Nástroj pro zpracování a vyhodnocení akustických dat pro potřeby Laboratoře techniky prostředí

Ondrej Mičina

Bakalárska práca
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Ondrej Mičina**
Osobní číslo:	**A18062**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Nástroj pro zpracování a vyhodnocení akustických dat pro potřeby Laboratoře techniky prostředí**
Téma práce anglicky:	**A Post-processing Tool for Acoustic Measurement Data Measurement in the Environmental Engineering Laboratory**

Zásady pro vypracování

1. Seznamte se s postupem akustických měření v laboratoři techniky prostředí a jejich vyhodnocení podle dotčených norem.
2. Prostudujte možnosti exportu z Nor850 a analyzujte strukturu výstupních dat z exportu.
3. Navrhněte strukturu nástroje s ohledem na možnost aktualizace vyhodnocovacích postupů norem včetně uživatelského rozhraní.
4. Implementujte nástroj za použití vhodných technologií.
5. Proveďte zpracování zkušebních dat včetně hodnocení základních akustických parametrů v podobě grafů a nejistot měření, a to vše do podoby protokolu z měření.

Forma zpracování bakalářské práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. ČSN EN ISO 11820. Akustika – Měření tlumičů in situ. Technická norma, 1998.
2. ČSN EN ISO 5136. Akustika – Určování hladin akustického výkonu vyzařovaného do potrubí ventilátory a jinými zařízeními s prouděním vzduchu – Metoda měření v potrubí. Technická norma, 2010.
3. ČSN EN ISO 7235. Akustika – Laboratorní měřicí postupy pro tlumiče hluku v potrubí a vzduchotechnické koncové jednotky – Vložný útlum, vlastní hluk a celková tlaková ztráta. Technická norma, 2010.
4. ČSN EN ISO 3741 – Akustika – Určování hladin akustického výkonu a hladin akustické energie zdrojů hluku pomocí akustického tlaku. Technická norma, 2011.
5. Instruction manual Norsonic Nor850 Software

Vedoucí bakalářské práce: **Ing. Pavel Drábek, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **15. ledna 2021**
Termín odevzdání bakalářské práce: **17. května 2021**

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN

prof. Mgr. Roman Jašek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

1. že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
2. že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Ondrej Mičina, v.r.
podpis studenta

ABSTRAKT

Predmetom mojej bakalárskej práce je návrh a vytvorenie nástroja pre spracovanie a vyhodnotenie akustických dát pre potreby Laboratória techniky prostredia. V teoretickej časti práce popisujem programovací jazyk Python a jeho výhody v oblasti dátovej analýzy, používané knižnice a softvérové nástroje. V praktickej časti popisujem prostredie laboratória, analyzujem problematiku zadanej úlohy a predstavujem programové postupy a riešenia.

Kľúčové slová: akustika, nástroj, návrh nástroja, dáta, spracovanie dát, dátová analýza, programovanie, objektovo orientované programovanie, softvér, Python, Pandas, Excel, Spyder IDE, Qt Designer ,GUI, OOP

ABSTRACT

The subject of this bachelor thesis is the design and creation of a tool for processing and evaluation of acoustic data for the needs of the Laboratory of Environmental Engineering. In the theoretical part of the work I describe the Python programming language and its advantages in the field of data analysis, used libraries and used software tools. In the practical part I describe the environment of the laboratory, analyze the problems of the assigned task and present the solutions including program procedures.

Keywords: acoustic, Post-processing Tool, tool design, data, data analysis, data processing, programming, object - oriented programming, software, Python, Pandas, Excel, Spyder IDE, Qt Designer, GUI, OOP

Moje poďakovanie patrí vedúcemu práce Ing. Pavel Drábek, Ph.D, za to že sa ma ujal a že mi poskytol všetky potrebné materiály a pomáhal s riešením zložitých úloh. Ďakujem aj obom mojim rodičom, ale aj ostatným členom rodiny, ktorí mi držali palce a boli mojou podporou počas písania tejto práce.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Laboratórium techniky prostredia UTB ve Zlíně pracuje s veľkým objemom dát. Mojou úlohou je tieto dáta analyzovať, spracovať a vyhodnotiť pomocou programového nástroja.

V teoretickej časti píšem o programovacom jazyku Python, konkrétne píšem o tom prečo je v dnešnej dobe obľúbený, dávam do popredia jeho technické spôsobilosti a využiteľnosť v rôznych oblastiach približujem objektové programovanie a predstavujem knižnice ktoré využívam. Okrem programovacieho jazyka spomínam aj programové nástroje ako napríklad vývojové prostredie Spyder a jeho výhody, QtDesigner pomocou ktorého aplikáciu dizanujem alebo prostredie Jupyter ktoré funguje vo webovom prehliadači.

V praktickej časti popisujem ako pracovisko laboratória vyzerá, z akých zariadení pozostáva a aké merania vykonáva. Formou obrázkov prezentujem prostredie interiéru a exteriéru laboratória techniky prostredia a poskytujem možnosť nahliadnuť do kalorimetrickej komory v ktorej sa vykonávajú nielen akustické merania. Zoznamujem sa s postupom merania analyzujem dáta a navrhujem riešenia ktoré prenášam do podoby funkcií a objektov.

I. TEORETICKÁ ČÁST

1 PROGRAMOVACÍ JAZYK PYTHON

1.1 Benefity jazyka

Python je moderný programovací jazyk, ktorého popularita stále rastie. Používajú ho napríklad v Google, YouTube, Dropbox, Mozilla, Quora alebo Facebook. Python beží na rôznych platformách, napríklad Windows, Linux, Mac. Je to freeware a tiež open source.

Na rozdiel od mnohých iných jazykov, ktoré sú kompilačné (napríklad Pascal, C/C++, C#) je Python interpreter čiže nevytvára spustiteľný kód.

Hlavným benefitom jazyka Python je napríklad veľmi jednoduchá a dobre čitateľná syntax. Na rozdiel od staticky typovaných jazykov, pri ktorých je treba dopredu deklarovať typy všetkých dát, je Python dynamicky typovaný, čo znamená, že neexistujú žiadne deklarácie, čím urýchľuje prácu programátora. Python obsahuje pokročilé črty moderných programovacích jazykov, napríklad podpora práce s dátovými štruktúrami alebo objektovo-orientovaná tvorba softvéru. Je to univerzálny programovací jazyk, ktorý poskytuje prostriedky na tvorbu moderných aplikácií, takých ako analýza dát, spracovanie médií, sieťové aplikácie a podobne. Python má obrovskú komunitu programátorov a expertov, ktorí tento jazyk obohacujú o užitočné knižnice a sú ochotní pomáhať svojimi radami a skúsenosťami.[1][2]

Obrázok 1. Logo Python

1.2 Technické spôsobilosti jazyka

Programovací jazyk Python je užitočný na vykonávanie úloh v reálnom svete - to je to, čo vývojári robia každý deň. Bežne sa používa v rôznych odvetviach ako nástroj na skriptovanie ďalších komponentov a implementáciu samostatných programov. Faktom je, že možnosti Pythonu ako univerzálneho jazyku sú prakticky neobmedzené: možno ho použiť na všetko, od vývoja webových stránok a hier, až po robotiku a riadenie kozmických lodí. Môžeme povedať, že tieto najbežnejšie úlohy Pythonu spadajú v súčasnosti do niekoľkých širokých kategórií. [3]

1.2.1 Systémové programovanie

Integrované rozhrania Pythonu so službami operačného systému sú ideálne na písanie prenosných a udržiavateľných nástrojov a pomôcok na administráciu alebo správu systému (niekedy nazývaných aj shell tools). Programy Python môžu prehľadávať súbory a adresárové stromy, spúšťať ďalšie programy, vykonávať paralelné spracovanie s procesmi a vláknami atď.

Štandardná knižnica Pythonu pre správu systému je vybavená väzbami POSIX a podporou všetkých obvyklých nástrojov OS: premenné prostredia, súbory, sokety, rúry (pipes), procesy, viacnásobné vlákna, zhoda vzorov regulárneho výrazu, argumenty príkazového riadku, štandardné rozhrania streamu, spúšťače príkazov shell, rozšírenie názvu súborov a ďalšie. Väčšina systémových rozhraní Pythonu je navyše navrhnutá tak, aby boli prenosné, napríklad skript, ktorý kopíruje adresárové stromy, zvyčajne beží nezmenený na všetkých hlavných platformách Pythonu. Za zmienku stojí aj systém Stackless Python, ktorý ponúka pokročilé riešenia požiadaviek na multiprocesing. [3]

1.2.2 Grafické používateľské rozhrania

Vďaka jednoduchosti je Python vhodný aj na programovanie grafického používateľského rozhrania. Python je dodávaný so štandardným objektovo orientovaným rozhraním pre Tk GUI API s názvom Tkinter (Tkinter v 2.6), ktoré umožňuje programom Python implementovať prenosné GUI s natívnym vzhľadom. Grafické používateľské rozhrania Python a Tkinter fungujú rovnako v systémoch Microsoft Windows, X Windows (v systémoch Unix a Linux) a Mac OS (na oboch Classic aj OS X).

Sady nástrojov na vyššej úrovni, ako sú PythonCard a Dabo, sú postavené na základných API, ako sú wxPython alebo spomínaný Tkinter. Vďaka správnej knižnici možno podporu grafického používateľského rozhrania použiť aj v iných súpravách nástrojov v Pythone, napríklad Qt s PyQt, GTK s PyGTK, MFC s PyWin32, .NET s IronPython a Swing s Jython (verzia jazyka Python v jazyku Java) alebo JPype. [3]

1.2.3 Internetové skriptovanie

Python sa dodáva so štandardnými internetovými modulmi, ktoré umožňujú programom Python vykonávať širokú škálu sieťových úloh v režimoch klienta a servera. Skripty môžu komunikovať cez sokety, prenášať súbory pomocou FTP, používať parsovanie, generovať a

analyzovať súbory XML, odosielať, prijímať, komponovať a parsovať e-maily, načítať webové stránky pomocou adries URL, analyzovať HTML a XML načítaných webových stránok, komunikovať cez XML-RPC, SOAP a Telnet a viac. Knižnice Pythonu robia tieto úlohy pozoruhodne jednoduchými. [3]

1.2.4 Databázové programovanie

Pre tradičné požiadavky na databázu existujú rozhrania jazyku Python, pre všetky bežne používané systémy relačných databáz ako napríklad Sybase, Oracle, Informix, ODBC, MySQL, PostgreSQL, SQLite a ďalšie. Svet Pythonu tiež definoval prenosné databázové API pre prístup k databázovým systémom SQL zo skriptov Pythonu, ktoré vyzerá rovnako na rôznych základných databázových systémoch. [3]

1.2.5 Vedecké programovanie a práca s číslami

Rozšírenie numerického programovania použitím knižnice NumPy pre Python ktorá obsahuje také pokročilé nástroje, ako je objekt poľa, rozhrania so štandardnými matematickými knižnicami a oveľa viac. Vďaka integrácii Pythonu s numerickými rutinami kódovanými v kompilovanom jazyku pre rýchlu zmenu, NumPy urobí z Pythonu sofistikovaný, ale ľahko použiteľný numerický programovací nástroj, ktorý často dokáže nahradiť existujúci kód napísaný v tradičných kompilovaných jazykoch ako FORTRAN alebo C ++. [3]

1.3 Objektovo orientované programovanie v Pythone

Objektovo orientované programovanie ponúka iný a často efektívnejší spôsob pohľadu na programovanie, pri ktorom zohľadňujeme kód tak, aby sa minimalizovala redundancia.

Hlavným nástrojom objektovo orientovaného programovania (OOP) v jazyku Python je trieda, angl. class. Triedy umožňujú implementáciu nových druhov objektov, ktoré si definuje programátor.[3]

Trieda v Pythone sa vytvorí kľúčovým slovom `class`, za ktorým nasleduje meno triedy a dvojbodka. V tele triedy sa nachádzajú jej atribúty (premenné) a metódy (funkcie). Prvá metóda `__init__` je tzv. konštruktor. Ten sa vykoná automaticky pri vyvolaní inštancie triedy. Deštruktor je možné vytvoriť metódou s menom `__del__ a` vyvolá sa vždy pri zániku inštancie.

