Fakulta matematiky fyziky a informatiky

Univerzita Komenského v Bratislave

Záverečná správa

Vzdialené ovládanie spektrálneho analyzátora

zimný semester 2014/2015

Dávid Dobiáš

Dominik Dobiáš

Matej Hudec

Michal Mesároš

Obsah

1. Úvod 4
1.1 Predmet špecifikácie 4
1.2 Rozsah projektu 4
1.3 Slovník pojmov a skratky 4
1.4 Odkazy 5
2. Celkový popis 5
2.1 Kontext systému 5
2.1.1 Používateľské rozhrania 6
2.1.2 Hardvérové rozhranie 6
2.1.3 Softvérové rozhranie 6
2.2 Funkcie systému 6
2.3 Triedy používateľov a ich vlastnosti 6
2.4 Budúca verzia systému 6
3. Špecifikácia požiadaviek 7
3.1 Vzdialené ovládanie cez LAN 7
3.2 Nastavenie analyzátora 7
3.3 Vykonanie kalibrácie 7
3.4 Uskutočnenie merania 8
3.5 Analýza dát 8
4. Ďalšie požiadavky 8
4.1 Výkonnostné požiadavky 8
4.2 Dostupnosť 8
4.3. Bezpečnostné požiadavky 8
5. Konceptuálna analýza 9
5.1 Stavové diagramy 9
5.2 Entitno-relačný diagram 12
5.3 Use-case diagram 13
5.4 Užívateľské rozhranie 13
5.5 Legenda 13
6. Analýza technológií 14

6.1 Používané technológie 14
6.1.1 Možnosti technológií pripojenia SVOSA k SA: 14
6.1.2 Hardwarové technológie 15
6.1.3 Softwarové technológie 15
7. Komponentný diagram 16
7.1 Komponent prvotné nastavovanie 16
7.2 Komponent meranie 17
7.3 Komponent spracovanie dát 17
7.4 Komponent uloženie dát 17
7.5 Komponent kreslič grafov 17
8. Triedny diagram 18
9. Dátový model zobrazený pomocou entitno-relačného diagramu 19
10. Testovacie scenáre 20
10.1 rlcparams 20
10.2 agilent 20
10.3 Popis validného testovacieho scenára 22
10.4 Ďalšie možné testovacie scenáre 22
11. Testovanie 23
11.1 Testovanie modulu agilent_SCPI 23
11.2 Testovanie hlavného modulu main 23
11.3 Testovanie fungujúceho modulu agilent 23
12. Záznam z odovzdania a predvedenia diela zadávateľovi 25
12.1 Plán stretnutia 25
12.2 Verzie a zmeny 25
13. Zhodnotenie 26
13.1 Objektívne hodnotenie celého týmu 26
13.2 Zhodnotenie Dávida Dobiáša 26
13.3 Zhodnotenie Michala Mesároša 27
13.3 Zhodnotenie Dominika Dobiáša 28
13.3 Zhodnotenie Mateja Hudeca 28

1. Úvod

1.1 Predmet špecifikácie

Tento dokument (Katalóg požiadaviek , ďalej už len ako KP) slúži ako špecifikácia požiadaviek pre prácu na projekte **implementáciu softvéru pre vzdialené ovládanie spektrálneho analyzátora** (ďalej ako ISVOSA) , ktorý spracováva fyzikálne údaje a zobrazuje ich na svojej obrazovke. Projekt má slúžiť na uľahčenie práce vedeckých pracovníkov, ktorí robia s týmto zariadením.

Dokument je súčasťou dohody medzi objednávateľom a dodávateľom. Bude slúžiť ako východisko pre vyhodnocovanie správnosti softvéru.

1.2 Rozsah projektu

Projekt má slúžiť na diaľkové ovládanie zariadenia, ktoré sa doteraz musí ovládať tlačidlami priamo na ňom umiestnenými.

Požiadavkou je systém, ktorý sa bude dať ovládať tak, že po vložení vzorky do spektrálneho analyzátora, pracovník môže dokončiť úlohu z iného zariadenia napojeného na sieti a sledovať priebeh testov, ktoré aktuálne prebiehajú.

Taktiež bude vedieť experiment prerušiť a meniť parametre experimentu a následné pokračovať v experimente pokiaľ však samozrejme, nebude analyzátor vyžadovať fyziku manipuláciu so vzorkami v ňom.

1.3 Slovník pojmov a skratky

KP- Katalóg požiadaviek, to je tento konkrétny dokument

ISVOSA - Implementácia softvéru pre vzdialené ovládanie spektrálneho analyzátora, meno projektu, ktorého sa týka KP.

SVOSA - softvéru pre vzdialené ovládanie spektrálneho analyzátora.

SA - Spektrálny analyzátor je základné meracie zariadenie ktoré meria veľkosť vstupného signálu a frekvenciu v plnom frekvenčnom rozsahu nástroja.

WiFi- je bezdrôtová technológia pre lokálne siete, ktorá umožňuje elektronickým zariadeniam výmenu údajov alebo pripojenie na internet.

TCP- je spojovo orientovaný, spoľahlivý komunikačný protokol transportnej vrstvy prenášajúci bajtový tok. Protokol zaručuje, že dáta odoslané z jedného konca spojenia budú prijaté na druhej strane spojenia v rovnakom poradí a bez chýbajúcich častí.

SICL-LAN – knižnica ktorá rieši pripojenie k hardvérovým laboratórnym zariadeniam cez lokálnu sieť

Agilnet I/O – knižnica umožnujúca komunikáciu SA s rôznymi vývojárskymi prostrediami (Agilent VEE Pro, Microsoft Visual Studio, atď.)

Ethernet je technológia počítačových sietí pre lokálne siete (LAN). Ethernet je založený na nápade, že počítače v sieti budú posielať správy spôsobom, ktorý pripomína rádio, ale prostredníctvom spoločného kábla alebo kanála, niekedy označovaného ako éter (ether - preto **Ethernet**).

KJFaB - katedre jadrovej fyziky a biofyziky na FMFI UK

FMFI UK – Fakulta matematiky fyziky a informatiky Univerzity Komenského

Spúšťací systém- je to systém ktorý má nastarosti úkony ako detekovanie štartu merania ako aj dovoliť/zakázať merania pre každé z vlákien. Spúšťací systém má dva stavy a to "system-wide" a "channel-wide", kde "system-wide" môže byť v režime "Hold", "Waiting for Trigger" alebo "Measurement" a "channel-wide" môže byť v režime "Idle" alebo "Initiate". [4]

1.4 Odkazy

[1] Dokumentácia pre programátorov k SA – ena-l_programmers_eng.pdf, kapitola Overview of Remote Control

[2] Dokumentácia pre programátorov k SA – ena-l_programmers_eng.pdf, kapitola Reading/Writing Measurement Data

[3] Dokumentácia pre programátorov k SA – ena-l_programmers_eng.pdf,kapitola Setting Up Analyzer

[4] Dokumentácia pre programátorov k SA – ena-l_programmers_eng.pdf, kapitola Making Meausurement

[5] Dokumentácia pre programátorov k SA – ena-l_programmers_eng.pdf, kapitola Analyzing Data

2. Celkový popis

2.1 Kontext systému

Systém je určený používateľovi na ovládanie SA, kde bude môcť nadstaviť všetky rôzne nadstavenia, ktoré sú mu aj prístupné pri ovládaní priamo na SA. Bude ho tiež vedieť uviesť do činnosti, čí napríklad pozastaviť činnosť . Získané hodnoty sa budú v koncovom zariadení, ktoré bude pripojene na SA zobrazovať a podľa toho bude vedieť používateľ, čo sa naozaj deje, a to s malou odozvou, ktorú zabezpečí ethernet pripojenie.

2.1.1 Používateľské rozhrania

Používateľské rozhranie bude vytvorené formou jednoducho ovládateľnej aplikácie vytvorenej v jazyku Python. Bude sa dať ovládať perifémymi zariadeniami – myš, klávesnica (možnosť používania klávesových skratiek na rôzne úkony).

2.1.2 Hardvérové rozhranie

Systém ma fungovať na súkromnej sieti pomocou technológie ethernet, ktorú SA podporuje a aj zariadenia na druhej strane ho musia podporovať. Väčšinou sa na zariadenie budú používatelia napájať buď cez ethernet kábel, ale aj inak napríklad keď bude SA zapojený do sieťového prvku siete (napríklad routra), tak aj pomocou iných technológii, ktoré podporuje sieťový prvok a koncové zariadenie súčasne (napríklad Wifi).

2.1.3 Softvérové rozhranie

Systém bude naprogramovaný v jazyku Python a bude schopný komunikovať s koncovým zariadením pomocou protokolu TCP.

2.2 Funkcie systému

Systém bude fungovať na obojstrannej komunikácii medzi SA (zariadením) a vedeckým pracovníkom (používateľom) pomocou vzdialeného prístupu. Systém bude schopný prijímať dáta zo SA a zobrazovať ich v textovom formáte pre potrebu ďalšieho vyhodnocovania takto na diaľku získaných dát . Ďalšou hlavnou funkciou systému bude nastavovanie podmienok pre meranie podobným spôsobom ako tomu je priamym nastavovaním na SA. Takisto bude možné meranie pozastaviť a meniť parametre a nastavenia za chodu merania. Toto všetko zabezpečí sieťová komunikácia cez ethernet kábel.

