciální metody nazývané konstruktor. Konstruktor se musí deklarovat a inicializovat[6].

K objektovému programování se ještě vztahují tzv. *gettery* a *settery*. Pro vysvětlení je nutné ještě dodat vysvětlení k označení *private* a *public*. Už intuitivně je jistě vnímáno, že slovo *private* bude označení pro takovou metodu nebo proměnou, pro niž nebude umožněn přístup přímo. Oproti stanovení *public* je tomu přesně naopak. Metoda či třídní proměnná je zcela zpřístupněna z vnějšku třídy.

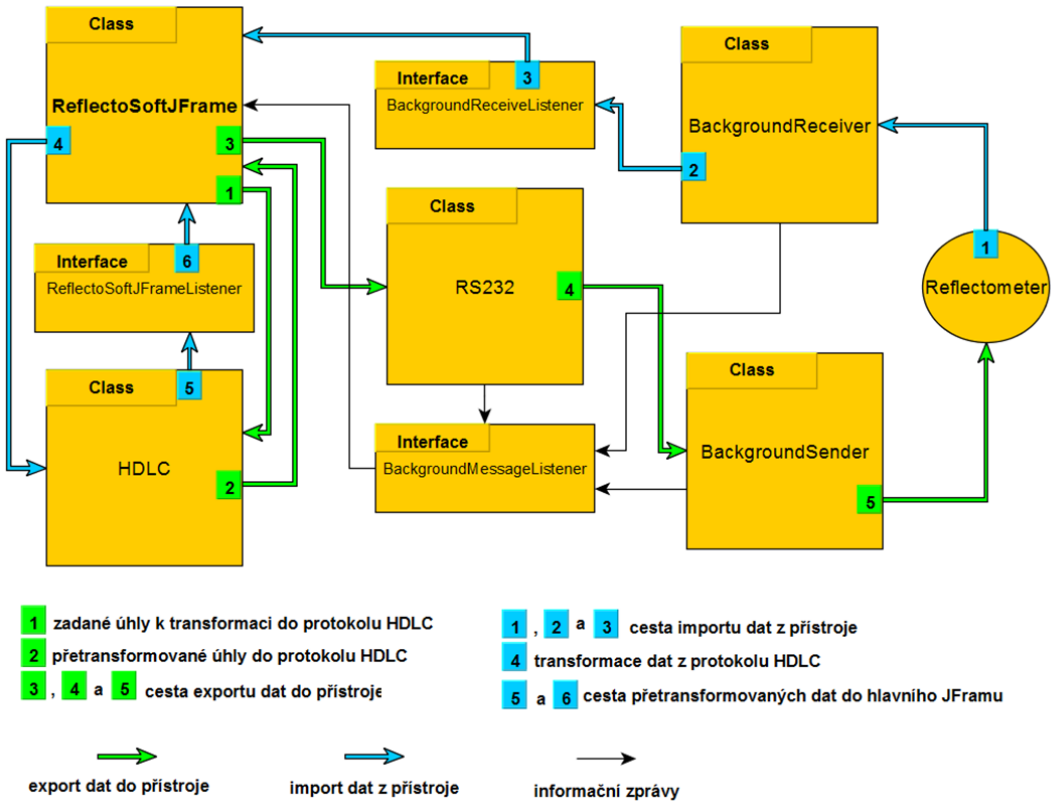
Nyní mohu přistoupit k samotným *getterům* a *setterům*. *Settery* jsou metody, u nichž při vyvolání nastavíme nějakou hodnotu do proměnné s označením před datovým typem *private*. Na druhou stranu *gettery* jsou metody, které vracejí uloženou hodnotu z proměnné.

V programu jsou také použity statické metody, značící se *static*. Takovéto metody jsou v třídách, u nichž není potřeba založení objektu.

Dále je také aplikována možnost *Interface*. V překladu rozhraní. Rozhraní je využíváno především pro implementování více rozhraní v jedné třídě. Je známo, že v Javě není možné dědit více, jak z jedné třídy a z tohoto důvodu Java nabízí tuto schopnost. V mém případě využívám této možnosti pro přenos dat mezi objekty.

Některé třídy, které jsou použity, dědí od dalších tříd nabízející Java API[9].