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age
```

Ukážka kódu 1. Definovanie triedy s názvom Person [4]

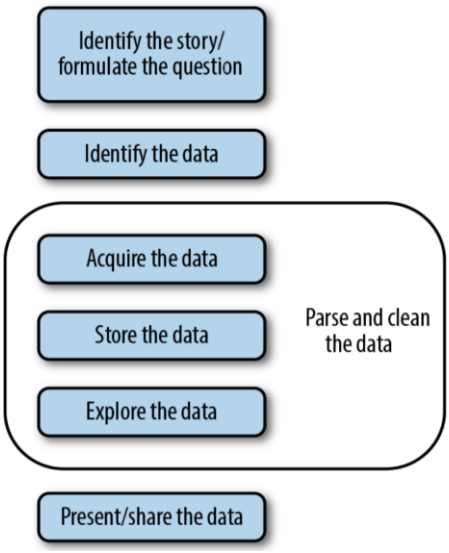
Argument `self` zodpovedá argumentu *this* z C++, pomocou neho pristupujeme k ostatným atribútom a metódam triedy. Meno self nie je záväzné (konvenčné), namiesto neho je možné použiť akýkoľvek iný identifikátor. Pri volaní metódy sa argument self (alebo jeho inak pomenovaný ekvivalent) nesmie uvádzať.[5]

Poskladanie programu z objektov je výhodnejšie a lacnejšie. Ak vieme, že objekty fungujú, sú otestované a udržiavané, môžeme programu veriť. Ak sa niekde vyskytne chyba, stačí ju opraviť na jednom mieste priamo v objekte. [6]

1.4 Analýza dát v Pythone

Pod pojmom analýza dát môžeme nazvať proces, začínajúci formulovaním otázky, ktorú chceme zodpovedať. Dáta vyhodnotené ako vhodné pre použitie zhromaždíme, spracujeme a užitočné informácie uložíme pre ďalšie skúmanie. (Obrázok 2.)[7]

Jedným z hlavných faktorov, prečo sa Python používa na analýzu dát, je jeho vynikajúci ekosystém. Existuje veľa knižníc alebo balíkov zameraných na pracovanie s dátami, vďaka ktorým je proces analýzy údajov veľmi rýchly a pohodlný. Pretože Python je jedným z jednoduchších jazykov a súčasne sa pravidelne vyvíja, je pre dátové vedy preferovaný viac ako iné programovacie jazyky.[8]



Obrázok 2. Proces spracovania údajov [9]

2 POUŽITÉ KNIŽNICE

2.1 Knižnica Pandas

Pandas slúži na prácu s dátami, ktoré možno reprezentovať 2D tabuľkou. Tento tvar dát môžeme nájsť v SQL databázach, súboroch CSV alebo v tabuľkových procesoroch. Základným dátovým typom, ktorý Pandas ponúka je **DataFrame**, čo je vlastne tabuľka v ktorej sú záznamy vo forme riadkov a hodnoty záznamov vo forme stĺpcov. (Obrázok 3.)

Pomocou Pandas vieme dáta čítať, zapisovať, zlučovať, spájať, filtrovať, indexovať a podobne. Tieto funkcie nám výrazne pomáhajú zefektívniť a urýchliť našu prácu.[10] [7]

Pre načítanie dát slúži sada funkcií začínajúca slovom `read_` ktorá získa dáta priamo zo súboru a uloží do spomínaného objektu typu DataFrame.

```
In [2]: actors = pandas.read_csv('static/actors.csv', index_col=None)
        actors
```

Out[2]:

	name	birth	alive
0	Terry	1942	False
1	Michael	1943	True
2	Eric	1943	True
3	Graham	1941	False
4	Terry	1940	True
5	John	1939	True

Obrázok 3. Získanie objektu DataFrame s názvom actors pomocou Pandas [7]

Stĺpce alebo inak *Series*, obsahujú sériu hodnôt prezentovaných ako zoznam. Okrem zoznamu objekt Series obsahuje aj názov stĺpca, dátový typ a index, ktorý jednotlivé hodnoty v zozname pomenováva. Stĺpce vieme zobraziť jednoducho a to zadaním názvu stĺpca do hranatých zátvoriek.

Pre priame vyberanie hodnôt je možné použiť indexery *loc* a *iloc*. Indexer loc vracia dáta riadku z DataFrame na zadanom indexe. Dáta zobrazujú hodnoty stĺpcov daného riadku, usporiadané podľa hlavičky tabuľky.

```
In [29]: actors.loc[2]

Out[29]: name     Eric
         birth    1943
         alive    True
         Name: 2, dtype: object
```

Obrázok 4. Použitie indexeru loc [7]

Indexer iloc funguje obdobne ako loc, rozdielom je že dáta vyberáme priamo podľa pozície poľa v DataFrame.

```
In [38]: actors.iloc[0, 0]

Out[38]: 'Terry'
```

Obrázok 5. Použitie indexeru iloc [7]

2.2 Knižnica Matplotlib

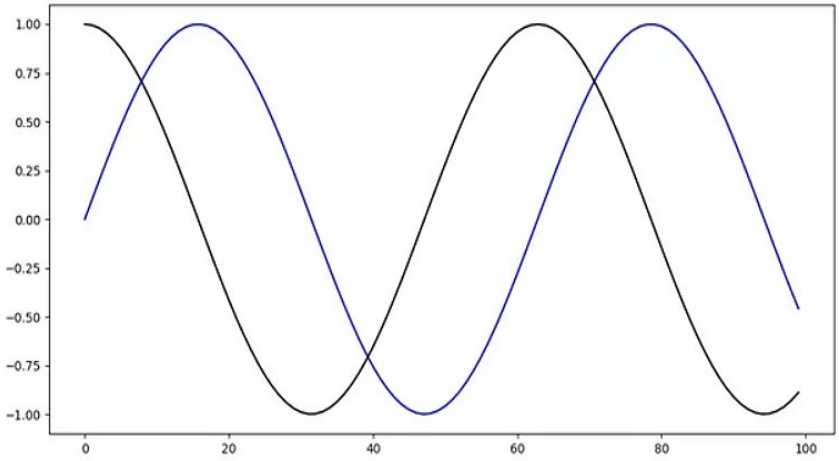
Matplotlib je multiplatformná knižnica na vizualizáciu a grafické vykreslenie údajov. Ponúka schopnú, open source alternatívu k MATLABu. [11]

Matplotlib je multiplatformná knižnica určená pre vizualizácie dát a grafické vykresľovania údajov vo forme rôznych grafov (koláčový, stĺpcový, histogram a pod.) prípadne animácií.

Knižnica ponúka schopnú, open source alternatívu k MATLABu. Je postavená na základoch NumPy a navrhnutá na prácu so širším balíkom SciPy. Jednou z najdôležitejších funkcií Matplotlibu je jeho schopnosť dobre pracovať s mnohými operačnými systémami a grafickými backendmi. Matplotlib podporuje desiatky backendov a typov výstupov, čo znamená, že sa môžeme spoľahnúť, že bude fungovať bez ohľadu na to, aký operačný systém používame alebo aký výstupný formát si prajeme.[11]