2.3 Triedy používateľov a ich vlastnosti

Systém bude mať jedinú triedu používateľov. Bude ňou vedecký pracovník. Je to ľubovoľná osoba s oprávnením pracovať na SA. Táto osoba bude môcť využívať úplnú funkcionalitu SVOSA prostredníctvom používateľského rozhrania priamo zo svojho osobného počítača (alebo ľubovoľného iného výpočtového zariadenia na ktorom bude SVOSA nainštalovaný.) prepojeného ethernet káblom so SA.

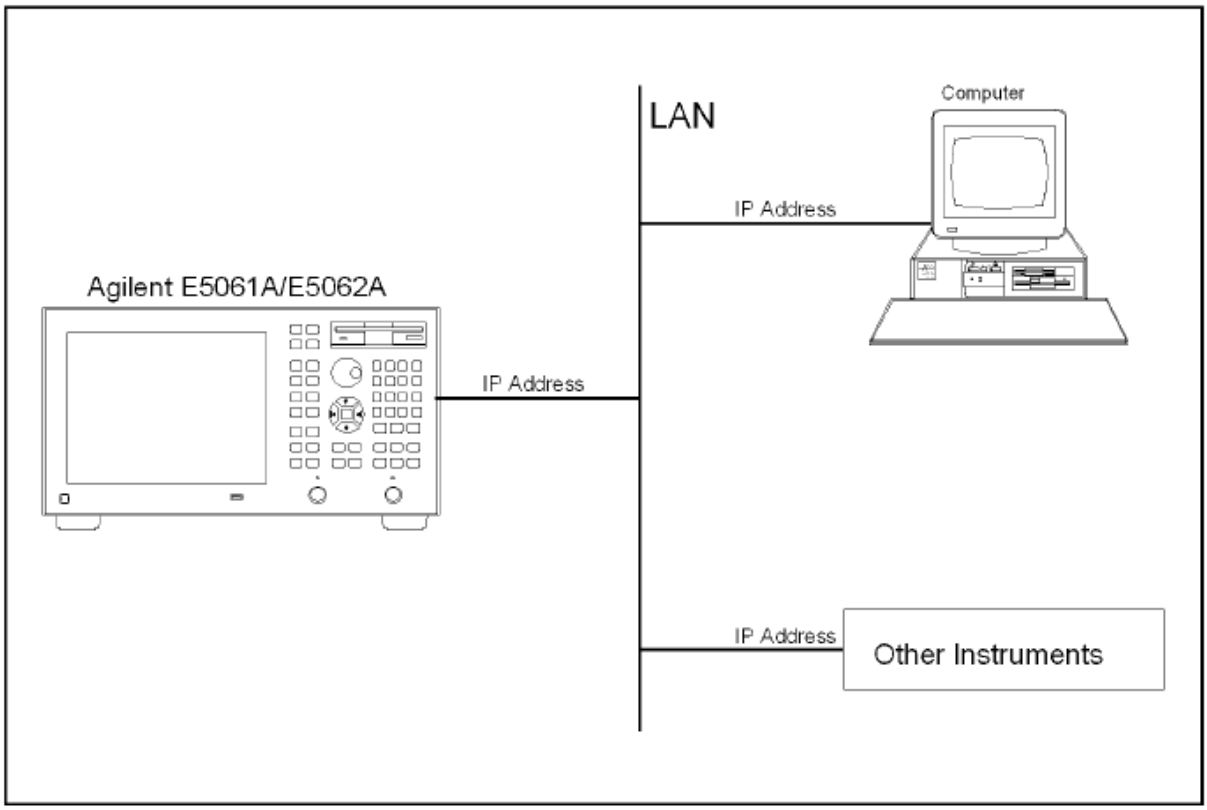
2.4 Budúca verzia systému

Tak ako je súčasná verzia rozšírením a upravením starej verzie softvéru na vzdialené ovládanie staršieho SA, v prípade renovácie inventáru a zakúpenie ďalšieho ešte novšieho SA na katedre jadrovej fyziky a biofyziky (ďalej už len KJFaB), bude možné tento celý systém príslušne upraviť a ďalej využívať.

3. Špecifikácia požiadaviek

3.1 Vzdialené ovládanie cez LAN

SVOSA musí nadviazať spojenie so SA prepojenými lokálnou sieťou LAN použitím SICL-LAN serveru a za pomoci Agilnet I/O knižnice .



Obrázok 3.1: Schéma prepojenia zariadení [1]

3.2 Nastavenie analyzátora

Uživaťeľ bude mať možnosť nastaviť

1. aktívny kanál/stopu [3]
2. maximálny počet stôp
3. meracie parametre pre každú stopu [3]
4. “sweep” podmienku [3]
5. zobrazovacie parametre [3]

3.3 Vykonanie kalibrácie

Užívateľ bude vedieť kalibrovať SA a to v nasledujúcich krokoch :

1. užívateľ vyberie kalibračnú sadu
2. užívateľ vyberie typ kalibrácie
3. SVOSA odošle povel na výpočet kalibráčných koeficientov (1, 2)

3.4 Uskutočnenie merania

1. užívateľ bude môcť zahájiť meranie pomocou spúštacieho systému
2. užívateľ bude môcť sledovať priebeh merania
3. užívateľ bude vedieť zistiť koniec merania
4. užívateľ bude môcť sledovať výsledky merania

3.5 Analýza dát

1. SVOSA získa namerané dáta zo SA v bode merania určenom požívateľom
2. SVOSA bude vedieť vyhľadať pozíciu spĺňajúcu užívateľom špecifikované kritériá pomocou označovačov. [5]

4. Ďalšie požiadavky

4.1 Výkonnostné požiadavky

Na aplikáciu sa nekladú žiadne výkonnostné požiadavky.

4.2 Dostupnosť

Aplikácia bude dostupná na internej (súkromnej) sieti.

4.3. Bezpečnostné požiadavky

Zariadenie je napojené na súkromnej sieti, takže zabezpečenie nie je potrebné riešiť. Zariadenie sa ovláda cez sieť (ethernet) pomocou nášho systému alebo ručné priamo na zariadení.

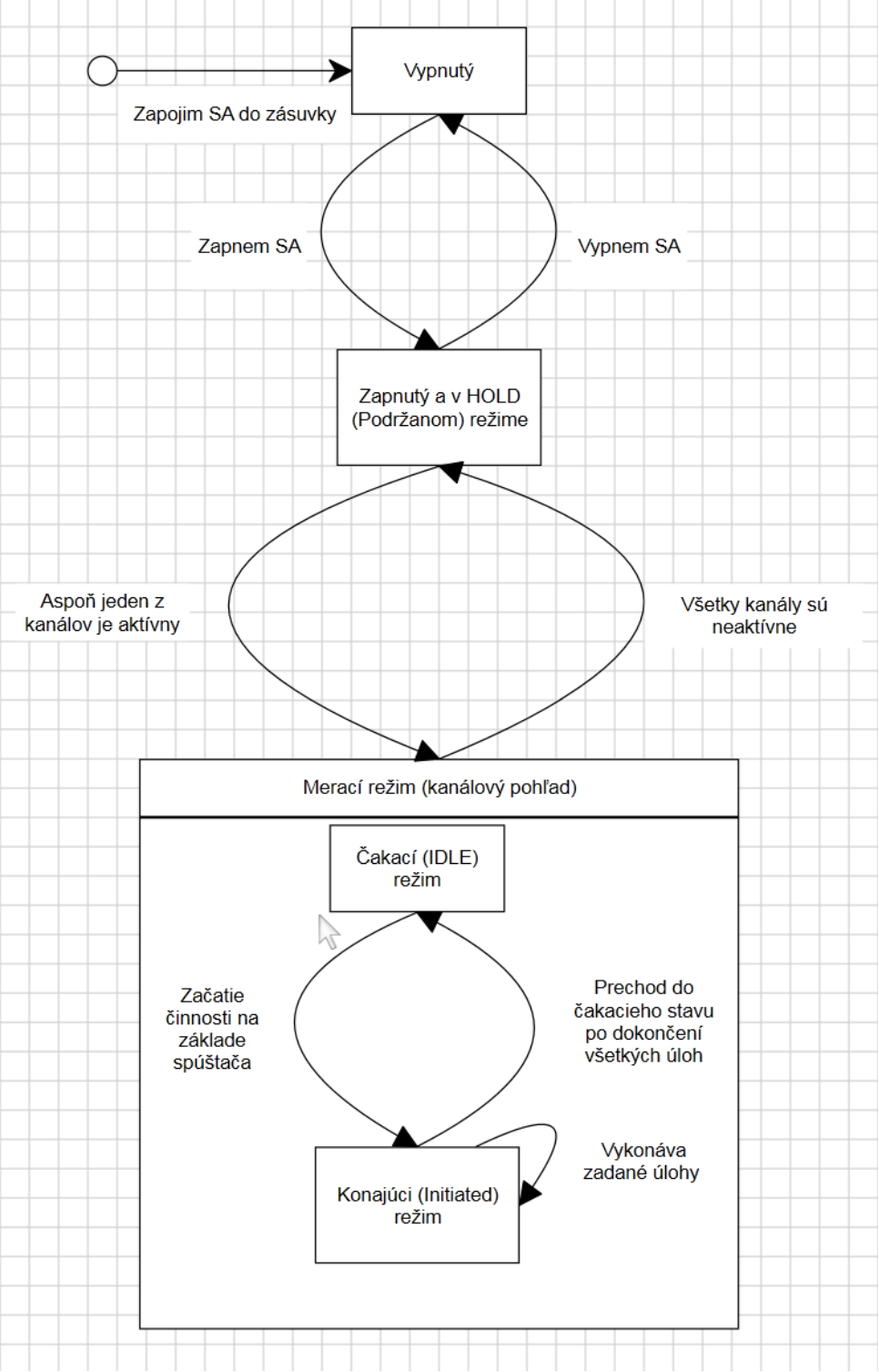
5. Konceptuálna analýza

Tento dokument slúži na logičke odvodenie požiadaviek z platného katalógu požiadaviek, súvislosti a prezentáciu základných dát. Obsahuje diagramy to entitno-relačný diagram, stavové diagramy, a use-case diagram. Na konci tohto dokumentu je legenda s vysvetlivkami pojmov a popis užívateľského rozhrania.