5.1 Zjednodušený popis ovládací aplikace



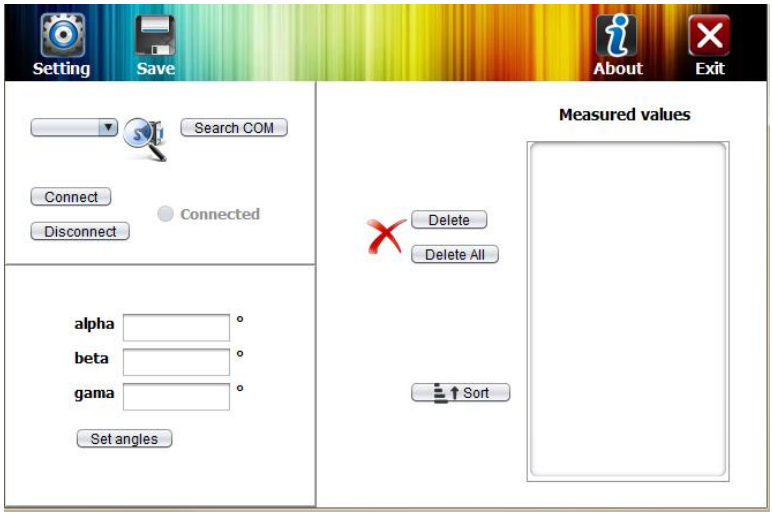
Obrázek 9: Zjednodušené schéma řídící aplikace Reflectosoft

Hlavní třídou je *ReflectoSoftJFrame*, kde je při zapnutí aplikace spuštěno hlavní vlákno programu. V základní třídě je prakticky uložen veškerý obsah nastavených ovládacích komponent pro interakci s přístrojem. Pro ukázku řízení přístroje slouží obrázek 9, kde je simplifikovaný nástin závislosti jednotlivých tříd aplikace. Oranžové čtverce s označením *Class* jsou zástupci tříd. Obdélníky ve stejné barvě zobrazují komunikační rozhraní, obsahující jednoduché kontraktní metody a elipsa je znázorněním měřicího přístroje reflektometru. Ve zjednodušeném schématu jsou různobarevně znázorněny trasy odchozích, příchozích dat a informačních hlášení. Pro export dat z aplikace do hardwaru přístroje náleží zeleně vyznačené šipky. Pro import dat z přístroje do programu jsou směrovky vyznačené modrou barvou a černé šipky zakreslují cestu informačních zpráv.

[9] API (Application Programming Interface), znamenající v překladu rozhraní pro programování aplikací a jde o sbírku balíků a tříd v Javě [8].

Uživatelem zadané úhly jsou zapouzdřeny do protokolu HDLC. Vytvoří se zabalený řetězec, který je poslán do měřicího přístroje. V měřicím přístroji je řetězec zpracován a řídicí elektrotechnika nastaví požadované úhly. Aplikace by měla dostat potvrzovací hlášení o zpracování dat. Přístroj vrátí hodnoty, podle kterých nastavil polohy motorů s naměřenými výsledky, které jsou rozbaleny z protokolu HDLC a jsou zobrazené v panelu naměřených hodnot k dalšímu využití.

5.2 Hlavní okno programu



Obrázek 10: Okno řídící aplikace pro reflektometr

Při spuštění aplikace jsou v hlavním okně vyobrazena veškerá tlačítka *JButton* panely *JPanel*, příkazové řádky *JTextField* a list pro zobrazení dat *JList*. Celkový vzhled se nachází na obrázku 10.

Pro komunikaci s přístrojem si uživatel musí nechat vyhledat COM port zapojený ke komunikaci počítače s měřicím přístrojem. K tomuto vyhledání slouží tlačítko *Search COM*. Po vyhledávání se v nabídce objeví několik možných portů k výběru. Ona nabídka je v Javě třídou, s názvem *JComboBox*, do které se ve formě pole uchovají názvy nalezených portů. Uživatel zvolí ten, který je připojený s přístrojem a může přistoupit ke spojení tlačítkem *Connect*. Ošetřen je rovněž případ, pokud již existuje spojení s daným portem. Při této události se objeví upozorňující zpráva uživateli. Pro deaktivace spojení slouží tlačítko *Disconnect* a komunikace se automaticky přeruší. Pro informovanost uživatele, pokud je spojení s portem navázáno, slouží *JRadioButton*. V této situaci jde o přepínací tlačítko, zobrazující stav připojení. V případě spojení se oblast spínače vyplní tečkou a naopak, je-li port odpojen, zůstane zóna tlačítka prázdná.