```
import matplotlib.pyplot as plt
import numpy as np
fig, ax = plt.subplots(figsize=(12, 6))
x = np.arange(0, 10, 0.1)
y = np.sin(x)
z = np.cos(x)
ax.plot(y, color='blue', label='Sine wave')
ax.plot(z, color='black', label='Cosine wave')
plt.show()
```

Ukážka kódu 2. Vytvorenie grafu pomocou Matplotlib [12]



Obrázok 6. Vytvorený graf pomocou Matplotlib [12]

2.3 Knižnica OS

Knižnica OS v Pythone poskytuje funkcie pre interakciu s operačným systémom. OS spadá pod štandardné obslužné moduly Pythonu. Tento modul poskytuje prenosný spôsob využívania funkcií závislých od operačného systému. Moduly * os * a * os.path * obsahujú mnoho funkcií na interakciu so súborovým systémom.

Knižnica OS možno použiť aj na:

- Vytvorenie adresárov - *os.mkdir()*, *os.makedirs()*
- Výpis zoznamu adresárov - *os.listdir()*
- Zmazanie súborov alebo adresárov - *os.remove()*, *os.rmdir()*
- Otváranie a zatváranie a premenovanie súborov - *os.open(), os.close(), os.rename()*

2.4 Knižnica openpyxl

Openpyxl je knižnica Pythonu na čítanie a zápis súborov programu Excel (s príponou xlsx / xlsm / xltx / xltm).[16] Openpyxl umožňuje:

- Vytváranie nových súborov programu Excel
- Upravovanie už vytvorených súborov
- Čítanie, vytváranie, formátovanie a zlučovanie ľubovoľných buniek
- Vytváranie a editovanie hárkov
- Vytváranie grafov
- Validáciu dát v bunkách [16]

3 POUŽITÉ SOFTVÉROVÉ NÁSTROJE

3.1 Prostredie Spyder

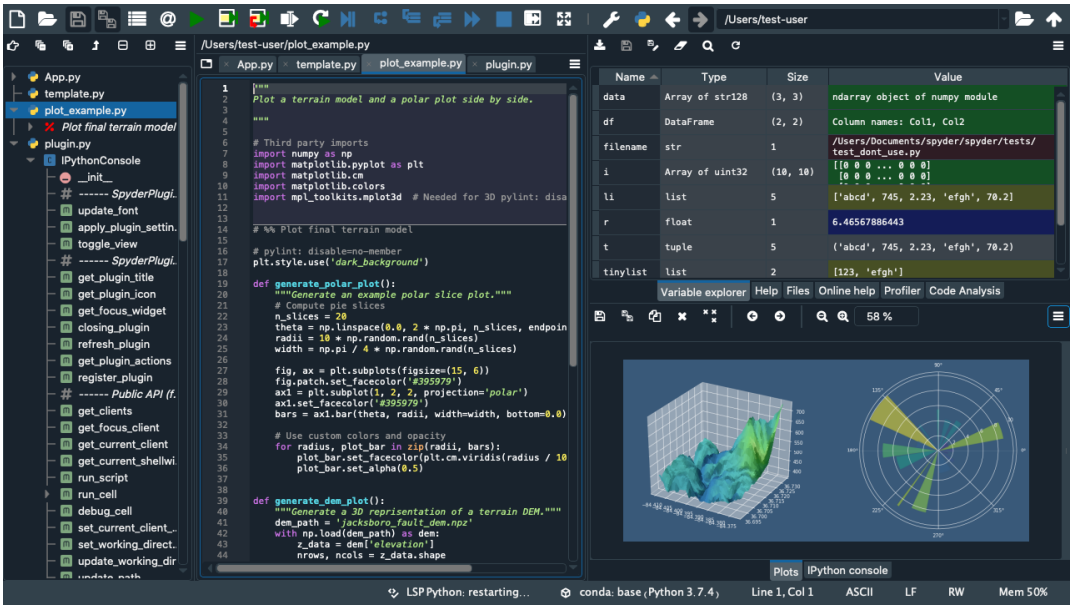
Spyder je open source multiplatformné IDE, ktoré je na rozdiel väčšiny IDE postavené špeciálne pre dátovú vedu. Integruje základné knižnice pre dátovú vedu, ako sú NumPy, SciPy, Matplotlib a IPython, okrem toho možno Spyder rozšíriť o ďalšie pluginy. [14]

Prostredie je súčasťou balíčka Anaconda (32 / 64 bit verzia), ktorý ponúka komplexné prostredie Python spolu so 195 najbežnejšie používanými modulmi Python a konzolou IPython, ktorá je alternatívou k IDLE prostrediu.

Hlavné okno sa skladá z troch častí. V ľavej časti je umiestnené pole na písanie zdrojového kódu. V pravej hornej časti nachádzame Object inspector (správca objektov), Variable explorer (správca premenných) a File explorer (správca súborov). Prostredníctvom týchto troch nástrojov dokážeme kontrolovať vstupy a výstupy zo skriptu. V pravej dolnej časti nachádzame konzolu IPython, ktorá spúšťa skripty napísané v ľavej časti hlavného okna. Program po napísaní spustíme tlačidlom F5. Jednou z výhod tohto prostredia je schopnosť odhaľovať chyby v zdrojovom kóde už počas jeho písania. Teda v prípade, že zadáme nekorektný príkaz, prostredie nás na to upozorní s krátkym časovým oneskorením (žltá výstražná ikona).[15]

Kľúčové vlastnosti:

- Prispôsobiteľné zvýraznenie syntaxe
- Možnosť breakpointov - bodov prerušenia (pre účely ladenia)
- Interaktívne vykonávanie, umožňujúce spustiť jednotlivý riadok, súbor, bunku atď.
- Konfigurácie pre výber pracovných adresárov
- Môže automaticky mazať premenné (alebo zadávať ladenie)
- Schopnosť preskúmať, čo sú funkcie, kľúčové slová a triedy, čo robia a aké informácie obsahujú
- Automatické vkladanie dvojbodky po if, while, atď.
- Riadkové zobrazenie pre grafiku vytvorenú pomocou Matplotlib [16]



Obrázok 7. Vývojové prostredie Spyder [17]

3.2 Nástroj Jupyter

Jupyter je bezplatný interaktívny webový nástroj s otvoreným zdrojovým kódom známy ako výpočtový notebook. Najväčšou výhodou je kombinácia zobrazení, kde v jednom dokumente možno zobraziť:

- softvérový kód,
- výpočtový výstup,
- vysvetľujúci text,
- multimediálne zdroje

Nástroj Jupyter taktiež umožňuje spúšťanie riadkov kódu jednotlivo, čo napomáha programátorovi overiť funkcionalitu daného úseku kódu bez nutnosti prekladania celého programu.

Výpočtové notebooky existujú už celé desaťročia, ale popularita za posledných pár rokov explodovala najmä Jupyter. Tomuto rýchlemu prijatiu pomohla nadšená komunita vývojárov používateľov a prepracovaná architektúra, ktorá umožňuje notebooku ovládať desiatky programovacích jazykov - čo sa odrazilo v jeho názve, ktorý bol podľa spoluzakladateľa Fernanda Péreza inšpirovaný programovacie jazyky Julia (Ju), Python (Py) a R.[18]

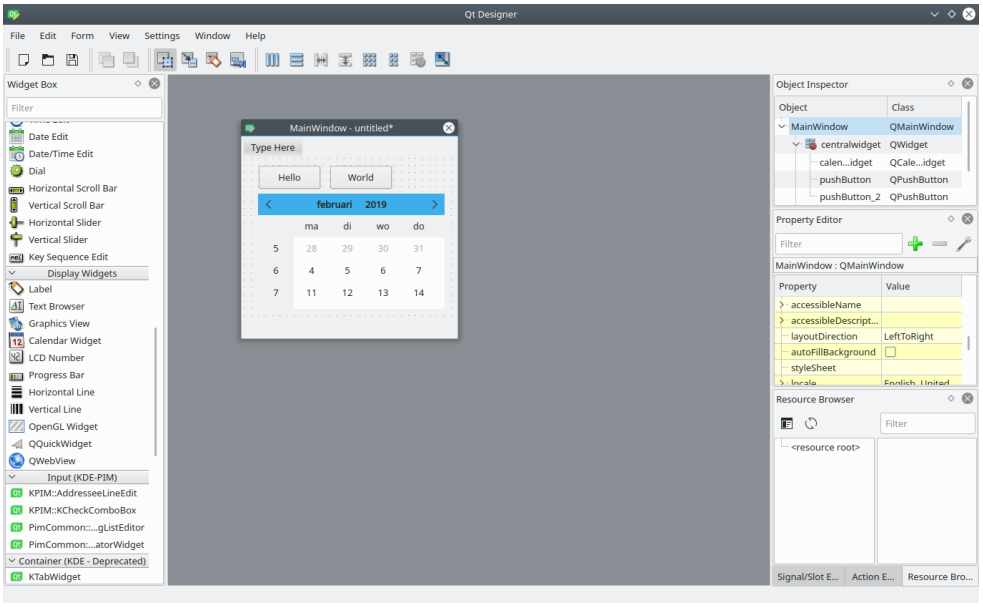
Obrázok 8. Nástroj Jupyter [19]

3.3 Qt Designer

Qt Designer je nástroj Qt na navrhovanie a vytváranie grafických používateľských rozhraní (GUI) s widgetmi Qt. Pomocou tohto nástroja môžeme navrhnúť a prispôsobiť okná aplikácie alebo dialógové okná spôsobom what-you-see-is-what-you-get (WYSIWYG), čo v preklade znamená že úpravy ktoré vykonávame, vidíme v reálnom čase. Nástroj Qt Designer ponúka možnosť testovania pomocou rôznych štýlov a rozlíšení.

Widgety a formuláre vytvorené pomocou Qt Designer sa bezproblémovo integrujú s programovaným kódom vďaka knižnici PyQt5. Prepojenie je funguje pomocou mechanizmov signálov a slotov (rozumej komunikáciou medzi objektmi) Qt, takže môžeme funkcie správania ľahko priradiť ku grafickým prvkom.

Keďže sa vytvorený štýl ukladá vo formáte XML, všetky vlastnosti nastavené v Qt Designer je možné dynamicky meniť v rámci kódu priamo v súbore. Funkcie ako propagácia widgetov a vlastné doplnky vám navyše umožňujú používať svoje vlastné komponenty s Qt Designer. [20]



Obrázok 9. Softvérový nástroj Qt Designer [21]

II. PRAKTICKÁ ČÁST

4 ZOZNÁMENIE SA S PROSTREDÍM LABORATÓRIA

Všetky dáta v tejto práci sú získané z meraní vykonaných v Laboratóriu techniky prostredia na Fakulte aplikovanej informatiky Univerzity Tomáše Bati ve Zlíně. Toto laboratórium pozostáva z univerzálnej kompenzovanej kalorimetrickej komory (Obrázok 13.), trate vzduchotechnických a hydraulických prvkov, meracej a riadiacej kabínky a pomocných priestorov, ktoré slúžia ako úložisko prístrojov alebo ako mechanická dielňa. [22]

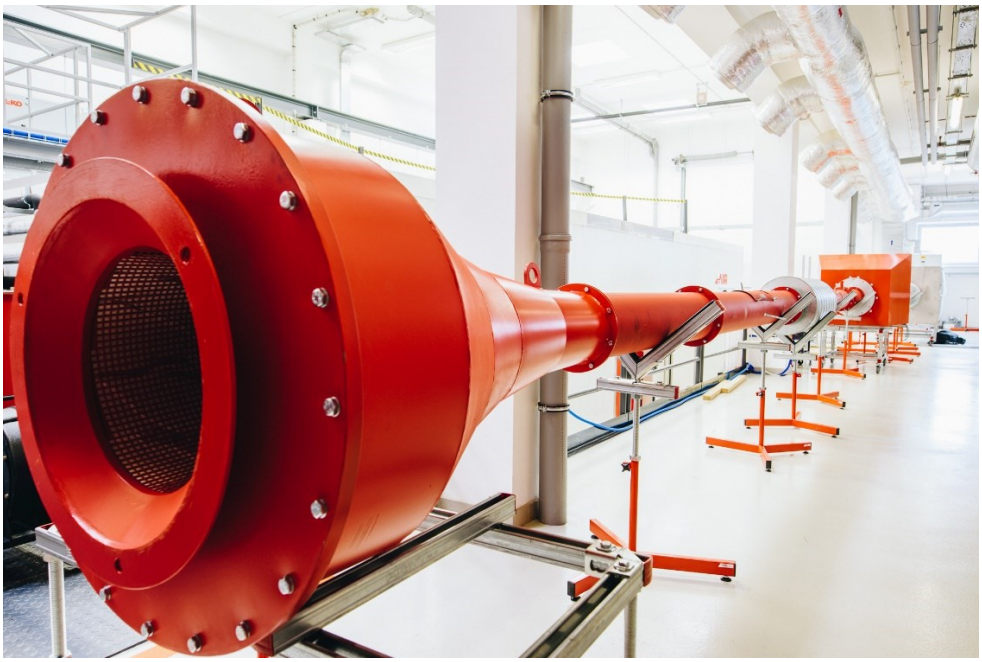
Obrázok 10. Exteriér laboratória

Obrázok 11. Vedúca miestnosť Laboratória techniky prostredia

Pracovisko môže vykonávať merania podľa nasledujúcich štandardov (a mnoho ďalších):

- ČSN EN ISO 3741 Akustika - Určování hladin akustického výkonu a hladin akustické energie zdrojů hluku pomocí akustického tlaku - Přesné metody pro dozvukové místnosti.
- ČSN EN ISO 5135 Akustika - Určování hladin akustického výkonu vyzařovaného do potrubí ventilátory a jinými zařízeními s prouděním vzduchu - Metoda měření v potrubí.
- ČSN EN ISO 11820 - Akustika - Měření tlumičů in situ.
- ČSN EN ISO 3382 - Akustika – Měření parametrů prostorové akustiky (doba dozvuku)
- ČSN EN ISO 7235 - Akustika - Laboratorní měřící postupy pro tlumiče hluku v potrubí a vzduchotechnické koncové jednotky - vložný útlum, vlastní hluk a celková tlaková ztráta.
- ČSN EN 14 511 Klimatizátory vzduchu, jednotky pro chlazení kapalin a tepelná čerpadla s elektricky poháněnými kompresory pro ohřívání a chlazení prostoru.

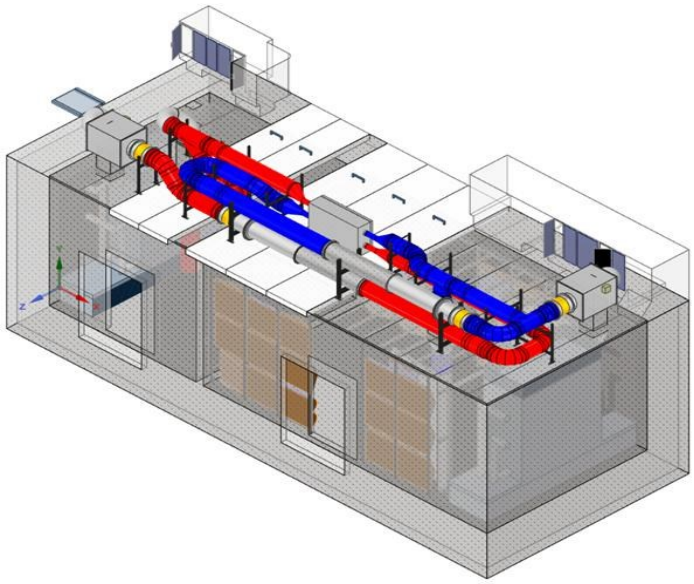
Merania akustických parametrov s laboratórnou triedou presnosťou je možné vykonať v prostredí dozvukovej komory vo frekvenčnom rozsahu 200 – 8 000 Hz alebo v prostredí ventilátorovej trati vo frekvenčnom rozsahu 50 – 16 000 Hz.



Obrázok 12. Ventilátorová trať - meranie podľa ČSN EN ISO 7235 a ČSN EN ISO 11820

4.1 Univerzálna kompenzovaná kalorimetrická komora

Kalorimetrická komora je zariadenie, vďaka ktorému je možné vytvoriť izolované vnútorné prostredie s nastaviteľnými parametrami teploty, vlhkosti a tlaku vzduchu. Snahou je nasimulovať také podmienky, s ktorými sa sledované zariadenia v prevádzke reálne stretávajú.[22]



Obrázok 13. Univerzálna kalorimetrická komora Laboratóriu techniky prostredia FAI UTB

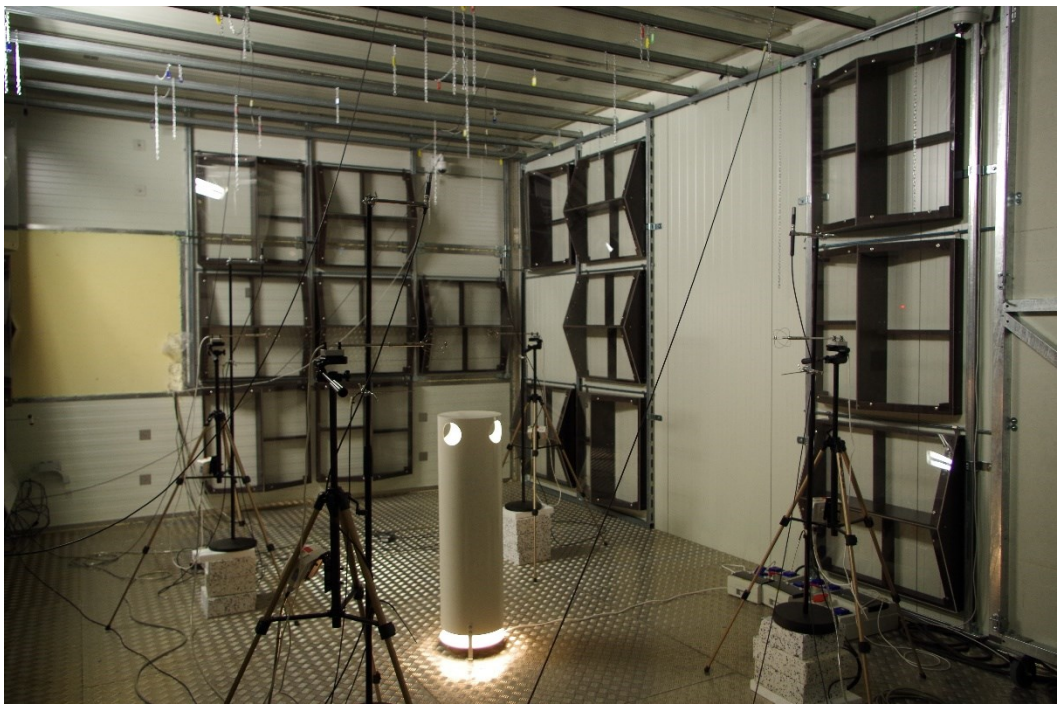
Kompenzovaná kalorimetrická komora pozostáva z dvoch oddelených priestorov s primárnym využitím pre skúšanie energetických a akustických parametrov zariadení ako napr. :

- Chladiacich jednotiek,
- Tepelných čerpadiel typu vzduch/vzduch a vzduch/voda,
- Chladiacich stropov a chladiacich trámcov,
- Vybraných vykurovacích prvkov

Okrem iného je možné testovať chovanie rôznych konštrukčných prvkov pri súčasnom pôsobení rozdielnych mikroklimatických podmienok na tieto prvky, alebo na meranie akustických parametrov. Špecializované prostredie laboratória umožňuje meranie:

- hladiny akustického výkonu strojových zariadení a ventilátorov,
- útlmových charakteristík tlmičov hluku,
- doby dozvuku.

Obrázok 14. Univerzálna kompenzovaná kalorimetrická komora - meranie LW podľa ČSN EN ISO 3741 a ČSN EN ISO 3743 a ČSN EN ISO 354



Obrázok 15. Využitie KK pre iné typy merania

5 PROBLEMATIKA A NÁVRH RIEŠENIA

5.1 Zoznámenie sa s problematikou

Laboratórium techniky prostredia na FAI UTB v Zlíne získa počas merania akustických parametrov veľké množstvo dát ktoré je potrebné spracovať. Vyhodnocuje sa každé oktávové pásmo z meraného rozsahu, každý vykonaný námer, každá séria merania, a to všetko pre každý mikrofón. Laboratórium využíva meraciu aparatúru Nor850-MF od spoločnosti Norsonic, ktorá umožňuje export nameraných dát do tabuľkového procesora. Tieto dáta sú v súčasnej dobe spracovávané pomocou makier v programe MS Excel a priamo vyhodnocovaná do podoby protokolu z merania. Požiadavkou je vytvoriť softvérový nástroj, ktorý proces vyhodnocovania automatizuje a zefektívni vyhodnocovací postup.

5.2 Stanovenie úloh

Po detailnom preštudovaní problému a rozhovore s vedúcim práce boli stanovené kľúčové úlohy:

- Oboznámenie sa s možnosťami exportu dát z meracej aparatúry
- Analyzovanie štruktúry dát z exportu
- Návrh nástroja
- Implementovanie nástroja
- Spracovanie skúšobných dát

5.3 Analýza štruktúry dát

Meracia aparatúra Nor850-MF od spoločnosti Norsonic umožňuje export nameraných dát do tabuľkového procesoru, konkrétne exportujeme súbor typu .xlsx.

Štruktúra dát v súbore je vo forme tabuľky. V hornej časti tabuľky, v hlavičke, sú zaznamenané frekvencie v riadku. Počet frekvencií nie je konštantný, a závisí od nastavenia aparatúry. Z tohto dôvodu je nutné pri každom súbore tento počet vždy zaznamenať.

Od tretieho riadku smerom nadol súbor obsahuje dáta o meraní. V prvom stĺpci naľavo je vždy názov oblasti v ktorej sú umiestnené mikrofóny aparatúry, ďalej smerom vpravo sa nachádza stĺpec s poradím merania a časom kedy bolo vykonané a nakoniec samotné hodnoty hladín akustického tlaku prislúchajúce frekvenciám v hlavičke. Riadky sú združené do sérií meraní pričom v každom z týchto riadkov je zaznamenaný rovnaký názov série. Poradie

merania v sérii vždy začína číslom #1. Každej sérii prislúchajú riadky s hodnotou Std – odchýlka a hodnotou Avg – priemer všetkých hodnôt v stĺpci. Počty sérií a počty meraní v jednej sérii opäť nie sú konštantné a preto ich bude nutné zaznamenávať.

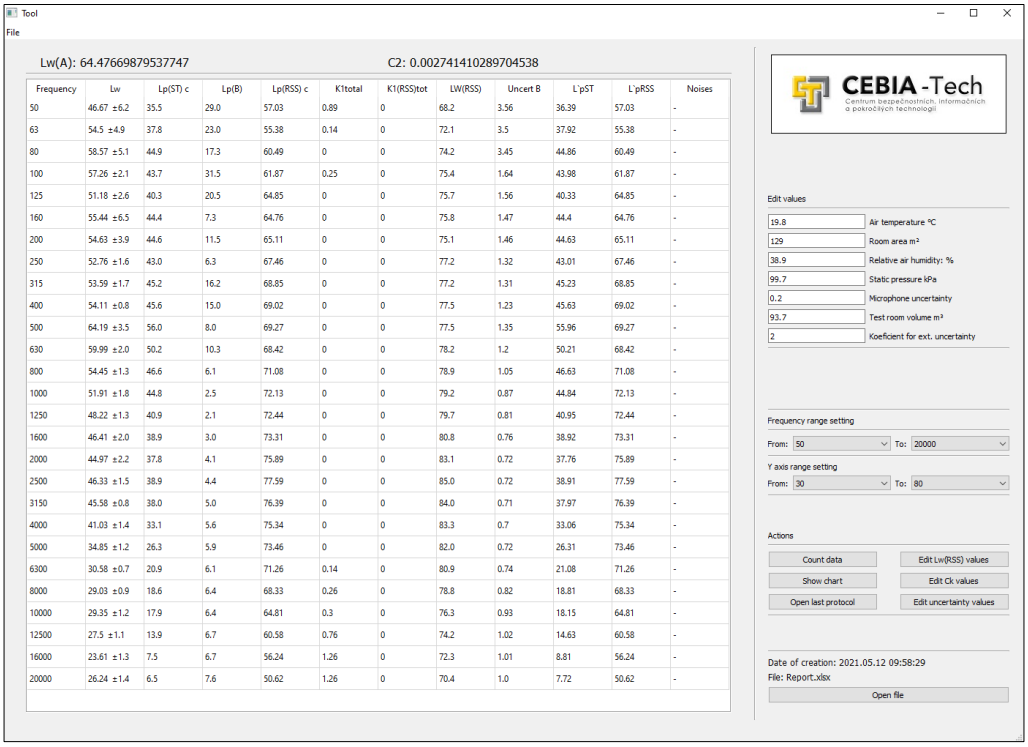
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Frequency				[Hz]	SumA	50	63	80	100	125	160	200	250	315	400	500	630	800	1000	1250	1600
2																						
3	Surface	Total	Average		[dB]	56,1	36,4	37,9	44,9	44	40,3	44,4	44,6	43	45,2	45,6	56	50,2	46,6	44,8	40,9	38,9
4				Std deviation			2,18	1,72	1,81	0,75	0,92	2,29	1,38	0,56	0,6	0,3	1,23	0,71	0,46	0,63	0,45	0,72
5				Avg counter			30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
6	Surface		#1 17:20:28		[dB]	55,6	36,9	34	41,5	45,2	38,7	42,7	44,6	42	45,8	45,4	55,4	49,4	46,8	43,7	40,3	39,7
7				Status																		
8	Surface		#2 17:20:28		[dB]	57,4	35,9	37,2	44,9	44,9	41	48,2	43,3	43	44,8	46,1	58,3	50	46,9	45,4	41,7	38,2
9				Status																		
10	Surface		#3 17:20:28		[dB]	56,1	37,2	38,2	42,5	44,1	40,5	42,6	42,7	42,7	45,5	45,4	55,6	51	46,9	44,7	40,8	39,9
11				Status																		
12	Surface		#4 17:20:28		[dB]	56	37,8	39	44,9	43,4	40,1	43	43,4	43,3	44,2	45,4	55,9	50,4	47,1	44,5	40,8	38,8
13				Status																		
14	Surface		#5 17:20:28		[dB]	55,9	31,4	38,2	46,6	43	41,3	44,5	45,1	43,7	45,7	45,9	55,5	50,9	45,8	44,8	40,8	38
15				Status																		
16	Surface		#6 17:20:28		[dB]	55,2	33,7	38,7	46,1	43,7	40,5	41,2	46,8	43,1	45,4	45,5	54,2	49,2	46,2	45,6	41	38,5
17																						
18	ReferenceSoundSource	Total	Average		[dB]	85,6	57	55,4	60,5	61,9	64,9	64,8	65,1	67,5	68,8	69	69,3	68,4	71,1	72,1	72,4	73,3
19				Std deviation			4,11	3,77	3,29	1,24	1,54	1,93	0,84	1,36	1,29	0,96	0,92	0,88	1,08	0,94	0,55	0,58
20				Avg counter			30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
21	ReferenceSoundSource		#1 15:41:11		[dB]	84,7	51,7	49,7	60,3	61,8	62,2	60,9	64,6	68,1	67,2	67,4	67,6	67,2	68,9	70,2	71,3	72,3
22				Status																		
23	ReferenceSoundSource		#2 15:41:11		[dB]	85,1	49,1	48,7	57,9	62,8	63,3	65,3	63,5	65	67,5	67,8	68,6	67,6	72,1	71,8	72,1	72,7
24				Status																		
25	ReferenceSoundSource		#3 15:41:11		[dB]	86,3	59,9	57,7	55,8	58,8	66,1	64,2	65,1	68,9	68,1	69,9	70,4	68,6	71,4	73,1	72,8	74
26				Status																		
27	ReferenceSoundSource		#4 15:41:11		[dB]	85,7	57,6	53,7	55,4	61,7	64,9	65,1	65,2	68,1	70	69,6	69	67,9	71,8	72,3	72,8	73,6
28				Status																		

Obrázok 16. Štruktúra dát v súbore

5.4 Navrhované riešenie

Po analýze štruktúry dát som sa rozhodol zvoliť jazyk Python. Python sa v súčasnosti hromadne využíva na spracovanie a analýzu dát čomu nasvedčuje aj množstvo dostupných informácií, postupov a návodov k tejto problematike.

Program je obsluhovaný jednoduchým GUI (Obrázok 17.), vytvoreným pomocou nástroja QtDesigner. GUI je rozdelené na dve časti, jednu ktorá slúži na zobrazenie výsledkov a drúhú, ktorá slúži ako obslužná časť. V tejto časti sú polia pre zadanie alebo upravenie hodnôt teploty vzduchu a atmosférického tlaku. Ďalej tlačidlá na načítanie súboru alebo na výpočet výsledných hodnôt, panel s nastavením rozsahu frekvencií, tlačidlo na zobrazenie grafu a tlačidlá na editovanie hodnôt Lw(RSS) a Ck. Vyberanie súborov a uloženie protokolu je realizované pomocou systémového prehliadača súborov. Vypočítané hodnoty sú zobrazené vo forme tabuľky uprostred obrazovky.



Frequency	Lw	Lp(ST) c	Lp(B)	Lp(RSS) c	K1total	K1(RSS)tot	LW(RSS)	Uncert B	L'pST	L'pRSS	Noises
50	46.67 ±6.2	35.5	29.0	57.03	0.89	0	68.2	3.56	36.39	57.03	-
63	54.5 ±4.9	37.8	23.0	55.38	0.14	0	72.1	3.5	37.92	55.38	-
80	58.57 ±5.1	44.9	17.3	60.49	0	0	74.2	3.45	44.86	60.49	-
100	57.26 ±2.1	43.7	31.5	61.87	0.25	0	75.4	1.64	43.98	61.87	-
125	51.18 ±2.6	40.3	20.5	64.85	0	0	75.7	1.56	40.33	64.85	-
160	55.44 ±6.5	44.4	7.3	64.76	0	0	75.8	1.47	44.4	64.76	-
200	54.63 ±3.9	44.6	11.5	65.11	0	0	75.1	1.46	44.63	65.11	-
250	52.76 ±1.6	43.0	6.3	67.46	0	0	77.2	1.32	43.01	67.46	-
315	53.59 ±1.7	45.2	16.2	68.85	0	0	77.2	1.31	45.23	68.85	-
400	54.11 ±0.8	45.6	15.0	69.02	0	0	77.5	1.23	45.63	69.02	-
500	64.19 ±3.5	56.0	8.0	69.27	0	0	77.5	1.35	55.96	69.27	-
630	59.99 ±2.0	50.2	10.3	68.42	0	0	78.2	1.2	50.21	68.42	-
800	54.45 ±1.3	46.6	6.1	71.08	0	0	78.9	1.05	46.63	71.08	-
1000	51.91 ±1.8	44.8	2.5	72.13	0	0	79.2	0.87	44.84	72.13	-
1250	48.22 ±1.3	40.9	2.1	72.44	0	0	79.7	0.81	40.95	72.44	-
1600	46.41 ±2.0	38.9	3.0	73.31	0	0	80.8	0.76	38.92	73.31	-
2000	44.97 ±2.2	37.8	4.1	75.89	0	0	83.1	0.72	37.76	75.89	-
2500	46.33 ±1.5	38.9	4.4	77.59	0	0	85.0	0.72	38.91	77.59	-
3150	45.58 ±0.8	38.0	5.0	76.39	0	0	84.0	0.71	37.97	76.39	-
4000	41.03 ±1.4	33.1	5.6	75.34	0	0	83.3	0.7	33.06	75.34	-
5000	34.85 ±1.2	26.3	5.9	73.46	0	0	82.0	0.72	26.31	73.46	-
6300	30.58 ±0.7	20.9	6.1	71.26	0.14	0	80.9	0.74	21.08	71.26	-
8000	29.03 ±0.9	18.6	6.4	68.33	0.26	0	78.8	0.82	18.81	68.33	-
10000	29.35 ±1.2	17.9	6.4	64.81	0.3	0	76.3	0.93	18.15	64.81	-
12500	27.5 ±1.1	13.9	6.7	60.58	0.76	0	74.2	1.02	14.63	60.58	-
16000	23.61 ±1.3	7.5	6.7	56.24	1.26	0	72.3	1.01	8.81	56.24	-
20000	26.24 ±1.4	6.5	7.6	50.62	1.26	0	70.4	1.0	7.72	50.62	-

Obrázok 17. Predbežný návrh GUI

Pre lepšiu orientáciu a manipuláciu s dátami vytváram objekty. Objekt pre oblasť slúži pre reprezentáciu jednej sekcie v rámci súboru. Oblasti má priradený názov, hodnoty riadkov Sdt a Avg a prislúchajúce hodnoty. Objekt merania, disponuje všetkými oblasťami a pomocou metód na výpočet hodnôt, vytvorených podľa požadovanej normy, môže k dátam z oblastí pristupovať.

Po získaní všetkých dát sa spúšťajú metódy na výpočet výsledkov a vzniká graf a protokol merania s výslednými hodnotami. Výsledky sa taktiež zobrazia v GUI a používateľ si ich môže prehliadnuť v tabuľke. V prípade že používateľ chce protokol uložiť, môže tak urobiť stlačením tlačidla pre uloženie. Vtedy sa protokol presunie alebo prekopíruje na používateľom vybrané miesto.

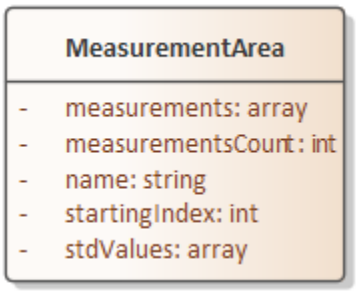
6 NÁVRH OBJEKTOV

Keďže jazyk python poskytuje možnosť použitia objektov, využívam ich na zjednodušenie a zefektívnenie práce. Pomocou objektov viem v programe logicky oddeliť oblasti merania a následne s nimi pracovať.

6.1 Objekt MeasurementArea

Dáta v súbore, s ktorým pracujem, sú rozdelené na rovnaké časti - oblasti. Pre lepšiu orientáciu tieto oblasti oddeľujem a to tak, že pre každú oblasť merania vytváram definovaný objekt typu *MeasurementArea* ktorý zahŕňa premenné:

- *name* - názov série,
- *startingIndex* - počiatočný index, na základe ktorého určujeme z na ktorej pozícii daná oblasť začína
- *measurements* - prislúchajúce dáta získavané z DataFrame,
- hodnoty *stdValues*.



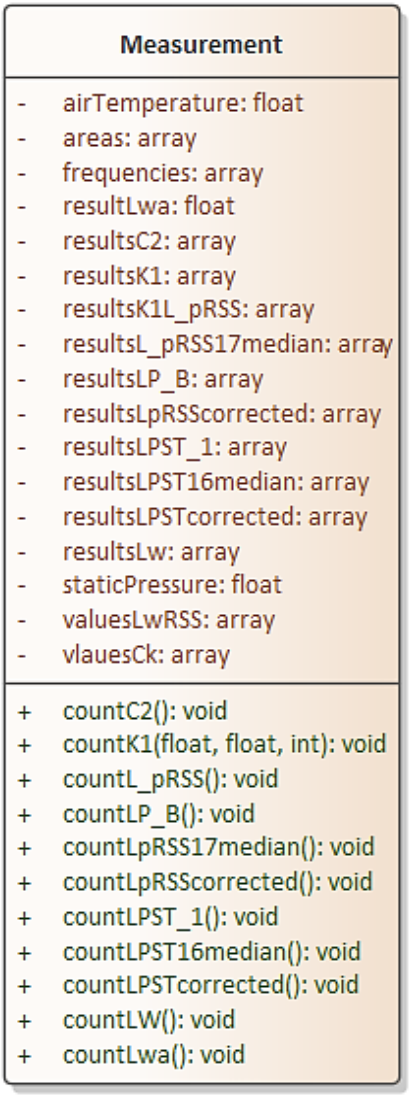
Obrázok 18. Návrh objektu MeasurementArea

Tento objekt funkcie neobsahuje a jeho obsah je vytváraný inými funkciami.

6.2 Objekt Measurement

Všetky výpočty a operácie s dátami sú vykonávané v rámci objektu *Measurement*. Objekt pri vytvorení obsahuje:

- hodnoty zadávané pri vytváraní – *frequencies* (frekvencie), *airTemperature* (teplota vzduchu), *staticPressure* (Barometrický/atmosferický tlak),
- prázdne polia *values* pre hodnoty LwRSS a Ck,
- prázdne pole *areas* ktoré slúži na ukladanie objektov typu *MeasurementArea* (6.1),
- prázdne premenné výsledkov, ktorých názov začína slovom *result*,
- funkcie na výpočet požadovaných výsledkov podľa normy ČSN EN ISO 3741



Obrázok 19. Predbežný návrh objektu Measurement

Objekt *Measurement* reprezentuje meranie ako celok. Funkcie v tomto objekte slúžia na výpočet požadovaných výsledkov. Každej výpočtovej operácii prislúcha funkcia ktorá používa hodnoty uložené v premenných. Tieto hodnoty sú následne spracované podľa normy [23] a výsledky sú uložené do premenných so začiatočným názvom *result*. Výhodou tohto spôsobu ukladania výsledkov spočíva v tom, že výsledky sú združené v rámci objektu a v prípade že sa bude vykonávať viac meraní, funkcie budú pristupovať len k hodnotám daného merania.

6.2.1 Funkcia countC2

Podľa normy [23] sa hodnota korekcie na vyžarovaciu impedanciu C_2 v decibeloch sa vypočíta podľa rovnice (1)

$$C_2 = -10\lg\frac{p_s}{p_{s0}}dB + 15\lg\left(\frac{273.15+\theta}{\theta_1}\right)dB \quad (1)$$

kde

p_s je statický tlak v skúšobnej miestnosti v dobe skúšky, v kilopascaloch

p_{s0} referenčný statický tlak, 101,325 kPa

θ teplota vzduchu v skúšobnej miestnosti v dobe skúšky, v stupňoch Celzia

θ_1 = 296 K

Výpočet je realizovaný pomocou funkcie *countC2*.