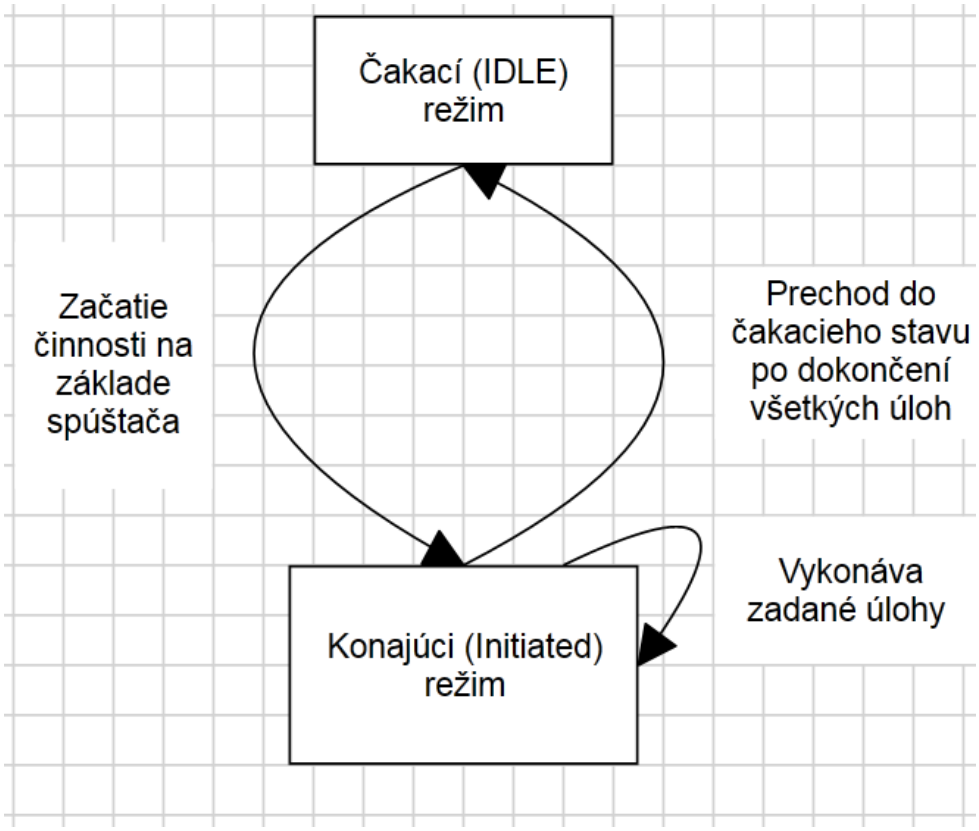
5.1 Stavové diagramy

Diagramy vysvetľujú ako z pohľadu SA funguje proces experimentu, teda jeho hlavnej činnosti.

Bližšie detaily sú vysvetlené v dokumentácii [2] na 81 strane.



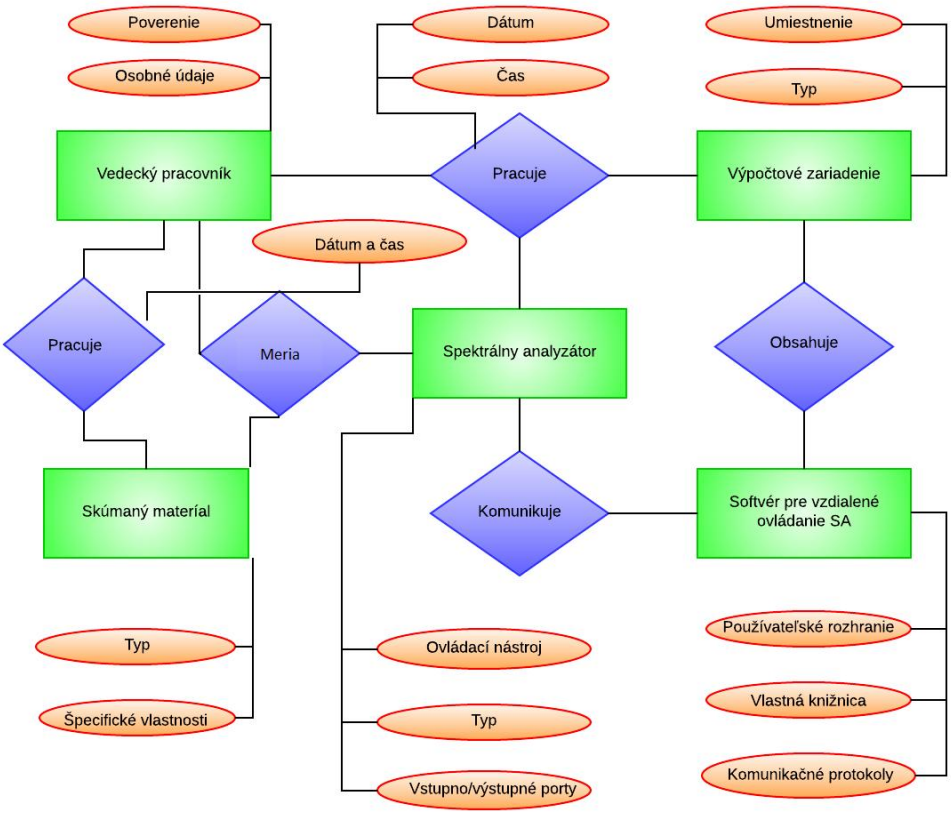
Obrázok 5.1 : Vysvetľuje systémový pohľad na proces experimentu z pohľadu SA.



Obrázok 5.2: Vysvetľuje kanálový pohľad na proces experimentu z pohľadu SA. Samotný SA vie vykonávať experiment vo viacerých kanáloch, čo zvyšuje rýchlosť experimentu, lebo vie robiť viac veci naraz.

5.2 Entitno-relačný diagram

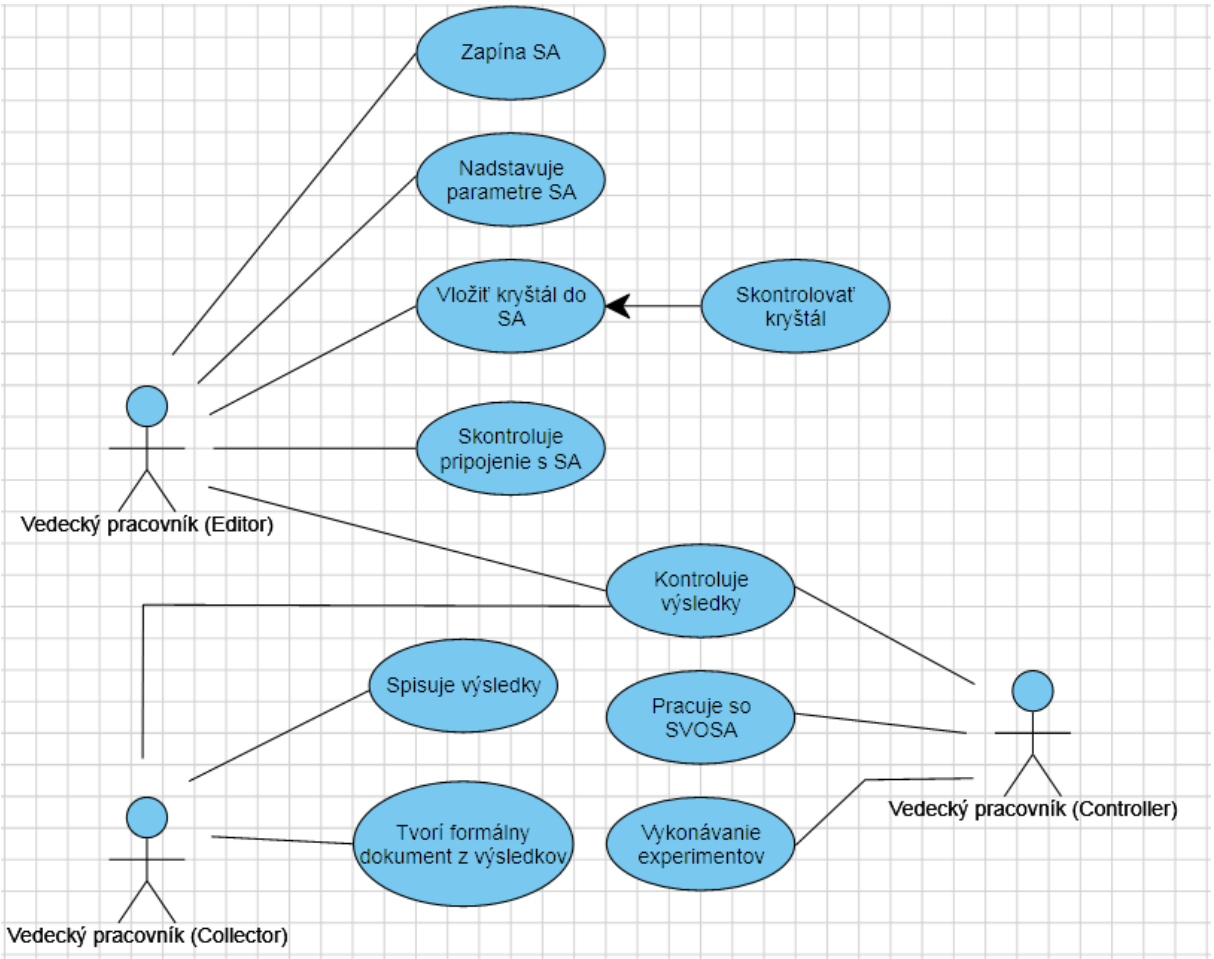
Diagram znázorňuje jednotlivé entity vstupujúce do experimentu a ich vzťahy (relácie) medzi sebou v priebehu tohto experimentu.