Nejzákladnější možnost programu je zcela jistě nastavování úhlů, kde si uživatel programu nastaví hodnoty do příkazového řádku, pod kterými se má provést měření. Při vyplňování úhlů do textového pole se automaticky kontroluje, zdali uživatel omylem nenapsal nějaké písmeno místo číslice, případně desetinnou čárku mimo desetinné tečky. To vše je hlídáno událostí *FocusLost*. Při ztrátě kurzoru v poli se vyvolá metoda, která kontroluje číselnou hodnotu, jestli se pohybuje v definovaném rozmezí. Nebylo by žádoucí, aby uživatel mohl nastavit jakýkoliv úhel. Pro akci odeslání dat do přístroje slouží tlačítko *Set angles*.

V hlavním okně programu se také nachází další tlačítka jako *Delete* nebo *Sort*. Dále pak je pro zobrazování dat panel *JList*, do kterého jsou přijímána z měřicího zařízení data, řazena pod sebe. Tlačítko *Delete* je naprogramováno pro několik způsobů výběru vymazání. V prvním případě jde o výběr vymazání pouze jednoho řádku dat. Ve druhém případě už je na pozadí metodou *getSelectedIndices()* sledováno, kolik je uživatelem označeno řádků k vymazání. Podle tohoto počtu je vytvořeno pole, kam se přesunou indexy označených řádků a rychlým *for* cyklem jsou data na označených řádkách odstraněny. V aplikaci je vymazání dat uskutečněno postupně, ale jedná se o tak velikou rychlost, že to uživateli přijde jako by se to provedlo najednou. V posledním případě jde o veškeré odstranění dat z celého panelu, k tomu slouží tlačítko *Delete All* a k tomu není co dodat. Na řadu přichází tlačítko *sort*, které má za úkol data v panelu přerovnat podle velikosti od nejnižší hodnoty po nejvyšší.

5.2.1 Bubble sort

Pro řazení dat je zvolen Bubble sort, do češtiny přeložené jako probublávání. Jedná se o nejjednodušší řadicí algoritmus vůbec a to byl taky nejvýznamnější důvod, proč je použit. Bubble sort v porovnání s dalšími řadicími algoritmy patří mezi pomalé, ale musím podotknout, že v případě této aplikace, kde se množství dat bude pohybovat maximálně v tisících, tak rychlost nebude nijak významná. Samozřejmě, kdyby bylo potřeba řadit opravdu velké množství dat v řádu už stovky tisíc, tak by bylo nutné zvolit mnohem rychlejší algoritmy, jako například Quick sort.

Nyní k samotnému „probublávacímu“ algoritmu. Bubble sort pracuje na základě testování jednotlivých prvků umístěných v řadě. Nejdříve porovnává člen na první pozici se členem druhé pozice, a jestliže je první člen větší než jeho soused, tak se prohodí. V opačném případě se nic neděje a číslice zůstanou na svých místech. Dále pak se srovnávají pozice druhá s třetí a opět se porovnají a případně prohodí své pozice. Program tímto způsobem dojede až na předposlední a poslední pozici. Takto popsaný algoritmus bude proveden *n-1* krát, kde *n* je počet prvků řady k setřídění.

5.3 Menu v hlavním okně aplikace

Vývojové prostředí Netbeans, respektive Java nabízí *JMenuBar* pro vytvoření výběru dalších možností. Toto poskytované menu je ale zcela stejná nabídka známá od zmiňovaného Microsoft Windows. Já jsem jednoduše zvolil čtyři tlačítka *JButton* a postupoval jsem identickou cestou, jako při nastavování tlačítek v hlavním okně aplikace.

Jde tedy o tlačítka ukládání dat *(Save)*, nastavení *(Setting)*, podání informace o mně (*About*), jakožto autoru aplikace a ukončující tlačítko *(Exit)*.

5.3.1 Ukládání dat

Mezi další možnosti aplikace je export dat do textového editoru ve formátu *.csv* nebo souboru ve formátu *.xml*.