```
def countC2(self):
    self.resultsC2=(-10*math.log(self.staticPressure/101.325,10))
      +15*math.log((273.15+self.airTemperature)/296,10)
    return self.resultsC2
```

Ukážka kódu 3. Funkcia countC2

6.2.2 Funkcia countLPST_1

Podľa normy [23] sa hladina nameraného časovo priemerovaného akustického tlaku v každom tretinooktávovom pásme sledovaného frekvenčného rozsahu a pre každú i-tú polohu alebo dráhu pohybu mikrofónu a zvolený prevádzkový režim skúšaného zdroja priemerovaná cez polohy zdroja $L'_{pi(ST)}$ sa vypočíta podľa rovnice(2):

$$L'_{pi(ST)} = 10\lg\left\{\frac{1}{N_s}\sum_{j=1}^{N_s} 10^{-0.1\left[L'_{pi(ST)}\right]_j}\right\}dB \quad (2)$$

kde

$\left[L'_{pi(ST)}\right]_j$ je nameraná (nekorigovaná) hladina časovo priemerovaného akustického tlaku v tretinooktávovom pásme v i-tej polohe mikrofónu alebo pri i-tú dráhu posunu mikrofónu a pre j-tú polohu zdroja za prevádzky skúšaného zdroja hluku (ST), v decibeloch;

N_s počet polôh zdroja.

Navrhnutá funkcia *countLPST_1* vyhľadá oblasť s názvom *Surface,* z ktorej získa všetky hodnoty $\left[L'_{pi(ST)}\right]_j$. Následne sa počítajú hodnoty $L'_{pi(ST)}$ pre všetky tretinooktávové pásma a uložia sa do poľa s názvom *resultsLPST_1*.