Obrázok 5.3: Znázorňuje jednotlivé vzťahy medzi entitami a ich atribúty v procese experimentu.

5.3 Use-case diagram

Use-case diagram je najjednoduchší spôsob reprezentácie vzťahov medzi užívateľom systému popísaný stavmi použitia. Tento diagram popisuje, ako užívateľ reaguje počas experimentu so SA a so softvérom SVOSA.



Obrázok 5.4: Popisuje interakcie užívateľov so systémom. Jednotlivé role môže vykonávať aj jeden človek, ale aj skupina.

5.4 Užívateľské rozhranie

Užívateľské rozhranie bude spočívať vo forme príkazového riadku, do ktorého používateľ bude volať procedúry a metódy SVOSA. Užívateľ potom môže dodefinovať ďalšie argumenty k programu. Ak uvedenie 1 argument, tak ten je definovaný ako dvojica frekvencie a impedancie. Ak uvedie ešte jeden parameter, bude to meno súboru, do ktorého sa uložia dáta do prehľadnej tabuľky (do relatívneho priečinka odkiaľ bol súbor spustený). Ostatné argumenty budú ignorované.

5.5 Legenda

SA - Spektrálny analyzátor.

SVOSA - Softvér pre vzdialené ovládanie spektrálneho analyzátora.

6. Analýza technológií

Ako bolo stanovené v sekcii 2.1.1 Používateľské rozhrania" projekt bude písaný v programovacom jazyku Python.

6.1 Používané technológie

6.1.1 Možnosti technológií pripojenia SVOSA k SA:

1. LAN - (Local Area Network) systém vzdialeného ovládania poskytuje dve metódy :

    6. SICL-LAN – v ovládacom systéme používajúcom SICL-LAN server, komunikácia medzi vonkajším ovládačom (klientom) a SA (serverom) je uskutočnená použitím SICL-LAN protokolu. Samotná komunikácia je uskutočnená pomocou SICL . Užívateľ môže ovládať SA programovaním pomocou

        1. SICL (Standard Instrument Control Library) - je vstupno/výstupná knižnica vyvýjaná Agilent (HP) ktorá je kompatibilná s mnohými rozhraniami a systémami. SICL je komunikačná knižnica pre zariadenia ktorá funguje s veľkou škálou počítačových architektúr, vstupno/výstupných rozhraní a operačných systémov. Aplikácie napísané v C/C++ alebo Visual Basic, používajúce túto knižnicu, môžu byť prerobené z jedného systému do druhého bezo alebo iba s veľmi málo zmenami.

        2. VISA (Virtual Instrument Software Architecture) - je je vstupno/výstupná knižnica vyvýjaná vzhľadom na VXI plug&play System Alliance štandardy, ktorá umožnuje aby softvér vyvýjaný rôznymi výrobcami fungoval na rovnakom sýstéme. VISA sa používa ak vývojár chce zachovať VXI plug&play ovládače vo svojej aplikácii alebo chce aby jeho aplikácii bola v súlade s VXI plug&play štandardmi.

    7. Telnet – v ovládacom systéme cez telnet server, komunikácia je uskutočňovaná cez pripojenie medzi zásuvnými modulmi poskytovanými procesmi vonkajšieho ovládača a SA na nadviazanie sieťového spojenia medzi nimi. Ovládací systém cez telnet môže komunikovať dvoma portami a to:

        1. port 23 – používa sa na dialógové ovládanie používajúce telnet (užívateľské rozhranie pre TELNET protokol) (existujúci softvér)

        2. port 5025 – používa sa na ovládanie z programu. (použitie pri implementácii telnet komunikácie vo vlastnom softvéri)

2. GBIP – (General Purpose Interface Bus) je špecifikácia digitálnej komunikácie zariadení na krátku vzdialenosť a štandardné rozhranie pre pripojenie počítačov a periférnych zariadení, ktoré podporuje nasledujúce medzinárodné štandardy : IEEE 488.1,IEC-625, IEEE 488.2, a JIS-

C1901 . GPIB rozhranie umožňuje ovládať Agilent SA z externého počítača. Počítač odosiela príkazy a pokyny na SA a prijíma dáta odoslané z SA cez GPIB .

Výber technológie pripojenia SVOSA k SA

Na základe špecifikácie požiadaviek sme sa rozhodli uprednostniť LAN technológiu, keďže má byť SA ovládaný cez lokálnu sieť, s použitím metódy SICL-LAN kôli portabilite a formátovanému vstupu/výstupu, konrétne SICL, kôli kompatibilite z viacerými systémami a architekrúrami.

6.1.2 Hardwarové technológie

1. Náš systém bude používať technológiu TCP/IP na spojenie sa s koncovým zariadením. Očakáva sa že to bude väčšinou notebook alebo PC. Prípadné rozšírenie je zapojenie zariadenia do sieťového prvku (napríklad rozbočovač).

6.1.3 Softwarové technológie

Náš systém bude používať rôzne knižnice pre dosiahnutie konkrétnych požiadaviek

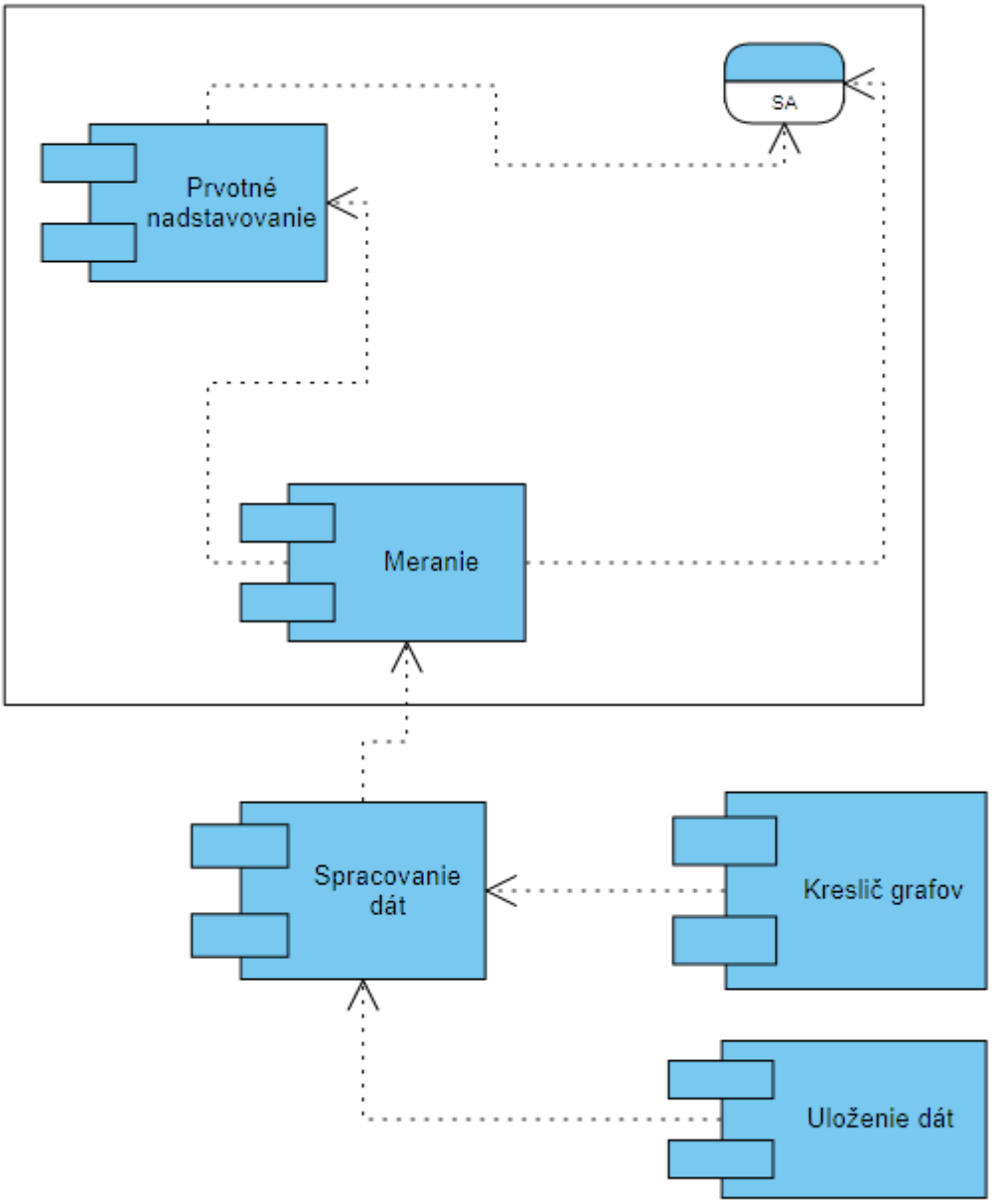
1. Na vykresľovanie grafov budeme používať knižnicu "Gnuplot", ktorá dokáže vykresľovať rôzne typy grafov. Všeobecnejšie sa dá používa aj ako nástoj v Linuxe na kreslenie grafov cez príkazový riadok.