Formát *.csv* byl zvolen z důvodu pro další práci s daty, především v Microsoft Excel je přímo funkce, která rozřadí hodnoty do sloupců podle čárky nacházející se mezi naměřenými hodnotami.

Export dat do formátu .xml ještě není dokončen, ale jde o ukládání naměřených dat pro zobrazování v programu s názvem *BSDFViewer*, který dokáže naměřené hodnoty zobrazovat v grafu.

5.3.2 Nastavení

Po stisknutí tlačítka *Setting* v menu aplikace se objeví další okno, ve kterém se budou nacházet další tlačítka pro ulehčení obsluhy aplikace. Jedná se především o ukládání a následné získaní názvů vyhledaných portů k připojení ze souboru. Tato eventualita by měla urychlit připojení k určitému volnému portu. Tato možnost se v nastavení nachází z důvodu ne příliš rychlého vyhledávání přístupných portů. Soubor, do něhož jsou nastavitelné názvy exportovány, má koncovku *.xml*.

Po načtení uložených názvů portů se vloží do nabídky *JComboBox*.

5.3.3 Informace o autorovi

Pod tlačítkem *About* se skrývá otevíratelné okno zobrazující informace o autorovi této práce. Jako u každé aplikace, tak i v této je podáno sdělení o tvůrci, zejména jméno a příjmení a popřípadě kontaktní email.

5.3.4 Ukončení aplikace

Před ukončením aplikace je uživateli otevřen dialog, ve kterém se aplikace ubezpečuje, jestli má být ukončena. Jde především o situaci, při níž by uživatel neúmyslně ukončoval aplikaci, a momentálně by probíhalo měření.

5.4 Komunikace vytvořené aplikace s měřicím přístrojem

Spojení řídicí aplikace s reflektometrem je postaveno na základě komunikačního rozhraní USB. Oficiální knihovny pro Javu k podpoře sériové komunikace nejsou podporovány, nicméně existují open-sourcové knihovny RXTX, jak pro 32bitovou platformu, tak i pro 64bitovou. Já jsem použil knihovnu pro 32bitovou platformu. V příloze je mimo jiné popsán postup instalace knihovny RXTX.

5.4.1 Komunikace USB pomocí protokolu HDLC[10]

Struktura protokolu HDLC je na obrázku 11. Jedná se o strukturovaný rámec, který je složen z bajtů *(byte)* neboli polí, které mají svojí funkci. Každý protokol obsahuje na začátku a na konci bajt s názvem *Flag*. Jde o vlajkový bajt, jenž je v binárním tvaru zapisován 0111 1110 nebo v hexadecimálním vyjádření 0x7E. Pokud jdou po sobě dva bajty *Flag*, tak přichází prázdný rámec, se kterým se dále nic neprovádí. Druhý bajt v pořadí je adresa *(Adress)*. Tento bajt v sobě zahrnuje adresu směru přenosu. Dalším polem je *Control*, který určuje typ rámce. Následující pole se nachází pod názvem *Data*, obsahující přenášená data a dále pak je bajt *FCS (Frame Check Sequence)* označující zabezpečující pole, jež detekuje vzniklé chyby přenosu, například ztráty bitů v rámci. Poslední bajt je opět *Flag*, který ukončuje rámec [5].

Obrázek 11: Obecný strukturovaný rámec protokolu HDLC

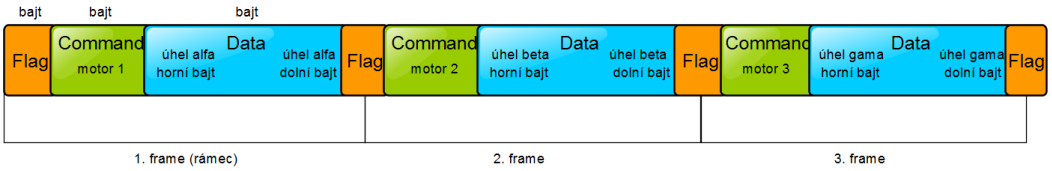
Předchozí struktura rámce protokolu HDLC byl obecný popis. V tomto projektu je použit jednodušší strukturovaný rámec HDLC, který lze nalézt na obrázku 12. Rámec je zjednodušený z důvodu nevyužitelnosti některých polí jako *Adress* a *FCS*.