```
def countLPST_1(self):
    partialResults=[]

    for area in self.areas:
        if "Surface" in area.name:
            for frequencyCollumnNumber in range(len(self.frequencies)):
                rowsData=[]
                for dataRow in area.measurements:
                    rowsData.append(
                         pow(10,(0.1*dataRow[frequencyCollumnNumber]))

                partialResults.append(
                  10*math.log(sum(rowsData)*(1/area.measurementsCount),10))

    self.resultsLPST_1=partialResults
    return self.resultsLPST_1
```

Ukážka kódu 4. Funkcia countLPST_1

6.2.3 Funkcia countLPB

Navrhnutá funkcia *countLB_P* slúži na výpočet hodnôt $L_{pi(B)}$, potrebných pre výpočet korigovaných hodnôt (kap. 6.2.5) na pozadie. Hodnoty sa počítajú podľa rovnice (3).

Princíp funkcie je totožný s funkciou *countLPST_1* (kap. 6.2.2) s rozdielom že dáta sú získavané z oblasti s názvom *Backgroud*.

```
def countLP_B(self):
    LPBlist=[]

    for area in self.areas:
        if "Background" in area.name:
            for frequencyCollumnNumber in range(len(self.frequencies)):
                arrayToSum=[]
                for dataRow in area.measurements:
                    arrayToSum.append(
                         pow(10,(0.1*dataRow[frequencyCollumnNumber])))

                collumnSum = sum(arrayToSum)
                LPBlist.append(
                  10*math.log(((1/area.measurementsCount) *collumnSum),10))

    self.resultsLP_B = LPBlist
    return self.resultsLP_B
```

Ukážka kódu 5. Funkcia countLP_B

6.2.4 Funkcia countL_pRSS

Navrhnutá funkcia *countL_pRSS* slúži na výpočet hodnôt $L'_{pi(RSS)}$. Princíp funkcie je totožný s funkciou *countLPST_1* (kap. 6.2.2) s rozdielom že dáta sa získavajú z oblasti s názvom *Reference*.

```
def countL_pRSS(self):
    partialResults=[]

    for area in self.areas:
        if "Reference" in area.name:
            for frequencyCollumnNumber in range(len(self.frequencies)):
                rowsData=[]
                for dataRow in area.measurements:
                    rowsData.append(
                         pow(10,(0.1*dataRow[frequencyCollumnNumber])))

                partialResults.append(
                  math.log(sum(rowsData)*(1/area.measurementsCount),10)*10)

    self.resultsL_pRSS=partialResults
    return self.resultsL_pRSS
```

Ukážka kódu 6. Funkcia countL_pRSS

6.2.5 Funkcia countK1

Podľa normy [23] sa korekcia na hluk pozadia K_1 v i - tej polohe mikrofónu alebo pre i -tú dráhu posunu mikrofónu pre každé tretinooktávové pásmo sa vypočíta podľa rovnice(4):

$$K_{1i} = -10\lg\left(1 - 10^{0.1\Delta L_{pi}}\right) dB \qquad (4)$$

kde

$\Delta L_{pi} = L'_{pi(ST)} - L_{pi(B)}$

kde

$L'_{pi(ST)}$ je nameraná (nekorigovaná) hladina časovo priemerovaného akustického tlaku v tretinooktávovom pásme pre i-tú polohe mikrofónu alebo pre i-tú dráhu pohybu mikrofónu za prevádzky skúšaného zdroja hluku, v decibeloch;

$L_{pi(B)}$ hladina časovo priemerovaného akustického tlaku hluku pozadia (B) v tretinooktávovom pásme, nameraná v i-tej polohe mikrofónu alebo pre i-tú dráhu posunu mikrofónu, v decibeloch.

Pokiaľ je $\Delta L_{pi} \geq 15$ dB, K_{1i}, sa považuje za nulovú a žiadna korekcia na hluk pozadia sa neuplatňuje.

Pokiaľ je6 dB $\leq \Delta L_{pi} < 5$ dB, pre tretinooktávové pásma so strednými frekvenciami 200 Hz a nižšími a 6300 Hz a vyššími, K1, sa vypočíta podľa rovnice (4).

Pokiaľ je 10dB $\leq \Delta L_{pi} < 15$ dB, pre tretinooktávové pásma so strednými frekvenciami 250 Hz až 5000 Hz, K_{1i} sa vypočíta rovnice (4).

Pokiaľ je $\Delta L_{pi} < 6$ dB pre jedno alebo viac tretinooktávových pásiem so strednými frekvenciami 200 Hz a nižšími a 6300 Hz a vyššími, potom K_{1i} = 1,26 dB (hodnota pre $\Delta L_{pi} = 6$ dB). Pokiaľ je $\Delta L_{pi} < 10$ dB pre jedno alebo viac tretinooktávových pásiem so strednými frekvenciami 250 Hz až 5000 Hz, potomK_{1i} = 0,46 dB (hodnota pre $\Delta L_{pi} = 10$ dB). V týchto prípadoch sa musí jasne uviesť v protokole o meraní rovnako ako v grafoch v tabuľkách výsledkov, že dáta v týchto pásmach predstavujú hornú hranicu hladiny akustického výkonu skúšobného zdroja hluku.

Funkcia `countK1` vypočíta hodnotu ΔL_{pi} pomocou hodnôt získaných z parametrov *L_piST* a *LpiB*. Následne sa prechádza všetkými podmienkami a v prípade jej splnenia je vykonaný príslušný kód ktorý vráti hodnotu K_{1i}.

```
def countK1(self,L_piST,LpiB,frequency):
    deltaLpi = L_piST - LpiB

    if deltaLpi>=15:
        return 0

    elif (6 <= deltaLpi < 15) and (200 >= frequency or frequency >= 6300):
        bracket=1-pow(10,(-0.1*deltaLpi))
        return (-10*math.log(bracket,10))

    elif (10 <= deltaLpi) < 15 and (250 <= frequency <= 5000):
        bracket=1-pow(10,(-0.1*deltaLpi))
        print("Bracket",bracket,"Frequency",frequency)
        return (-10*math.log(bracket,10))

    elif (deltaLpi < 6) and (200 >= frequency or frequency >= 6300):
        return (1.26)

    elif (deltaLpi < 10) and (250 <= frequency <= 5000):
        return (0.46)

    else :
        return ("error")
```

Ukážka kódu 7. Funkcia countK1

6.2.6 Funkcia countLPSTcorrected

Podľa normy [23] sa musia všetky namerané hladiny časovo priemerovaného akustického tlaku korigovať na prítomnosť hluku pozadia. Korigovaná kladina časovo priemerovaného akustického tlaku v tretinooktávovom pásme $L_{pi(ST)}$, v decibeloch v i-tej polohe mikrofónu alebo pre i-tú dráhu posunu mikrofónu, za prevádzky skúšobného zdroja hluku je:

$$L_{pi(ST)} = L'_{pi(ST)} - K_{1i} \tag{5}$$

kde

$L'_{pi(ST)}$ je nameraná (nekorigovaná) hladina časovo priemerovaného akustického tlaku v tretinooktávovom pásme v i-tej polohe mikrofónu alebo i- tú dráhu posunu mikrofónu za prevádzky skúšobného zdroja hluku, v decibeloch;

K_{1i} korekcia na hluk pozadia, v decibeloch.

Funkcia *countLPSTcorrected* spustí funkcie *countLPST_1* a *countLPB*. Potom ako získa potrebné dáta pre výpočty, v cykle for vypočíta hodnoty K_{1i} (kap. 6.2.5) a hodnoty $L_{pi(ST)}$ pre všetky tretinooktávové pásma. Výsledky hodnôt K_{1i} sa uložia do poľa *resultsK1* a výsledky hodnôt $L_{pi(ST)}$ sa uložia do poľa *resultsLPSTcorrected*.

```
def countLPSTcorrected(self):
    LPSTreturn = []

    self.countLPST_1()
    self.countLPB()

    k1Array=[]
    for i in range(len(self.frequencies)):
        k1Value=self.countK1(
            self.resultsLPST_1[i],self.resultsLPB[i],self.frequencies[i])
        k1Array.append(k1Value)
        LPSTreturn.append(self.resultsLPST_1[i]-k1Value)

    self.resultsLPSTcorrected = LPSTreturn
    self.resultsK1=k1Array
    return self.resultsLPSTcorrected
```

Ukážka kódu 8. funkcia LPSTcorrected

V nasledujúcich tabuľkách sú zobrazené vypočítané hodnoty K_{1i} a $L_{pi(ST)}$ pomocou navrhovanej funkcie *countLPSTcorrected* pri vsupných hodnotách $L'_{pi(ST)}$ a $L_{pi(B)}$.

f	2500	2500	2500	2500	2500	2500	2500	2500	2500
$L'_{pi(ST)}$	38,91	37,97	33,06	26,31	21,08	18,81	18,15	14,63	8,81
$L_{pi(B)}$	4,40	4,95	5,55	5,86	6,14	6,38	6,40	6,68	6,71

Tabuľka 1. Vstupné hodnoty pre funkciu countLPSTcorrected

f	2500	2500	2500	2500	2500	2500	2500	2500	2500
K_{1i}	0,00	0,00	0,00	0,00	0,14	0,26	0,30	0,76	1,26
$L_{pi(ST)}$	38,91	37,97	33,06	26,31	20,94	18,56	17,85	13,87	7,55

Tabuľka 2. Vypočítané hodnoty pomocou funkcie countLPSTcorrected

6.2.7 Funkcia countLpRSScorrected

Funkcia *countLpRSScorrected* slúži na výpočet hodnôt $L_{pi(RSS)}$. Princíp funkcie je totožný s funkciou *countLPSTcorrected* (kap. 6.2.6) s rozdielom že v rovnici na výpočet korekcie (5) je $L'_{pi(ST)}$ nahradené $L'_{pi(RSS)}$, vypočítané funkciou *countL_pRSS* (kap. 6.2.4). Výsledné hodnoty sa uložia do poľa s názvom *resultsL_pRSScorrected*.