2. Taktiež na nízko úrovňovú sieťovú komunikáciu s SA budeme potrebovať aj knižnicu menom "socket". Budeme vedieť pomocou nej sa pripojiť na konkrétnu IP adresu na konkrétny port nášho SA. Konkrétne sa najprv inicializuje socket . Vyberie sa adresnú rodinu konkretne : "socket.AF_INET" ,ktorá je používaná pre IP adresovaný socket. Skratka AF_INET je skratka od Adress Family InterNET. Taktiež si pri inicializácii akým spôsobom má komunikovať. My sme si zvolili " socket.SOCK_STREAM " čo nám poskytuje sekvenčný, spoľahlivý, obojsmerný , spojitý tok byte-ov.

Potom sa napojíme na ip adresu zariadenia v sieti (konkrétne táto : 192.168.1.102) a na príslušný port (konkrétne 5025). Potom už len posielame správy SA v textovej forme alebo si od neho pýtame určité množstvo textu (konkrétne cyklicky 1024 znakov pokiaľ uznáme za vhodné). A na koniec spojenie ukončujeme príkazom close(), ktorý štandardne ukončuje spojenie medzi spojenými zariadeniami.

3. Knižnica "math" budeme používať na výpočet matematických vzorcov a volanie math.atan2() funkcie, ktorá ráta arkus tangens a vracia výsledok v radiánoch. Pričom táto funkcia má 2 parametre, ktoré musia byť čísla.

7. Komponentný diagram



Obrázok 4.1 : Komponentný diagram popisujúci jednotlivé komponenty systému. V rámčeku je vyznačená oblasť, ktorú má riešiť náš projekt a ostatné komponenty mimo neho sú komponenty, ktoré sú už spravené a budú používané našim systémom.

7.1 Komponent prvotné nastavovanie

Komponent slúži na nastavenie všetkých potrebných parametrov potrebných na napojenia sa pomocou siete na SA. Okrem iného sa v ňom nastavujú dôležité konštanty ako napríklad adresa zariadenia na siete či príslušný port, ktoré užívateľ ďalej nemôže meniť. Tu sa budú dať aj nastaviť ostatné parametre ako bolo už definované v katalógu požiadaviek.

7.2 Komponent meranie

Komponent slúži na posielanie pokynov na meranie a ich spätné prijímanie v nastavenom formáte. Dáta sa ukladajú do internej štruktúry. Spracovanie dát zo štruktúry je už spracúvaný iným systémom, ktorý dátam rozumie.

7.3 Komponent spracovanie dát

Komponent prečíta dáta z internej štruktúry a preloží do ďalších potrebných štruktúr a počítajú sa potrebné veličiny z dodaných dát pre ich ďalšie využitie.

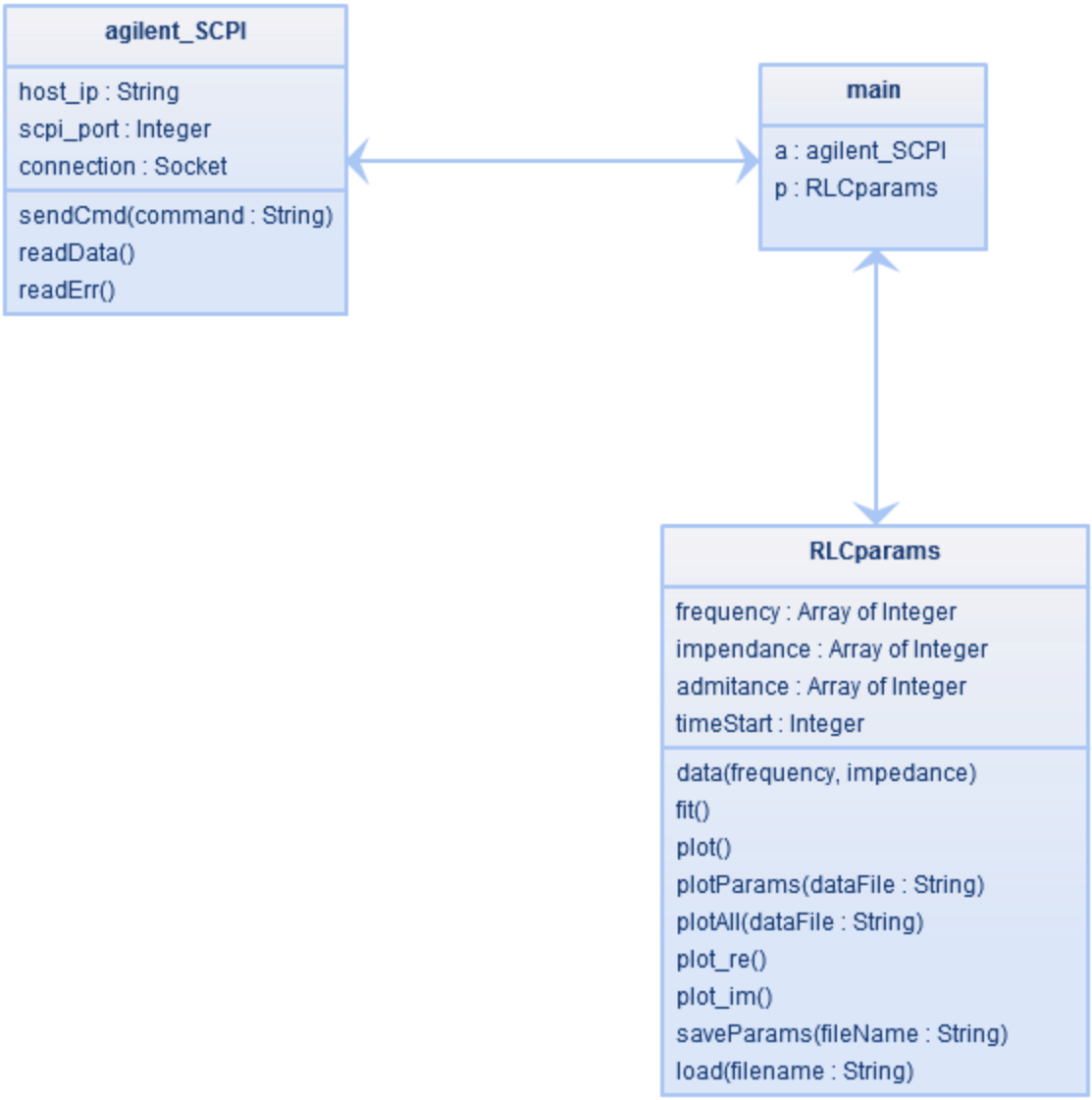
7.4 Komponent uloženie dát

Dáta sa ukladajú do prehľadnej tabuľky, ak sa užívateľ rozhodne inak sa dáta nebudú ukladať.

7.5 Komponent kreslič grafov

Z poskytnutých spracovaných a vyrátaných dát a veličín dokáže tento komponent zobraziť potrebné grafy.

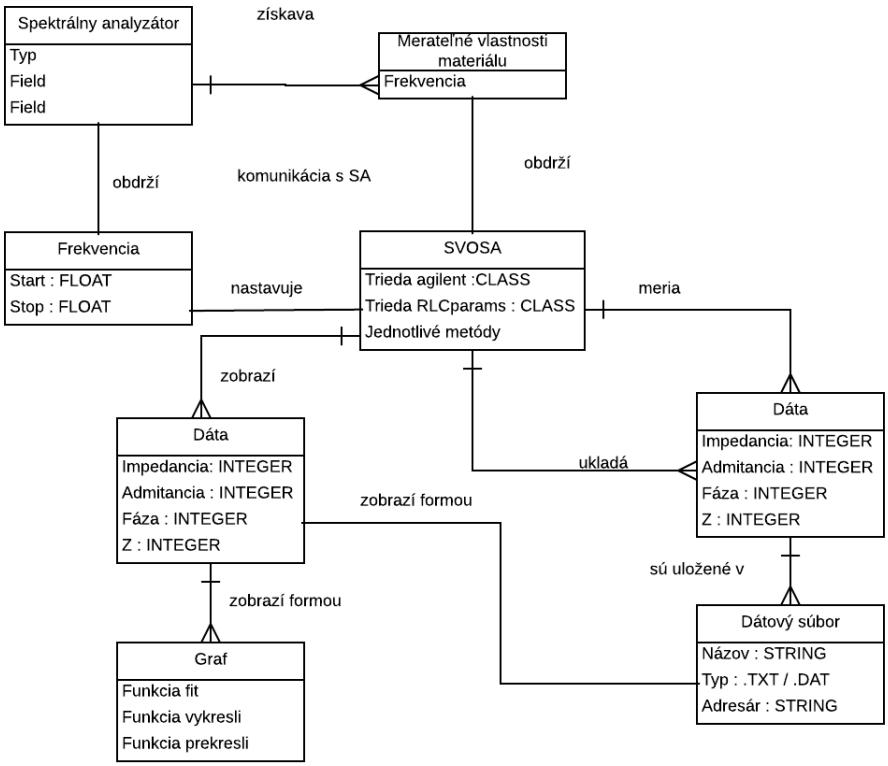
8. Triedny diagram



Obrázok 5.1 : Triedny diagram nášho systému. Sú v ňom ukázané mená jednotlivých premenných a mena metód jednotlivých tried.

Triedny diagram znázorňuje štruktúru projektu, popisuje jeho triedy a ich vzájomné prepojenie. Hlavný program main spravuje a inicializuje triedy agilent_SCPI a RLCparams. SVOSA pomocou triedy agilent_SCPI nadviaže spojenie zo SA, a následne pomocou tejto triedy aj príjme dáta, ktoré po úspešnom prijatí zanalyzuje a graficky znázorní pomocou triedy RLCparams.