Simplifikovaný strukturovaný rámec protokolu HDLC obsahuje stejně jako v předchozím případě počáteční a ukončující vlajkový bajt *Flag*. V druhém bajtu je pole označené jako *Command*, které má za úkol dávat povely motorům v přístroji. Následný bajt *Data* je označen stejně jako v obecném případě.

[10] High-level Data Link Control znamená v překladu vysoce úrovňová kontrola datové spojení [11].

Obrázek 12: Zjednodušený strukturovaný rámec protokolu HDLC, použitý v tomto projektu.

Pro názornou ukázku funkce zapouzdření do strukturovaného rámce protokolu HDLC slouží obrázek 13, na kterém je objasněn příklad zapouzdřených úhlů k odeslání do měřicího přístroje. Na obrázku jsou vidět spojené rámce. První rámec přenáší nastavený úhel alfa do motoru 1. V druhém rámci je bajt s označením *Flag* sdílen s prvním rámcem. Druhý frame přenáší nastavený úhel beta do motoru 2 a třetí frame obsahuje pole *Flag*, opět sdílené tentokrát s druhým rámcem a přenáší úhel gama do motoru 3.

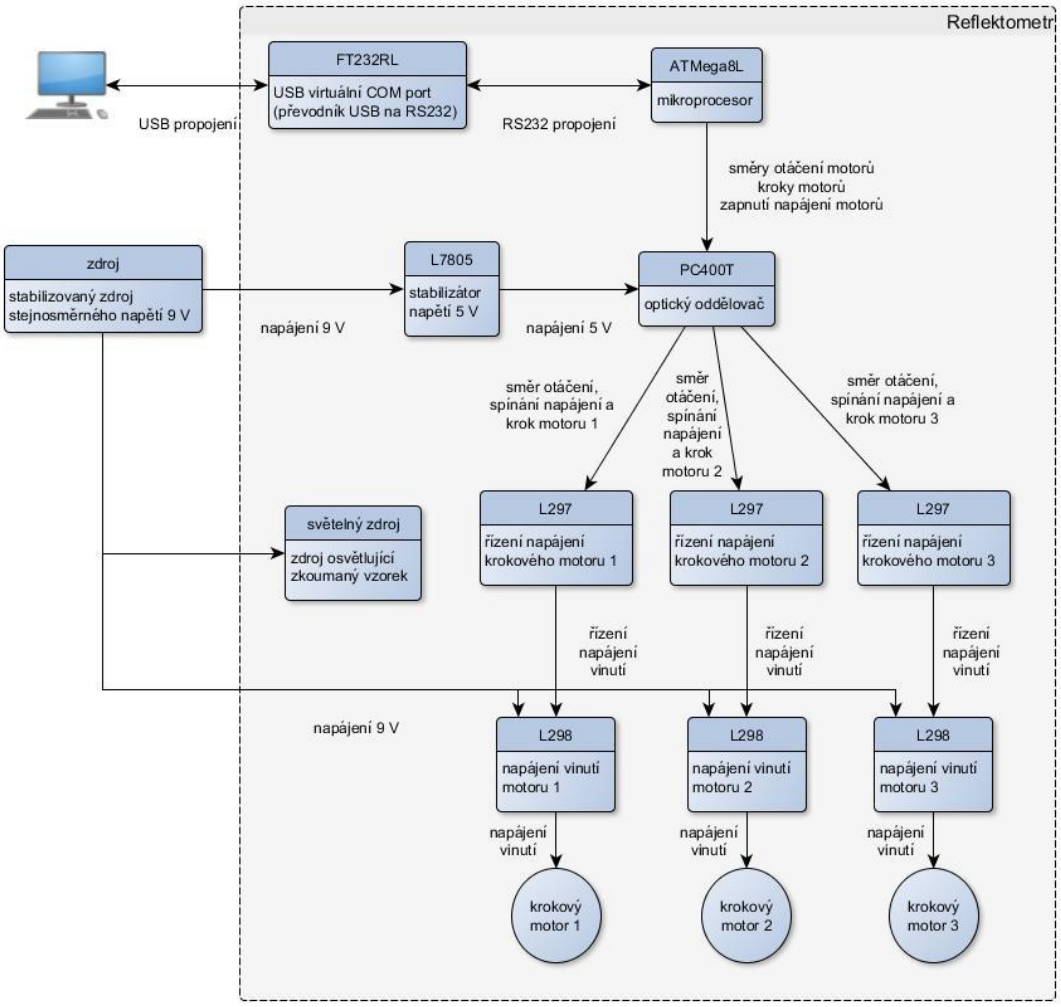


Obrázek 13: Příklad zapouzdření nastavených úhlů v řídicí aplikaci

V zapouzdřování nastává problém, ke kterému dochází, když přenášená data mají stejnou hodnotu jako bajt *Flag*, tedy 0x7E. Při této situaci dojde k úpravě nahrazením hodnoty 0x7E na hodnotu 0x7D a 0x5E. Tomuto nahrazení se říká *escape sequence*. Může nastat další problém jako v předešlé situaci, v již se bude jednat o hodnotu rovnající se 0x7D. Řešením je opět záměna. Místo hodnoty 0x7D se na její pozici vloží hodnoty 7D a 5D.

5.4.2 Popis reflektofotometru

Pro popis měřicího přístroje slouží diagram, který je na obrázku 14. Jednotlivé panely v diagramu představují komponenty reflektometru.