```
def countLpRSScorrected(self):
    L_pRSScorrectedReturn = []

    self.countL_pRSS()
    self.countLPB()

    k1L_pRSS17Array=[]
    for i in range(len(self.frequencies)):
        k1value=self.countK1(
            self.resultsL_pRSS[i],self.resultsLPB[i],self.frequencies[i])
        k1L_pRSS17Array.append(k1value)
        L_pRSScorrectedReturn.append(self.resultsL_pRSS[i]-k1value)

    self.resultsLpRSScorrected = L_pRSScorrectedReturn
    self.resultsK1L_pRSS =k1L_pRSS17Array
    return self.resultsLpRSScorrected
```

6.2.8 Funkcia countLPST16median

Podľa normy [23] sa hladina strednej hodnoty časovo priemerovaného akustického tlaku v tretinooktávovom pásme korigovaná na hluk pozadia v skúšobnej miestnosti za prevádzky skúšobného zdroja hluku $L_{pi(ST)}$ sa vypočíta podľa rovnice(6):

$$\overline{L_{p(ST)}} = 10\lg\left[\frac{1}{N_M}\sum_{i=1}^{N_M} 10^{0.1L_{pi(ST)}}\right] dB \qquad (6)$$

kde

$L_{pi(ST)}$ je korigovaná hladina časovo priemerovaného akustického tlaku v tretinooktávovom pásme v i-tej polohe mikrofónu alebo pre i-tú dráhu posunu mikrofónu za prevádzky skúšaného zdroja, v decibeloch;

N_M počet polôh mikrofónu alebo jednotlivých dráh posunu mikrofónu.

Funkcia *countLPST16median* vypočíta hodnotu $\overline{L_{p(ST)}}$ pre každé tretinooktávové pásmo a výsledné hodnoty uloží do poľa *resultsLPST16median.*

```
def countLPST16median(self):
    LPST_16medianArray=[]

    for i in range(len(self.frequencies)):
        bracket=pow(10, (0.1*self.resultsLPSTcorrected[i]))
        resultTemp=10*math.log(bracket,10)
        LPST_16medianArray.append(resultTemp);

    self.resultsLPST16median = LPST_16medianArray
    return self.resultsLPST16median
```

Ukážka kódu 9. Funkcia countLPST_16

6.2.9 Funkcia countLpRSS17median

Podľa normy [23] sa hladina strednej hodnoty časovo priemerovaného akustického tlaku v tretinooktávovom pásme korigovaná na hluk pozadia zdroja zvuku (RSS) v skúšobnej miestnosti $\overline{L_{p(RSS)}}$ sa vypočíta podľa rovnice (7):

$$\overline{L_{p(RSS)}} = 10\lg\left[\frac{1}{N_M}\sum_{i=1}^{N_M} 10^{0.1L_{pi(RSS)}}\right] dB \qquad (7)$$

kde

$L_{pi(RSS)} = L'_{pi(RSS)} - K_{1i(RSS)}$

kde

$L'_{pi(RSS)}$ je nameraná (nekorigovaná) hladina časovo priemerovaného akustického tlaku referenčného zdroja zvuku v tretinooktávovom pásme v i-tej polohe mikrofónu alebo pre i-tú dráhu posunu mikrofónu, v decibeloch;

$K_{1i(RSS)}$ korekcia na hluk pozadia pre referenčný zdroj zvuku v i -tej polohe mikrofónu alebo pre i -tú dráhu posunu mikrofónu vypočítaná podľa rovnice (*korekcie), kde $L'_{pi(ST)}$ je nahradené$L'_{pi(RSS)}$, v decibeloch;

N_M počet polôh mikrofónu alebo jednotlivých dráh posunu mikrofónu.

Funkcia *countLpRSS17median* vypočíta hodnoty $\overline{L_{p(RSS)}}$ pre každé tretinooktávové pásmo podľa rovnice (7) a výsledné hodnoty uloží do poľa *resultsLpRSS17median*.

```
def countLpRSS17median(self):
    LpRSSmedianList=[]

    for i in range(len(self.frequencies)):
        bracket=pow(10, (0.1*self.resultsLpRSScorrected[i]))
        resultTemp=10*math.log(bracket,10)
        LpRSSmedianList.append(resultTemp)

    self.resultsLpRSS17median = LpRSSmedianList
    return self.resultsLpRSS17median
```

Ukážka kódu 10. Funkcia countLpRSS17median

6.2.10 Funkcia countLW

Podľa normy [23] sa hladina akustického výkonu skúšobného zdroja hluku v každom tretinooktávovom pásme pri referenčných meteorologických podmienok sa vypočíta podľa rovnice (8):

$$L_W = L_{W(RSS)} + \left(\overline{L_{p(ST)}} - \overline{L_{p(RSS)}}\right) + C_2 \tag{8}$$

kde

$L_{W(RSS)}$ je hladina akustického výkonu referenčného zdroja zvuku v tretinooktávovom pásme určená podľa ISO 6926 a korigovaná na meteorologické podmienky v dobe skúšky, v decibeloch;

$\overline{L_{p(ST)}}$ korigovaná hladina strednej hodnoty časovo priemerovaného akustického tlaku v tretinooktávovom pásme v skúšobnej miestnosti za prevádzky skúšobného zdroja hluku, v decibeloch;

$\overline{L_{p(RSS)}}$ hladina strednej hodnoty časovo priemerovaného akustického tlaku referenčného zdroja zvuku v tretinooktávovom pásme korigovaná na hluk pozadia v skúšobnej miestnosti, v decibeloch;

C_2 korekcia na vyžarovaciu impedanciu, v decibeloch, vypočítaná podľa rovnice (1)

Funkcia *countLW* vypočíta hodnoty L_W podľa rovnice (8) pre všetky tretinooktávové pásma a uloží ich do poľa *resultsLw*.

```
def countLW(self):
    LwValues=[]
    for i in range(len(self.frequencies)):
        result=self.valuesLwRSS[i]+(self.resultsLPST_16[i] -
                  self.resultsL_pRSS17[i]) + self.resultsC2
        LwValues.append(result)
    self.resultsLw= LwValues
    return self.resultsLw
```

Ukážka kódu 11. Funkcia countLw

6.2.11 Funkcia countLwa

Podľa normy [23] sa hladina akustického výkonu L_{WA} vypočíta podľa rovnice (9):

$$L_{WA} = 10\lg \sum_{k=k_{min}}^{k_{max}} 10^{0.1(L_{Wk}+C_k)} \text{ dB} \tag{9}$$

kde

L_{Wk} je hladina akustick-ho výkonu v k-tom tretinooktávovom pásme, v decibeloch

k, C_k hodnoty uvedené v tabuľke normy [23] alebo udávané výrobcom,

k_{min}, k_{max} hodnoty k odpovedajúce najnižšiemu a najvyššiemu meranému tretinooktávovému pásmu.

Na výpočet hodnoty L_{WA} je navhrnutá funkcia *countLwa*.

```
def countLwa(self):
    arrayToSum=[]

    for fColNum in range(len(self.frequencies)):
        result=math.pow(
            10,0.1*(self.resultsLw[fColNum]+self.valuesCk[fColNum]))
        arrayToSum.append(result)
    resultValue=10*math.log(sum(arrayToSum),10)

    self.resultLwa=resultValue
    return self.resultLwa
```

Ukážka kódu 12. Funkcia countLwa

6.2.12 Funkcia countUncertaintyValues

Funkcia *countUncertaintyValues* vypočíta hodnoty neistoty pre všetky frekvenicie. Na začiatku sú definované prázdne polia a pomocou funkcie for sa vykonajú výpočty podľa normy [23]. Vvýsledky sú uložené v poliach s počiatočným názvom results.

```
def countUncertaintyValues(self,mic_uncertainty,reverberation_time_array,ro
om_volume,room_area,koeficient):

    uncertaintyB_array=[]
    uncertaintyCombined_array=[]
    uncertaintyExtended_array=[]

    for frequencyCollumnNumber in range(len(self.frequencies)):
        if self.frequencies[frequencyCollumnNumber] < 100:
            method_uncertainty=3
        else:
            method_uncertainty=0.3

        part1= math.pow(method_uncertainty, 2)
            + math.pow(mic_uncertainty,2) + math.pow(0.002 , 2)
        part2 = math.sqrt(
            ((2.42 * reverberation_time_array[frequencyCollumnNumber])/
                  self.frequencies[frequencyCollumnNumber]) +((0.2 * 0.2)/
                  120))
        part3 = 20.05*math.sqrt(273+self.airTemperature)
        part4 = (240 * room_volume) / (0.9 * 0.9 * room_area * part3)
        part5 = math.pow(((4.3 /
                  reverberation_time_array[frequencyCollumnNumber])-
                        part4), 2)

        uncertaintyB_array.append(math.sqrt(part1+ part2 * part5 +
            math.pow(0.02,2) + math.pow(0.05,2)))

        uncertaintyCombined_array.append(math.sqrt(math.pow(
            self.areas[0].stdValues[frequencyCollumnNumber], 2)+math.pow(
                  self.areas[0].stdValues[frequencyCollumnNumber], 2)))

        uncertaintyExtended_array.append(
            koeficient*uncertaintyCombined_array[frequencyCollumnNumber])

    self.resultsUncertaintyB=uncertaintyB_array
    self.resultsUncertaintyCombined=uncertaintyCombined_array
    self.resultsUncertaintyExtended=uncertaintyExtended_array
    return True
```

Ukážka kódu 13. Funkcia pre výpočet neistoty merania

6.3 Objekt UI

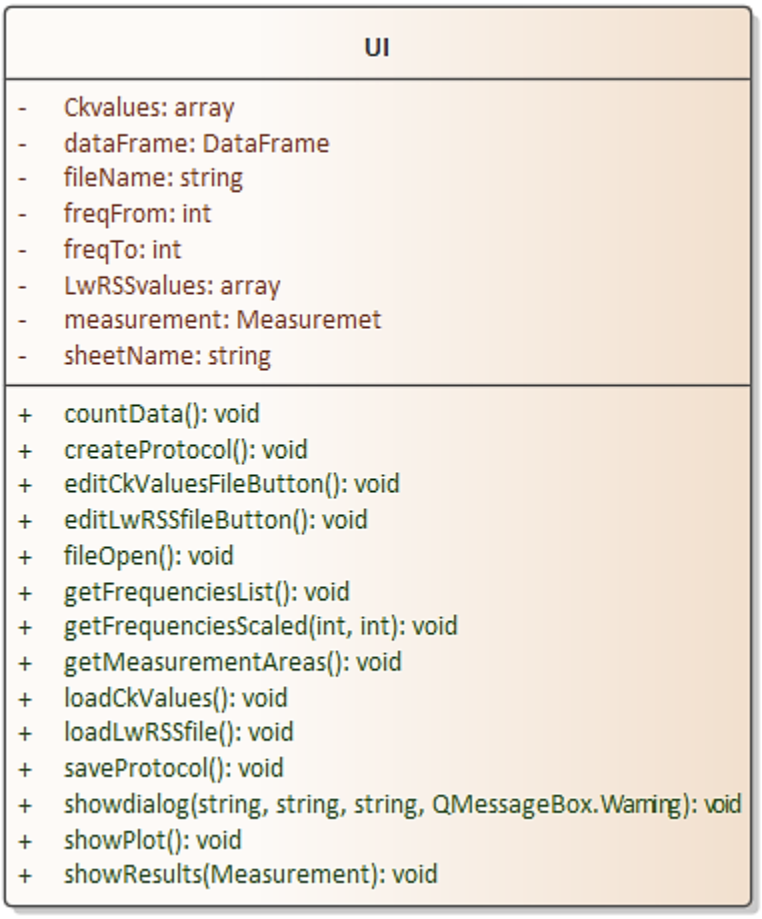
Objekt UI je sadou funkcií, ktoré sa vykonávajú po interakcii niektorého komponentu v grafickom používateľskom rozhraní. Nachádza sa tu:

- funkcia na otvorenie súboru,
- funkcia na vyhľadanie oblastí merania
- funkcia na výpočet výsledkov
- funkcie na upravenie hodnôt potrebných pre výpočty,
- funkcia na zobrazenie výsledkov,
- funkcia na generovanie protokolu.

Okrem funkcií sú v objekte UI definované premenné hodnôt LwRSS a Ck, potrebné na výpočet, ďalej názov súboru s ktorým program pracuje a samotný DataFrame objekt.

Za dôležitú premennú možno považovať premennú s názvom *measurement.* Jedná sa o nami vytvorený objekt typu Measurement. Funkcie s týmto objektom pracujú, pristupujú k dátam, spúšťajú výpočtové funkcie a zobrazujú výsledky. Môžeme povedať, že objekt UI, slúži na ovládanie funkcií objektu Measurement.

Vysvetlenie funkcií objektu je popísané v kapitole Popis



Obrázok 20. Objekt UI

7 POPIS SPRACOVANIA A VYHODNOTENIA AKUSTICKÝCH DÁT

Táto kapitola popisuje udalosti v programe vykonané od získania dát zo súboru až po vytvorenie protokolu.

7.1 Získanie DataFrame zo súboru

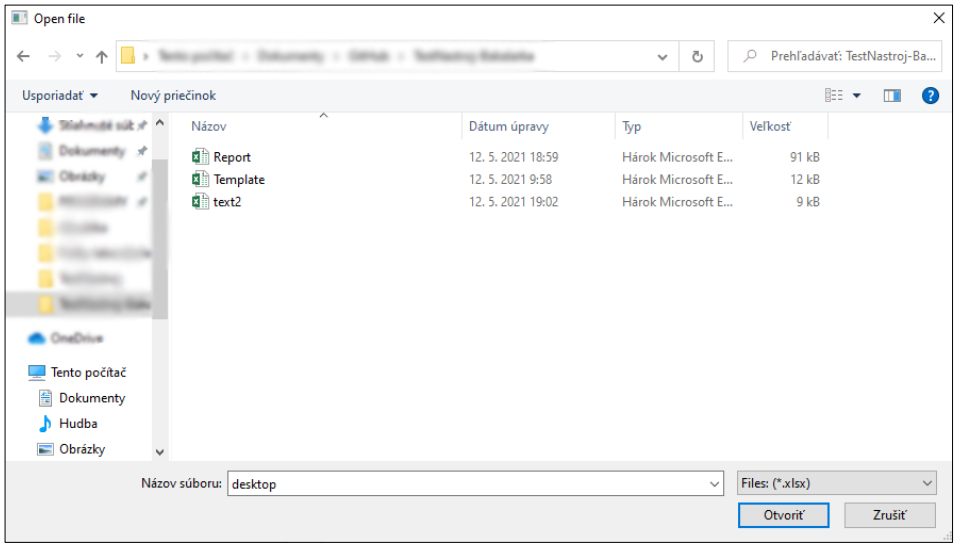
Pre získanie dát zo súboru je navrhnutá funkcia s názvom *fileOpen* v rámci objektu UI, ktorá je spustená po kliknutí na príslušné tlačidlo.

Funkcia vyvoláva dialógové okno kde používateľ vyberie súbor typu .xlsx. Následne funkcia uloží cestu do premennej *fileName* pre neskoršie použitie a pomocou knižnice Pandas a príkazu read_excel získa objekt *dataFrame* s dátami. Po vykonaní týchto operácií sa v GUI zobrazí názov súboru a dátum jeho vytvorenia.

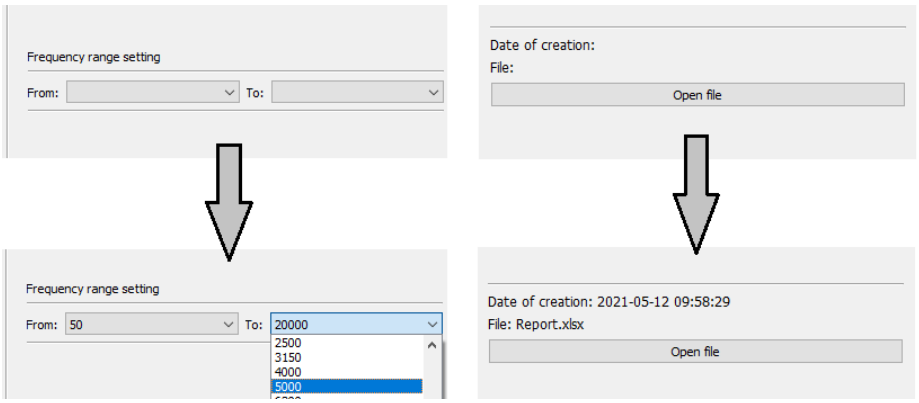
Pomocou funkcie *getFrequenciesList* sa získajú hodnoty frekvencií. Tieto frekvencie sa načítajú do kombo boxov pomocou ktorých používateľ vyberie s akým rozsahom chce vykonať výpočty.