9. Dátový model zobrazený pomocou entitno-relačného diagramu



Obrázok 6.1: Podrobnejší entitno-relačný diagram znázorňujúci dátový model Informačného systému.

SVOSA pracuje a analyzuje získané dáta od SA a tie následne dokáže graficky znázorňovať, alebo ich dokáže ukladať (exportovať) do textového, alebo dátového súboru s ktorým môžu pracovníci ďalej pracovať.

10. Testovacie scenáre

Pri používaní spektrálneho analyzátora môže dôjsť k rôznym situáciám. Tomuto sa venuje kapitola testovacie scenáre, ktorá popisuje rôzne prípady, aké môžu nastať počas merania na spektrálnom analyzátore pri použití SVOSA. Pri spustení programu sa najskôr inicializujú triedy (vytvoria sa inštancie objektov) Agilent a RLCparams aby sa s nimi dalo ďalej pracovať, nachádzajúce sa v samostatných jednotkách (agilent.py, respektíve rlcparams.py). Nasleduje popis tried a priebeh testovania v rámci behu kódu:

10.1 rlcparams

Triede RLCparams sa veľmi podrobne venovať nebudem. Súvisí s hlbšou matematikou a fyzikou. Nachádza sa tu konštruktor triedy, kde si inicializujeme atribúty triedy na nulové(prázdne hodnoty). Atribúty frequency, impedance, admitance, vyrez, vyref, vyrezimag, vyref_fit, yre_fit, yim_fit sú štruktúrované ako polia (lists) ďalej tu sú atribúty: R1, C0, C1, L1, fResonance, fResonanceFit, gnuplot, gnuplot1 , gnuplot2 (objekty Gnuplot.Gnuplot()) a timeStart. Trieda má metódy : data(self, frequency, impedance) - nastavenie dát; fit(self) - výpočet rezonancie pomocou fitovacej funkcie; metóda plot(self) - metóda na grafické zobrazenie celej krivky a výrezu a metódy plotParams(self, dataFile); plotAll(self, dataFile); plot_re(self) ; plot_im(self); saveParams(self, fileName = '') a load(filename). Táto trieda je plne funkčná a preto ju nebude treba špeciálne testovať.

10.2 agilent

V triede agilent má 3 základné atribúty = gnuplot = None a 2 atribúty typu slovník: data = {} a frequency = {}. Ako prvú vec si pri inicializácii vytvoríme objekt agilent_scpi, ktorý inicializujeme. Táto trieda sa nachádza v jednotke(unite) agilent_scpi.py a nach=adzaj=u sa v nej metódy určené priamo pre komunikáciu so SA : metóda na odosielanie príkazov(povelov): send_cmd, metóda na čítanie prichádzajúcich dát zo SA - read_data a metóda na zber chybových správa (errorov) - read_err .Trieda má 3 atribúty: _host_ip - tu sa nastavuje a ukladá ip adresa prístroja ('192.168.1.102'), atribút _scpi_port - kde sa nastavuje komunikačný port (5025) a atribút _connection: kde sa pomocou knižnice socket nastaví vhodný protokol, kde sa zadá ip adresa a port prístroja. Pri inicializovaní triedy si zadefinujeme v slovníku data pojmy: ['frequency'], ['Z'], ['impedance'], ['admitance'] a ['phase']. Ďalej ideme nastavovať SA prvým príkazom pomocou send_cmd a tým je príkaz ':syst:pres' (preset), ktorý nastaví SA na pôvodné továrenské nastavenie, kde je nastavený aj východiskový kanál, ako kanál 1. Ďalej nastavujeme formát dát z viacerých možných budeme v testoch používať formát typu Smythovho diagramu $(R+jX)$ pomocou commandu ':calc1:form smit'. Potom nastavujeme východiskový režim na nepretržitý pomocou volania príkazu ':init1:cont on' a následne sa zavolá abort pomocou príkazu ':abor' , ktorý zastaví merania a zmení spúšťacie poradie pre všetky kanály do u pokojného stavu. Ďalej nastavíme počet meracích bodov (hustota zobrazovania nameraných údajov -

sweeping point) v ukážkovom teste na maximálnu možnú hodnotu pre náš typ SA, pre väčšiu presnosť merania a tou je 1601 bodov (minimum sú 2) príkazom ':sens1:swe:poin 1601'. V ďalšom kroku nastavíme formát pre prenos dát na ASCii formát vďaka príkazu ':form:data asc' a povieme analyzátoru, aby počkal na vykonanie všetkých príkazov, ktoré boli doposiaľ analyzátoru zaslané. Nasleduje opýtanie sa na počiatočnú hodnotu sweep rangu - krivky(grafu)zobrazujúcej výsledky merania (pýtame sa na frekvenciu, čiže výslednú hodnotu dostávame v Hertzoch) pomocou commandu ':sens1:freq:star?' a načítame túto odpoveď do slovníka frequency ako jeho zložku ['start'] ako reálne číslo typu float pomocou funkcie read_data(). Takisto sa opýtame na koncovú hodnotu frekvencie príkazom ':sens1:freq:stop?' a toto si zapíšeme do slovníka frequency a jeho zložky ['stop'] podobným spôsobom ako pri start. Ďalej nasleduje samotná inicializácia Gnuplotu (triedy pre vizuálne zobrazenie pomocou grafov - Tejto triede sa budem venovať neskôr).

metóda set_frequency(self, start='', stop='', center='', span=''):

Najprv nastavíme SA vstupné parametre vďaka funkcii send_cmd a to tak, že pre start použijeme príkaz ':sens1:freq:star ' + start - (teda k nemu rovno "prilepíme" samotnú číselnú hodnotu pre štartovaciu rekvenciu zo vstupnej premennej start). Nastavenie končiacej frekvencie: ':sens1:freq:stop ' + stop , nastavenie stredovej hodnoty pre sweep range (meranie) pomocou príkazu ':sens1:freq:cent ' + center , a nakoniec nastavenia rozpätia pre sweep range vďaka príkazu ':sens1:freq:span ' + span . Nasleduje príkaz '*wai' , teda povieme analyzátoru, aby počkal až, kým sa nevykonajú všetky zadané príkazy a vypýtame si od SA nové štartovacie a koncové hodnoty frekvencií a uložíme si ich do slovníka data :

self.send_cmd(':sens1:freq:start?') , self.frequency['start'] = float(self.read_data())

self.send_cmd(':sens1:freq:stop?'), self.frequency['stop'] = float(self.read_data())

metóda measure(self):

Metóda pre samotné meranie. V prvom kroku si vyprázdnime celý slovník data (vyprázdnime jeho položky ['frequency'],['Z'],['impedance'],['admitance'] a ['phase']). V ďalšom kroku nastavíme režim SA na hold, teda vypnutie kontinuálneho režimu príkazom: ':init1:cont off' . Počkáme kým sa tak udeje (príkaz *wai) a pre aktívnu stopu kanála 1 prečítame formátované dátové polia vo formáte, aký bol predtým nastavený, teda v našom prípade to je ASCI, pomocou príkazu ':calc1:data:fdat?' a tieto dáta ukladáme do slovníka data a budeme ich ďalej spracovávať. Opäť sa spýtame na začiatočné a koncové frekvencie a hodnoty si aktualizujeme v slovníku frequency, počkáme a nastavíme režim SA naspäť do kontinuálneho meracieho režimu ':init1:cont on' .

Nasleduje spracovávanie dát. Najprv ich rozdelíme pomocou metódy split, oddeľovacou čiarkou na pole stringov. vypočítavame hodnotu gamma pre každú rozdelenú hodnotu z dát = complex(float(data[i]), float(data[i+1])) ďalej pridávame jednotlivé frekvencie, získavame hodnoty čísla Z , impedanciu, admitanciu a fázu a všetko si to ukladáme do jednotlivých zložiek slovníka data.

metóda plot_data(self):

Metóda triedy agilent zobrazujúca dáta pomocou Gnuplot u do formy grafu.

metóda save_data(self, fileName=''):

Pokiaľ nie je zadaný názov súboru, do ktorého chceme dáta ukladať, metóda neurobí nič, inak nám uloží dáta do nového súboru, ktorý pomenuje podľa vstupného parametra fileName a to nasledovne : vynechá dva riadky, zapíše # a 140krát pomlčku (oddeľovač), do ďalšieho riadku zapíše - '# frequency\t\t| Z.real\t\t| Z.imag\t\t| |Z|\t\t\t| phase\t\t\t| admitance\n' , nasleduje ďalší oddeľujúci riadok a potom cyklicky zapisujeme ako do tabuľky všetky hodnoty slovníka data v poradí frekvencia, Z - imaginárna, Z - reálna časť, impedancia, fáza a admitancia pekne po riadkoch až do konca a zavrieme súbor - datafile.close().

10.3 Popis validného testovacieho scenára

Meranie sa spúšťa v jednotke(unite) main.py. Vytvoríme objekt a = agilent a p = rlcparams. Nastavíme frekvenciu pomocou metódy objektu a (funkcie) set_frequency s hodnotami pre center = '8000000' a pre span = '500000' . Voláme metódu measure() = meraj. A po vykonaní celého merania zavoláme metódu plot_data() na zobrazenie výsledku merania. Do premennej m si uložíme maximálnu hodnotu admitancie - vytiahneme ju príkazom max(a.data['admitance']) zo slovníka data. Zapamätáme si index maximálnej admitancie do premennej maxIndex a nastavíme novú stredovú (center) frekvenciu na hodnotu frekvencie zo zložky frequency v slovníku data pod indexom maxIndex - maximálnej admitancie : a.set_frequency(center = '%f' % a.data['frequency'][maxIndex]) . Nastavíme aj nové rozpätie merania (span) na hodnotu 2000000 : a.set_frequency(span = '2000000') . Opäť voláme metódu measure() = meraj a zobrazíme výsledné dáta po jej ukončení.