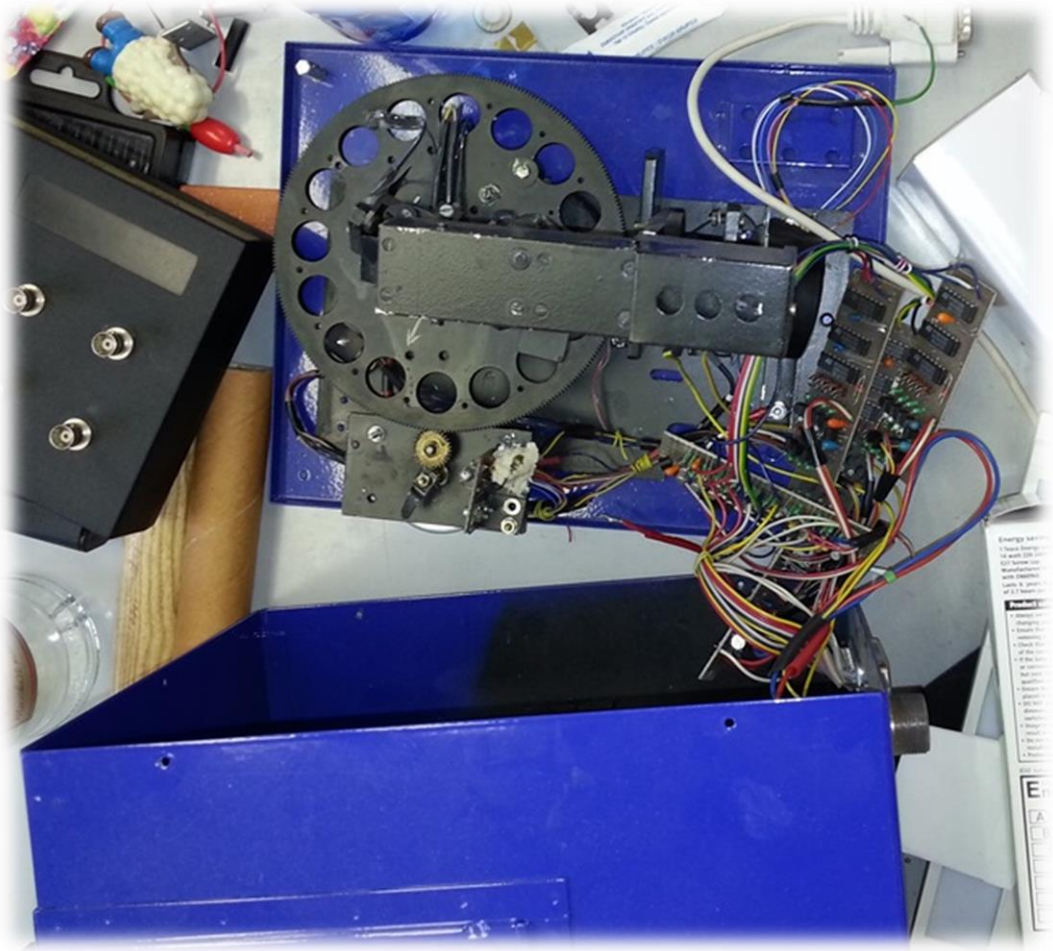


Obrázek 14: Popis reflektometru

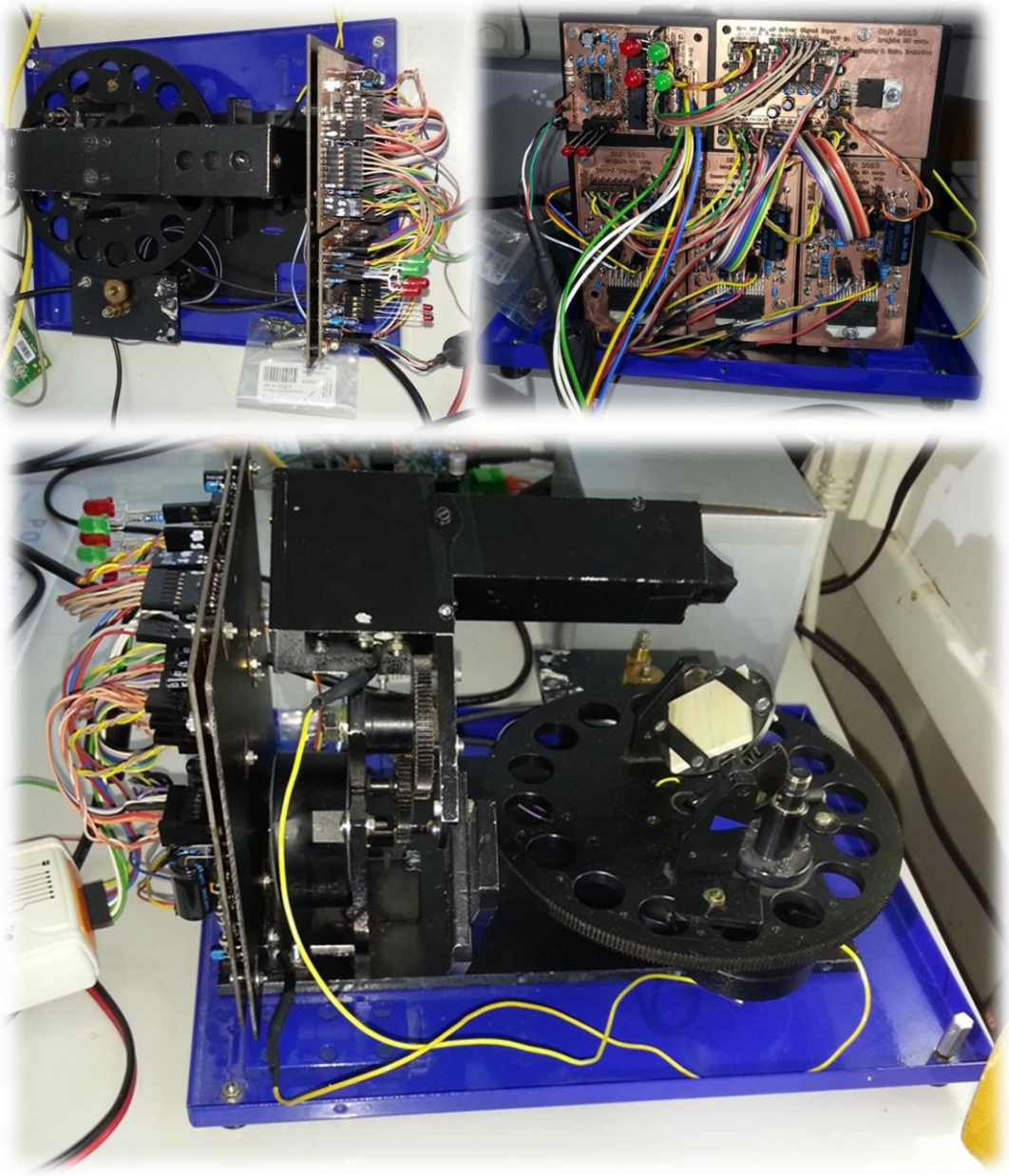
6 Závěr

Řídící aplikace pojmenována Reflectosoft je navržena pro intuitivní ovládání, aby mohla být bez jakýchkoliv zábran užívána k řízení reflektometru pro měření prostorového rozložení jasu od odrazných povrchů různých materiálů.