```
def getFrequenciesList(self):
    if self.dataFrame is not None:
        fqList=[]
        fqList = self.dataFrame.keys().values[6:]
        return fqList
```

Ukážka kódu 14. Funkcia getFrequenciesList



Obrázok 21. Dialógové okno pre výber súborov typu .xlsx



Obrázok 22. Načítanie hodnôt frekvencií do kombo boxov a zobrazenie názvu súboru

```
def fileOpen(self):
        file=QFileDialog(self)
        fName = file.getOpenFileName(self, "Open file",
                  "/desktop","Files: (*.xlsx)")

        if fName[0] is None:
            win32api.MessageBox(0, 'Cesta nebola zadaná',
                    'Warning', 0x00001000)
            return
        else:
            self.fileName=fName[0]
            print(self.fileName)

            self.dataFrame= pandas.read_excel(self.fileName,
                  sheet_name=self.sheetName)

            self.labelFileName.setText(
                  "Súbor: "+os.path.basename(self.fileName))

            self.fileCreationDate.setText(
                  "Dátum vytvorenia súboru: %s" % time.strftime(
                        "%Y-%m- %H:%M:%S",time.strptime(time.ctime(
                              os.path.getctime(self.fileName)))))

            self.buttonCountData.setEnabled(True)
            allfreq=self.getFrequenciesList()
            self.comboBoxfrequencyFrom.addItems(map(str, allfreq))
            self.comboBoxfrequencyTo.addItems(map(str, allfreq))
            self.comboBoxfrequencyTo.setCurrentIndex(len(allfreq)-1)
```

Ukážka kódu 15. Funkica fileOpen

[23]:

	Frequency	Unnamed: 1	Unnamed: 2	Unnamed: 3	[Hz]	SumA	50	63	80	100	...
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
1	Surface	Total	Average	NaN	[dB]	56.1	36.40	37.90	44.90	44.00	...
2	NaN	NaN	NaN	Std deviation	NaN	NaN	2.18	1.72	1.81	0.75	...
3	NaN	NaN	NaN	Avg counter	NaN	NaN	30.00	30.00	30.00	30.00	...
4	Surface	NaN	#1 17:20:28	NaN	[dB]	55.6	36.90	34.00	41.50	45.20	...
...	...	...	...	...	...	...	...	...	...	...	...
177	NaN	NaN	NaN	Status	NaN	NaN	NaN	NaN	NaN	NaN	...
178	BackgroundNoise	NaN	#24 15:39:51	NaN	[dB]	19.1	28.70	20.10	17.30	23.30	...
179	NaN	NaN	NaN	Status	NaN	NaN	NaN	NaN	NaN	NaN	...
180	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...
181	Workstation	Total	Average	NaN	[dB]	NaN	NaN	NaN	NaN	NaN	...

Obrázok 23. Získaný objekt dataFrame v prostredí Jupyter

7.2 Vyhľadanie oblastí merania

Na vyhľadanie oblastí v rámci súboru je navrhnutá funkcia s názvom *getMeasurementAreas*. Funkcia prejde každým riadkom súboru a vyhľadá oblasti. Do premennej *value* sa pomocou indexera iloc uloží hodnota riadku v treťom stĺpci.

```
0 nan
1 Average
2 nan
3 nan
4 #1  17:20:28
5 nan
6 #2  17:20:28
7 nan
8 #3  17:20:28
9 nan
10 #4  17:20:28
11 nan
12 #5  17:20:28
13 nan
14 #6  17:20:28
```

Obrázok 24. Hodnoty value prvých 15 riadkov

Z týchto hodnôt funkcia vyberie riadky v ktorých sa nachádzajú znaky *#1*, čo značí začiatok novej oblasti. Pre každý z týchto riadkov je vytvorený objekt typu MeasurementArea s počiatočným menom oblasti definovaného v prvom stĺpci riadku a indexu na ktorom riadok leží. Nakoniec pomocou funkcie append pridáme všetky vytvorené objekty do jedného poľa ktoré táto funkcia vracia.

```
def getMeasurementAreas(self):
    measurementAreas=[]

    lastIndex=self.dataFrame.index.stop

    for i in range(lastIndex):
        value = self.dataFrame.iloc[i].values[2]
        if isinstance(value, str):
            if "#1 " in value:
                areaToAppend = MeasurementArea(
                  self.dataFrame.iloc[i].values[0],i)
                measurementAreas.append(areaToAppend)
    return measurementAreas
```

Ukážka kódu 16. Funkcia getMeasurementAreas

7.3 Procesy spracovania dát

7.3.1 Vytvorenie objektu merania

Funkciu *countData* v objekte UI možno považovať za najdôležitejšiu a to z toho dôvodu že vykonáva kľúčovú úlohu – spracováva dáta, ukladá výsledky a zobrazuje ich.

Táto funkcia sa spúšťa po stlačení tlačidla určeného na výpočet. Prvým krokom je overenie existencie DataFrame. V prípade že podmienka nie je splnená program zobrazí dialógové okno so správou a funkcia nepokračuje.

V prípade že DataFrame existuje, získa sa rozsah frekvencií a funkcia vytvorí objekt typu Measurement ktorý bude reprezentovať vykonávané meranie. Objektu sa priradia hodnoty frekvencií ktoré sú získané z DataFrame pomocou funkcie *getFrequenciesScaled* a hodnoty teploty a tlaku.

Následne sa načítajú hodnoty LwRSS a Ck zo zoznamov uložených v programových súboroch. Na načítanie sa používajú funkcie *loadLwRSSvalues* a *loadCkValues* ktoré prejdú zoznamom frekvencií a skontrolujú či existuje hodnota pre každú frekvenciu. V prípade, že pre niektorú frekvenciu neexistuje hodnota, funkcie vrátia hodnotu false, zobrazí sa dialógové okno so správou a program sa ďalej nevykonáva.

```
def getFrequenciesScaled(self,freqIndexFrom,freqIndexTo):
    allFrequencies = self.getFrequenciesList()
    scaledFrequenciesList = allFrequencies[freqIndexFrom:freqIndexTo]
    return scaledFrequenciesList
```

Ukážka kódu 17. Funkcia getFrequenciesScaled

```
def countData(self):
    if self.dataFrame is not None:

        self.freqFrom=self.comboBoxFrequencyFrom.currentIndex()
        self.freqTo=self.comboBoxFrequencyTo.currentIndex()+1

        self.measurement = Measurement(
            frequencies=self.getFrequenciesScaled(
                   self.freqFrom,self.freqTo),
            airTemperature=float(self.input_airTemperature.text()),
            staticPressure=float(self.input_pressure.text()))

        if self.loadLwRSSfile() == False:
            return
        if self.loadCkValues() == False:
            return
        ...
    else:
        self.showdialog("Warning","Nebol vybraný súbor",
            "Nemožno načítať DataFrame",QMessageBox.Information)
```

Ukážka kódu 18. Začiatok funkcie countData

7.3.2 Pridanie oblastí a získanie dát

Po vytvorení objektu typu Measurement a načítaní potrebných hodnôt sa vytvorí pole oblastí s názvom *measurementAreas*. Do tohto poľa sa pridajú oblasti merania použitím funkcie *getMeasurementsAreas* popisovanej v kapitole 7.2. Použitím indexeru iloc sa získajú hodnoty *stdValues* a hodnoty *measurements* priamo z riadkov DataFrame pre každý objekt typu MeasurementArea v poli *measurementAreas*.

Keďže hodnoty začínajú až v šiestom stĺpci, pre získanie hodnôt je použítý index [6:] a následne index vo forme [od:do], pre vybranie hodnôt podľa zadaného rozsahu. Po získaní všetkých dát sa hodnoty poľa *measurementAreas* pridajú do objektu measurement, konkrétne do premennej *areas.*

```
...
measurementAreas = self.getMeasurementAreas()

for area in measurementAreas:
    areaName = area.name
    startingIndex = area.startingIndex

    area.stdValues =self.dataFrame
      .iloc[startingIndex-2].values[6:][self.freqFrom:self.freqTo]

    for row in range(startingIndex,self.dataFrame.index.stop):
        nameValue = self.dataFrame.iloc[row].values[0]
        if  nameValue == areaName:
            area.measurements.append(
            self.dataFrame.iloc[row].values[6:][self.freqFrom:self.freqTo])
            area.measurementsCount += 1

    self.measurement.areas.append(area)
...
```

Ukážka kódu 19. Získanie dát z oblastí

7.3.3 Výpočet výsledkov

Po získaní dát pre každú oblasť, funkcia *countData* postupne spustí všetky výpočtové funkcie ktoré objekt typu Measurement ponúka. Tieto funkcie následne spracujú dostupné dáta a podľa definovaných vzorcov vypočítajú všetky požadované výsledky. Výsledky sa uložia do premenných so začiatočným menom *result* v danom objekte *measurement*. Funkcie sa musia vykonať v určenom poradí pretože sú závislé od vypočítaných výsledkov z predchádzajúcich. Nakoniec sa spustí funkcia s názvom *showResults* pre zobrazenie výsledkov v grafickom prostredí.

```
...
self.measurement.countLPSTcorrected()
self.measurement.countLPST16median()
self.measurement.countC2()
self.measurement.countLpRSScorrected()
self.measurement.countLpRSS17median()
self.measurement.countC2()
self.measurement.countLW()
self.measurement.countLwa()

self.showResults(self.measurement)

self.createProtocol(self.measurement)

self.buttonShowPlot.setEnabled(True)
```

Ukážka kódu 20. Spustenie funkcií na výpočet a zobrazenie výsledkov

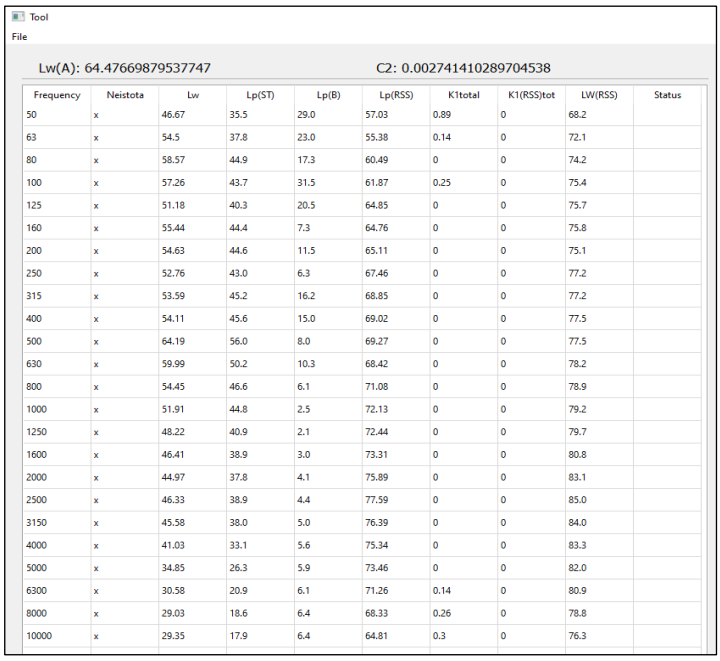
7.4 Zobrazenie výsledkov

Po zavolaní funkcie *showResults* objektu sa nastavia názvy stĺpcov v hlavičke tabuľky. Vynuluje sa počet riadkov v tabuľke a zobrazia sa hodnoty C2 a Lw(a) získané z objektu measurement.

Následne sa pomocou funkcie for vytvoria riadky s frekvenciou a prislúchajúcimi hodnotami výsledkov pre danú frekvenciu.