Nasleduje podobný proces, teda nájdeme si znova najväčšiu admitanciu z nového merania a zapamätáme si jej index a prednastavíme stredovú hodnotu merania na hodnotu frekvencie pod týmto indexom. Nastavíme rozpätie na 40 000 Hertzov a v cykle spúšťame opäť meranie pomocou metódy a.measure() . Ak chceme uložíme dáta, inak pokračujeme a voláme metódu data inštancie objektu p (rlcparams) a následne voláme fitovaciu funkciu - taktiež metódu objektu p, v ktorej pomocou zložitých matematických funkcií a pravidiel dostávame optimálny (maximálny) výrez impedancie. Výsledok vypíšeme ako string pomocou príkazu: print str(p.fResonanceFit) + '\t' + str(p.R1) + '\t' + str(p.C0) + '\t' + str(p.C1) + '\t' + str(p.L1).

10.4 Ďalšie možné testovacie scenáre

Chceli by sme taktiež otestovať viaceré možné chybové hlášky a to napríklad chybnou inicializáciou triedy agilent : nastavením iného typu diagramu, ako smith chart, napríklad PHAS - ktorý nastaví formát na fázový formát (':calc1:form phas), alebo REAL - reálny formát (':calc1:form real'). ďalej nastaviť sweeping point cez povolenú hranicu 1601 bodov a aj pod najnižšiu povolenú hranicu 2 body - ':sens1:swe:poin 1800' / ':sens1:swe:poin 1'. Vyskúšať nastaviť iný formát dát ako ASCii pomocou príkazu ':form:data *' na real a real32 (sú iba 3 možné formáty). Ďalej by sme sa chceli zamerať v testoch na nastavenie frekvencií a to presiahnutím rozmedzia pre center aj span (stred a rozpätie) center má rozmedzie od 3E5 do 3E9 (podľa mojich odhadov to znamená 3 na 5 až 3 na 9) TO DO zistiť spresniť, a pre span je to rozmedzie od 0 do 2.9997E9.

Potom by sme sa radi zamerali na série meraní s správnymi hodnotami - v rámci rozmedzia, ale prednastavenými podľa náhodného výberu - vďaka random generatoru a sledovať výsledky a priebeh meraní

11. Testovanie

11.1 Testovanie modulu agilent_SCPI

Bolo to testovanie zamerané na overenie funkčnosti komunikácie zariadenia(Spektrálneho analyzátora) a počítača cez ethernet pripojenie pomocou SICL-LAN protokolu. Nastavili sme novú hostiteľskú ip adresu na 192.168.1.102 a komunikačný port 5025 pričom sme použili spojenie cez socket so vstupnými argumentmi `socket.AF_INET, socket.SOCK_STREAM.`
V triede agilent_SCPI sme testovali metódy sendCmd(): a readData(): na voľných dátach, teda k prístroju nebola pripojená žiadna testovacia vzorka. Obe metódy obstáli testovanie bez problémov, takže základná komunikácia bola vyriešená.

11.2 Testovanie hlavného modulu main

Tu sme testovali funkčnosť príkazov pre prevedenie, kompletného merania na reálnych vzorkách. Bola to hlavná zaťažkávajúca skúška funkčnosti celého softvéru.
Prvotné testovanie na pripravených vyššie uvedených scenároch (kapitola 9.) neobstálo a padalo v priebehu merania v module rlcparams na 119 riadku :
fit_params, fit_error = leastSquaresFit(func, guess, datafit) pre zlý formát vstupných dát do funkcie.
V priebehu procesu ladenia sme dospeli k záveru, že dáta nekompletne spracovávame, teda nečakáme na dokončenie meracieho cyklu a berieme si len useknuté dáta, pri ktorých následnom spracovávaní nám dochádza k posunom k zlým hodnotám, ktoré modul rlcParams nevie spracovať.
Samotná procedúra merania measure():
sa nachádza v triede agilent. Museli sme teda zmeniť prístup v postupe merania v triede agilent.

11.3 Testovanie fungujúceho modulu agilent

Chyba počas predchádzajúcich meraní, bola v nastavení a nesprávnom používaní trigger systému (spúšťacieho systému). Doteraz sme používali prednastavené vnútorný mód (Internal) ktorý fungoval automaticky v kontinuálnom režime a dával nám nesprávne výsledky pre jednotlivé merania. Ako ďalší scenár pripravený na otestovanie funkčnosti chodu aplikácie sme tento trigger systém zmenili z Internal na Manual mód.

V metóde measure sme zmenili meranie z Internal (vnútorného) módu, v ktorom sa merania spúšťali kontinuálne a automaticky na meranie v manuálnom móde volaním príkazu send_cmd(':TRIG:SOUR MAN') v ktorom sa merania spúšťajú manuálne pomocou príkazu send_cmd(':TRIG:SING') a vykonajú sa okamžite po zavolaní tohto príkazu. Vykonávanie príkazu sa ukončuje po dokončení kompletného merania. Pre istotu čakáme na údaje pomocou príkazu send_cmd('*wai') a nakoniec sme si vytiahli namerané údaje príkazom send_cmd(':calc1:data:fdat?') a uložili sme si ich do štruktúry data, kde sme s nimi neskôr pracovali rovnako ako vo vyššie uvedenom skúšobnom scenári v kapitole 9. Po úspešnom meraní a uložení dát prepíname režim naspäť do vnútorného módu a kontinuálneho režimu príkazom send_cmd(':TRIG:SOUR INT').

Meranie sme pomocou ISVOSA sme uskutočnili na reálnej vzorke a po upravení spôsobu merania, nám meranie fungovalo bezproblémovo a dávalo nám správne výsledky fyzikálnych veličín nameraných meraním pre typ materiálu danej vzorky. Nakoľko meranie prebiehalo a vyhodnocovalo zaznamenané výsledky na novom prístroji oveľa rýchlejšie ako na starom type prístroja, pridali sme do programu jednosekundové oneskorenie (pomocou funkcie sleep()), aby mali pracovníci dostatok času na prácu počas merania a zlepšila sa tak aj prehľadnosť celého procesu merania.

12. Záznam z odovzdania a predvedenia diela zadávateľovi

12.1 Plán stretnutia

Kým sa nám podarilo odovzdať poslednú funkčnú verziu aplikácie, museli sme si dohodnúť zopár ďalších stretnutí a to v týchto časoch

- 7.2. 16:00

- 4.2. 18:30

- 28.1. 18:30

- 16.1. 18:30

- 4.1. 16:30

Problém bol v tom, že ak sme si chceli byt istí, že program funguje, mohli sme to otestovať iba priamo na mieste spolu so zadávateľom. Dohadovali sme si ich postupne podľa potreby a zdržali sme sa vždy najdlhšie v danom termíne ako to išlo. Počet stretnutí vzrástol kvôli tomu, že sa nám nedarilo dať do prevádzky základne časti programu a to aj komunikačný protokol.

12.2 Verzie a zmeny

Koncom januára sme opravili komunikačný protokol, ale merania boli z nejakého dôvodu nepresné. Výsledné grafy boli trhané a namiesto krásnej spojitej krivky. Najprv sme zistili, že pri komunikácií posielame príkazy do pristroja v zlom poradí. Nakoniec sa nám to podarilo vyriešiť, ale znovu nám nastala chyba pri vykresľovaní grafov. Zistili sme nakoniec, že dôvodom bolo priskoré meranie na prístroji, takže údaje, čo sme vykresľovali, neboli skutočné namerané dáta. Po zistení tohto problému sme zmenili spôsob spracovania meraní, s ktorým sa nám podarilo nakoniec uskutočniť merania, ktoré odpovedali výsledkom skutočného merania. Nakoniec sme do programu pridali už len vyššiu odozvu, aby sa program ľahšie vypínal.

13. Zhodnotenie

13.1 Objektívne hodnotenie celého týmu

S výslednou prácou sme spokojný. Projekt robí to čo má a spĺňa požiadavky zadávateľa. Na prvých stretnutiach sme si predviedli fungovanie starej verzie softwaru na starom stroji a vysvetlili sme si, čo bude treba spraviť so softwarom, aby fungoval na novšom stroji. Problém nastali pri testovaní aplikácie. Naštudovali sme si príslušnú programátorskú a aj ostatnú dokumentáciu ku stroju a začali sme spisovať scenáre pre funkčnú komunikáciu a spracovanie údajov zo zariadenia. Keď sme mali pripravené testovacie scenáre, tak sme si dohodli stretnutie so zadávateľom projektu, aby sme mohli otestovať naše scenáre. Jeho prítomnosť bola nevyhnutná, aby zabezpečil bezpečný a plynuli chod testovania a správne zachádzanie s prístrojom spolu s testovaným kryštálom. Avšak keď sme zahájili testovanie vyskytlo sa veľa neočakávaných chyb, s ktorými sme nepočítali, pretože ich slabo pokrývala dokumentácia. Toto nás zdržiavalo a museli sme postupne riešiť jeden problém za druhým, pričom sme museli zakaždým si dohodnúť stretnutie so zadávateľom.