Ačkoli jsem měl za úkol provést první měření a zpracovat naměřená data, tak kvůli komplikacím s doděláním přístroje, se měření neuskutečnilo. Na obrázku 15 je zobrazen přístroj před zahájením práce a na obrázku 16 je ukázka rozpracovaného reflektometru.



Obrázek 15: Reflektometr před zahájením práce



Obrázek 16: Reflektometr v rozpracovaném stádiu

Řídicí aplikace je tedy ještě také ve fázi ladění a dokončování, nicméně je zcela připravena k zapouzdření zadaných úhlů do protokolu HDLC. Tyto zapouzdřené data jsou připravené k odesílání do přístroje. Přijímání o rozbalování dat z přístroje ještě není zcela hotové, ale aplikace je přichystána importovaná data ukládat do formátu .csv a řadit od nejnižších hodnot po nejvyšší. Vyřešeno je vyhledávání portů a dále pak je možné zprostředkovat spojení mezi přístrojem a aplikací, pomocí USB kabelu. Také je zhotoveno informační hlášení o stavu připojení s portem.

Za svůj přínos považuji sestavení základních částí aplikace pro další vývoj. Jde především o třídy určené pro komunikaci s reflektometrem, které jsou naprogramovány spořádaně a přehledně, aby se v případě potřeby daly doplňovat o další metody nebo je jednoduše měnit.

A na závěr bych chtěl ještě poznamenat, že do budoucna je naplánována eventualita nastavování úhlů pomocí zautomatizovaného krokování v určitém nastaveném rozmezí, aby uživatel nemusel vyplňovat každou hodnotu ručně.

Použitá literatura

[1] HABEL, Jiří. *Světlo a osvětlování*. Praha: FCC Public, 2013, 622 s. ISBN 978-80-86534-21-3.

[2] ŽÁRA, Jiří, Bedřich BENEŠ, Petr FELKEL a Jiří SOCHOR. *Moderní počítačová grafika*. Vyd. 1. Praha: Computer Press, 1998, xvi, 448 s. ISBN 80-722-6049-9.

[3] HEROUT, Pavel. *Java grafické uživatelské prostředí a čeština*. Vyd. 1. České Budějovice: KOOP, 2001, 316 s. ISBN 80-723-2150-1.

[4] BSDF Viewer. *Radiance-online* [online]. 2013 [cit. 2014-05-22]. Dostupné z: http://www.radiance-online.org/download-install/bsdf-viewer

[5] HDLC Protocol Description. *Interfacebus* [online]. 1998, 2012 [cit. 2014-05-22]. Dostupné z: http://www.interfacebus.com/HDLC_Protocol_Description.html

[6] HEROUT, Pavel. *Učebnice jazyka Java*. 5., rozš. vyd. České Budějovice: Kopp, 2010, 386 s. ISBN 978-80-7232-398-2.

[7] http://www.tutorialspoint.com/java/java_overview.htm

[8] Wikipedia. *Wikipedia* [online]. 2014 [cit. 2014-05-22]. Dostupné z: http://cs.wikipedia.org/wiki/API

[9] Archimedes. *Wikipedia* [online]. 2014 [cit. 2014-05-22]. Dostupné z: http://en.wikipedia.org/wiki/Archimedes#cite_ref-30

[10] BAYER, Rudolf. Automatizovaný systém měření činitele odrazu. In: BAYER, Rudolf. Automatizovaný systém měření činitele odrazu. Ostrava: Ediční středisko VŠB-TU Ostrava, 2013, s. 200-205. ISBN 978-80-248-3173-2.

Obrázek pod číslem 14 mi poskytl Ing. Rudolf Bayer, vedoucí práce.

Seznam příloh

V přiloženém CD je:

- postup instalace open-sourcové neoficiální knihovny RXTX pro Javu
- zdrojové kódy dosud naprogramované aplikace
- elektronická podoba bakalářské práce