V nasledujúcich verziách softwaru by bolo vhodne pridať možnosť používateľom nadstaviť vstupne frekvencie, na ktorých bude prebiehať meranie a možnosť si určiť dĺžku merania. Teraz pristroj meria stále pokiaľ nemá vynútené ukončenie a frekvencie meria v rámci limitov kryštálov, ktoré má fakulta k dispozícii a ktoré sa pri prístroji používajú.

Po prvom stretnutí sme si dohodli plán, ktorý budeme dodržiavať v priebehu semestra. Stanovili sme si ciele a požiadavky od zadávateľa. Samotné požiadavky na výsledok práce sa v čase nemenili, avšak čo sa menilo boli postupne kroky, ktoré sme museli spraviť k dosiahnutiu k finálnej verzie softwaru. Väčšinou to bolo kvôli tomu, že nám naše testovacie scenáre nefungovali. Najväčší problém bol, že sme si nemohli tieto testovacie scenáre skúšať doma.

Založili sme si úložisko, kde budú uložené všetky naše dokumenty a celý projekt so všetkými zápisnicami a neskôr sme si aj založili aj hromadnú online komunikáciu s celým týmom. Tento prístup mal takú výhodu, že nám dovolil komunikovať medzi sebou z rôznych zariadení pomocou internetu. Tu sme si aj rozdeľovali prácu na rovnocenne diely a zo stretnutí sme spisovali zápisnice, ktoré sme dávali na úložisko. Aj napriek tomu, že tento spôsob komunikácie je okamžitý, nie vždy sa nám darilo vyriešiť konkrétny problém v krátkom čase. V niektorých prípadoch by bolo vhodnejšie mať osobné stretnutie namiesto toho elektronického.

13.2 Zhodnotenie Dávida Dobiáša

S výsledkom našej práce som spokojný. Projekt funguje ako má a splnili sme všetko čo od nás staleholder vyžadoval. Problém pri vývoji ktorý komplikoval a predlžoval implementáciu je spôsob akým sme mohli pracovať so zariadením (Spektrálnym analyzátorom). Mohli sme si iba z dokumentácie pripraviť možné scenáre, ktoré nie vždy fungovali. Podľa chovania zariadenia sme potom vykonali ďalšie kroky podobným spôsobom. S takýmto zariadením sme ešte nikdy nerobili (ja

som dokonca nikdy nerobil s laboratórnym prístrojom) a vyvíjať kód na jeden z nich bolo občas ako "streľba na slepo". Čo moc nepreferujem.

Ak by sa mal projekt rozšíriť, tak by som pridal možnosti pre používateľa, ako napríklad možnosť nastavenia vstupných parametrov konkrétne frekvencie.

Ďalší problém, čo sme mali bol problém s komunikáciou so zadávateľom. Veci, ktoré nám povedal na začiatku semestra, mali byť kompletná špecifikácia našej práce, čo napokon nebola, lebo sa stále trocha naša práca menila a neboli sme si nejakú dobu istý, že čo napokon má byť presne našou prácou. Avšak počas decembra sme boli pri cieli celkom blízko, takže tam sa zadávateľ už vedel lepšie vyjadriť a nám to už bolo jasnejšie a v podstate s malými zmenami. Tým smerom sme pokračovali a na každom stretnutí sme boli bližšie a bližšie, pokiaľ sme až nedosiahli úspech.

Z pohľadu týmu a práce v ňom sme si na začiatku boli rovnocenný kolegovia. Avšak už po pár týždňoch som sa ujal pozície vedúceho týmu, lebo náš tým potreboval usmerňovať, lebo jednotlivý členovia boli pasívny v určitých dôležitých veciach. Jedna z nich bola rozdeľovanie si úloh na ďalší týždeň. V podstate mojou úlohou bolo predtým, než si spravím svoju časť úlohy, rozdeliť týždňovú úlohu na 4 približné veľké časti. To sa avšak nedalo vždy, tak niekedy niekto dostal viac roboty, ale zas dostal menej iný týždeň.

Najviac mi však vadila v týme komunikácia. Na sociálnej sieti sme si založili "skupinový chat", kde sme si mohli písať správy, ktoré ostali v histórii komunikácie, takže sa k ním dalo kedykoľvek vrátiť. Niekedy sa členovia týmu dozvedeli dôležitú vec, ktorú chceli ihneď zdieľať so ostatnými, tak toto bol ideálny spôsob, lebo všetci máme účty v tejto sociálnej sieti. Avšak aj keď prebehla komunikácia medzi 2 členmi týmu a vyriešili daný problém, ostatný si to nie vždy prečítali a o konkrétnej udalosti nevedeli. Väčšinu tej komunikácie som zapísal ja a pochybujem, že všetky správy boli prečítane od každého. Preto som sa musel aj niekoľkokrát opakovať, aj keď som to do komunikácie písal. Avšak komunikácia nebola až tak zlá, aby sme ako tým nefungovali. Všetko fungovalo len neinformovanosť nás zdržiavala a v podstate zbytočne. To mi na tom vadilo.

Posledná vec, čo mi prekážala bola dodržiavanie termínov. Samotné termíny boli dodržiavané, ako sme si stanovili, ale bol by som šťastný, kebyže sa práca spraví s pár denným predstihom, aby si aj ostatní vedeli pozrieť, čo dotyčný spravil a povedať mu pripomienky, čo by mal opraviť pred finálnym odovzdaním dokumentu, aby sme sa vyhli prepisovaniu dokumentu.

Ak to mám celkovo zhodnotiť. Práca postupovala približne po rovnakých krokoch každý týždeň s určitými problémami, ktoré sme však zvládli a projekt úspešne dokončili.

13.3 Zhodnotenie Michala Mesároša

Tento projekt mi priniesol mnoho cenných skúseností ohľadom tvorby softvéru na komunikáciu a ovládanie laboratórneho prístroja ako je spektrálny analyzátor, ale najmä skúsenosti s prácou v tíme. Bohužiaľ takmer do Vianoc aj napriek častým stretnutiam so zadávateľom, ani jeden člen nášho tímu poriadne nepochopil jeho predstavu o cieli tohto projektu a tak väčšina našej práce vyšla na zmar. Škoda bola aj že nikto z nášho tímu nevedel pracovať s GITom a tak sme nemali prehľad vo verziách nášho kódu, čo nás veľa krát výrazne zdržalo. Som rád, že sa nám nakoniec

podarilo softvér sfunkčniť, ale moje celkové hodnotenie tohto projektu je negatívne. Snáď nám to poslúži aspoň ako poučenie do budúcna.

13.3 Zhodnotenie Dominika Dobiáša

S konečným produktom som celkom spokojný. Neprebiehalo všetko podľa mojich predstáv, ale nakoniec sa nám po útrapách všetko podarilo. Z počiatku som si myslel, že aké to bude celé jednoduché a v novembri sa už budeme nudiť, avšak nastal úplný opak, že sme si vydýchli až úplne na konci semestra. Problém bol v tom, že sme si nemohli skúšať, čo sme si naštudovali a napísali. Vždy sme sa museli stretnúť a zistiť, že sme niekde spravili chybu a ísť domov a opraviť to. Potom čo nám niečo nešlo nám zadávateľ poradil, čo by sme mali spraviť ďalej a snažili sme sa to nasledovať, ale väčšinou nás čakal neúspech. Prvé stretnutia boli také, že sme len skúšali len časti programu. Tie niekedy išli, niekedy nie. Ale najväčší problém sme mali, keď sme sa snažili spraviť celý projekt funkčný. Museli sme veľa robiť veľa kompromisov, keďže podľa plánu nám skoro nič nevychádzalo. Jeden z dôvodov bol ,že sme si nikdy neboli na sto percent istí, či nám určité veci pôjdu napríklad či nám podarí spraviť komunikačný protokol.

Úlohy sme si rozdeľovali viac menej rovnakým dielom. Práca sa odovzdávala na poslednú chvíľu, ale na čas. Problém bol trochu s komunikáciou, tiež v podstate na poslednú chvíľu. V budúcnosti by som to riešil inak, a to napríklad vždy ústnym podaním a robiť si podrobný transcipt a z toho spraviť zápisnicu. Každopádne s celkovým pohľadom na vec som spokojný.

13.3 Zhodnotenie Mateja Hudeca

S výsledným dielom som spokojný. Splnili sme to čo od nás zadávateľ projektu vyžadoval a odovzdali sme mu funkčný projekt. Počas vývoja síce nastávalo viacero neočakávaných komplikácií, ale s tým sa pri každom vývoji musí počítať. Na tomto predmete sme si ukázali, ako asi môže vyzerať tímová spolupráca a je samozrejmé, že nikdy neprebieha všetko hladko a dokonale a najmä keď sa niečo takéto robí prvý krát, ale aj tak toto všetko považujem ako cennú skúsenosť. Možno by som do osnovy predmetu zapracoval aj nejaké cvičenia správne naučenie sa práce so subversioningom, nakoľko aj naša práca s git repositary nebola asi poňatá úplne správne a síce sa držím zásady, že prax naučí najviac, ale prečo už nemať túto správnu prax trebárs na škole. Tímová komunikácia bola priznávam niekedy náročnejšia a určite aj do veľkej miery aj s mojej strany, ale aj toto sa dá pochopiť, nakoľko má každý z nás množstvo iných aktivít a nedalo sa venovať sto percent času jednému predmetu. Ale vždy sme sa vedeli nakoniec ako tým dohodnúť a podržať jeden druhého, a takto nejako si aj tímovú spoluprácu predstavujem. Celkovú prácu, ktorú sme odviedli a úskalia ktorými sme si prešli, hodnotím ako pozitívny prínos do budúcnosti.