```
def showResults(self,measurement):
    self.tableWidget.setHorizontalHeaderLabels(['Frequency','Neistota',
    'Lw','Lp(ST)', 'Lp(B)','Lp(RSS)', 'K1total','K1(RSS)tot','LW(RSS)'])
    self.tableWidget.setRowCount(0)
    self.labelC2.setText("C2: "+str(measurement.resultsC2))
    self.labelLwA.setText("Lw(A): "+str(measurement.resultLwa))
    for index in range(len(measurement.frequencies)):
        if index not in []:
            rowPosition = self.tableWidget.rowCount()
            self.tableWidget.insertRow(rowPosition)
            self.tableWidget.setItem(rowPosition , 0,
            QTableWidgetItem(str(measurement.frequencies[index])))
            self.tableWidget.setItem(rowPosition , 1,
            QTableWidgetItem(str('x')))
            ...
```

Ukážka kódu 21. Funkcia showResults



Frequency	Neistota	Lw	Lp(ST)	Lp(B)	Lp(RSS)	K1total	K1(RSS)tot	LW(RSS)	Status
50	x	46.67	35.5	29.0	57.03	0.89	0	68.2	
63	x	54.5	37.8	23.0	55.38	0.14	0	72.1	
80	x	58.57	44.9	17.3	60.49	0	0	74.2	
100	x	57.26	43.7	31.5	61.87	0.25	0	75.4	
125	x	51.18	40.3	20.5	64.85	0	0	75.7	
160	x	55.44	44.4	7.3	64.76	0	0	75.8	
200	x	54.63	44.6	11.5	65.11	0	0	75.1	
250	x	52.76	43.0	6.3	67.46	0	0	77.2	
315	x	53.59	45.2	16.2	68.85	0	0	77.2	
400	x	54.11	45.6	15.0	69.02	0	0	77.5	
500	x	64.19	56.0	8.0	69.27	0	0	77.5	
630	x	59.99	50.2	10.3	68.42	0	0	78.2	
800	x	54.45	46.6	6.1	71.08	0	0	78.9	
1000	x	51.91	44.8	2.5	72.13	0	0	79.2	
1250	x	48.22	40.9	2.1	72.44	0	0	79.7	
1600	x	46.41	38.9	3.0	73.31	0	0	80.8	
2000	x	44.97	37.8	4.1	75.89	0	0	83.1	
2500	x	46.33	38.9	4.4	77.59	0	0	85.0	
3150	x	45.58	38.0	5.0	76.39	0	0	84.0	
4000	x	41.03	33.1	5.6	75.34	0	0	83.3	
5000	x	34.85	26.3	5.9	73.46	0	0	82.0	
6300	x	30.58	20.9	6.1	71.26	0.14	0	80.9	
8000	x	29.03	18.6	6.4	68.33	0.26	0	78.8	
10000	x	29.35	17.9	6.4	64.81	0.3	0	76.3	

Obrázok 25. Zobrazenie výsledkov v GUI vo forme tabuľky

7.5 Vytvorenie a zobrazenie grafu

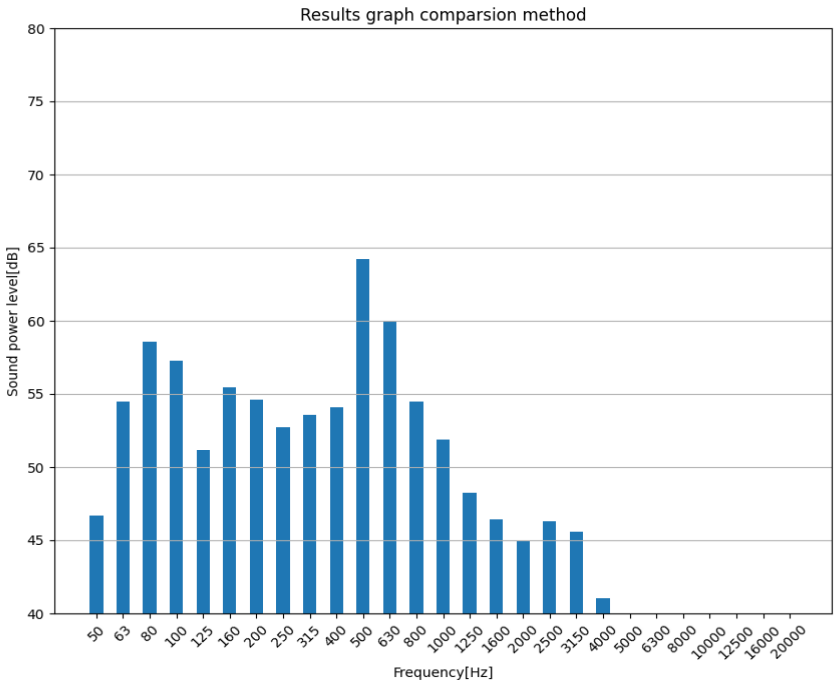
Pre zobrazenie grafu stĺpcového typu je navrhnutá funkcia *showPlot* pomocou ktorej si používateľ môže graf prehliadnuť. Na vytvorenie grafu je použitá knižnica **matplotlib.pyplot**. Funkcia najprv získa hodnoty frekvencií ktoré sa umiestnia na os x, a hodnoty výsledkov ktoré sa umiestnia na os y. Následne sa upraví štýlovanie grafu, konkrétne sa nastavuje veľkosť, šírka stĺpcov, natočenie popisu stĺpcov a pomenovanie osi x a y. Pomocou funkcie ylim vieme limitovať najväčšiu a najmenšiu zobrazovanú hodnotu na osi y.

Keďže vygenerovaný graf bude možno vyžadovaný aj pri iných úkonoch (ako príloha k protokolu) rozhodol som sa pre uloženie grafu vo forme obrázku pomocou príkazu *plt.savefig*. Obrázok grafu sa uloží ihneď po vytvorení a bude dostupný pokým ho nenahradí nový.

Zobrazenie grafu sa otvorí uložený obrázok v predvolenom programe na prehliadanie obrázkov, slúži na to funkcia *image.show()*.

```
def showPlot(self):
        plt.rcdefaults()

        frequencies = self.measurement.frequencies
        x_pos = np.arange(len(frequencies))
        performance = self.measurement.resultsLw

        plt.figure(figsize=(10, 8))
        plt.bar(x_pos, performance, align='center', alpha=1,width=0.5)
        plt.xticks(x_pos, frequencies)
        plt.xticks(rotation=45)
        plt.grid(axis = 'y')
        plt.ylim([40, 80])

        plt.xlabel('Frequency[Hz]')
        plt.ylabel('Sound power level[dB]')
        plt.title('Results graph comparsion method')
        plt.savefig('generatedGraph.png')

        img = Image.open('generatedGraph.png')
        img.show()
```

Ukážka kódu 22. Funkcia showPlot



Obrázok 26. Výsledok vygenerovaného grafu

7.6 Vytvorenie a uloženie protokolu

Po dokončení všetkých výpočtov sa spúšta funkcia createProtocol, ktorá vytvorí kópiu predlohy, otvorí ju a prepíše hodnoty buniek na vybraných miestach. Pomocou funkcie for pridá hodnoty LW do tabuľky, z ktorej sú používane pre vopred vytvorený graf.

Protokol sa po vyplnení uloží pod názvom last.xlsx do prečinku s programom a pomocou funkcie *os.system* sa zobrazí na obrazovke. Používateľ ho môže upraviť a následne uložiť v požadovanom formáte.

```
def createProtocol(self,measurement):

    print(self.fileName)
    print("Creating TemplateTemp")
    shutil.copy2( 'TemplateNewDocument.xlsx','TemplateTemp.xlsx')
    xfile = openpyxl.load_workbook('TemplateTemp.xlsx')
    sheet = xfile["MAIN"]

    sheet['K9']="%s" % time.strftime("%d.%m.%Y %H:%M:%S",
            time.strptime(time.ctime(os.path.getctime(self.fileName))))

    sheet['F15']=self.measurement.staticPressure
    sheet['F16']=self.measurement.airTemperature
    sheet['F17']=self.air_humidity.text()
    sheet['F18']=self.room_volume.text()
    sheet['F19']=self.room_area.text()
    sheet['J17'] = self.measurement.resultsC2
    sheet['K48'] = self.measurement.resultLwa

    rowNumber = 24
    for index in range(len(measurement.frequencies)):
        sheet.cell(row=rowNumber, column=2).value =
                  measurement.frequencies[index]
        sheet.cell(row=rowNumber, column=3).value  =
                  measurement.resultsLw[index]
        sheet.cell(row=rowNumber, column=4).value  = '± '+ str(round(
                  measurement.resultsUncertaintyExtended[index],1))
        rowNumber = rowNumber+1

    xfile.save('last.xlsx')
    xfile.close()
    os.remove('TemplateTemp.xlsx')

    os.system("start EXCEL.EXE last.xlsx")
```

Ukážka kódu 23. Funkcia createProtocol

CEBIA-Tech
Centrum bezpečnostních, informačních a pokročilých technologií

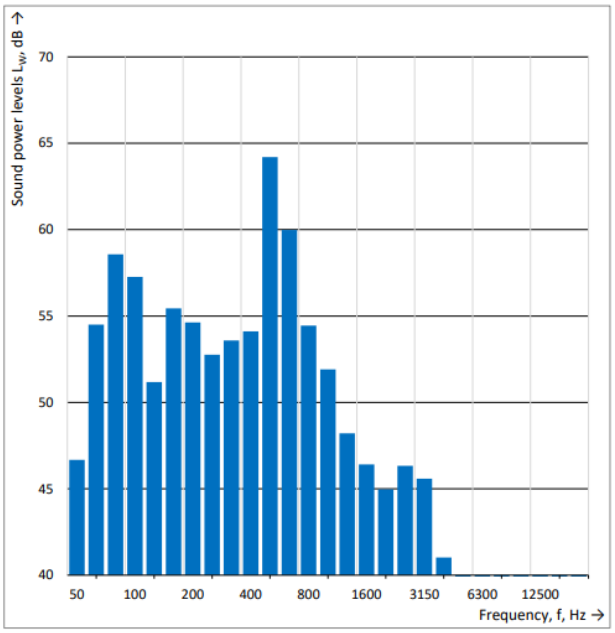
Sound power levels according to ISO 3741:2010

Precision methods for reverberation test rooms - comparism method

Date and time of test: 12.05.2021 09:58:29

Client:	Name		
Tested object:	Object		
Mounting conditions:	The measuring device was mounted asymmetrically on the solid framework.		
Static pressure:	99,7 kPa	Test object dimensions:	550x550x240 mm
Air temperature:	19,8 °C		
Relative air humidity:	38.9	Korection C2:	0,00274141 dB
Test room volume:	93.7	Note:	
Area of test room:	129	*The resulting values refer to 100% load of the unit.	

Frequency f [Hz]	LW 1/3 octave [dB]	U(Lw) [dB]
50	46,7	± 6.2
63	54,5	± 4.9
80	58,6	± 5.1
100	57,3	± 2.1
125	51,2	± 2.6
160	55,4	± 6.5
200	54,6	± 3.9
250	52,8	± 1.6
315	53,6	± 1.7
400	54,1	± 0.8
500	64,2	± 3.5
630	60,0	± 2.0
800	54,4	± 1.3
1000	51,9	± 1.8
1250	48,2	± 1.3
1600	46,4	± 2.0
2000	45,0	± 2.2
2500	46,3	± 1.5
3150	45,6	± 0.8
4000	41,0	± 1.4
5000	34,9	± 1.2
6300	30,6	± 0.7
8000	29,0	± 0.9
10000	29,3	± 1.2
12500	27,5	± 1.1
16000	23,6	± 1.3
20000	26,2	± 1.4



Sound power level LW(A): 64,5 dB

Obrázok 27. Vytvorený protokol

ZÁVER

Cieľom tejto práce je navrhnutie nástroja ktorý spracuje dáta pre potreby Laboratória techniky prostredia.

Po zoznámení sa s postupom vykonávania akustických meraní a analýze súboru s dátami som navrhol softvérový nástroj pomocou ktorého sa dáta získavajú, spracovávajú a vyhodnocujú podľa príslušných noriem. Detailne predstavujem svoje nápady a riešenia problematiky pomocou ukážok kódu so stručným popisom.

Program slúži ako náhrada pre spracovanie dát pomocou programu MS Excel avšak v prípade nasadenia programu do bežného používania ho bude potrebné dostatočne otestovať nielen v časti GUI ale aj pomocou testovacích dát a sledovať či ich program vyhodnocuje správne.

Vďaka tejto práci som získal nové vedomosti v oblasti programovania v jazyku Python, konkrétne v oblasti spracovania dát a taktiež som mal možnosť pracovať na niečom čo môže niekto v budúcnosti reálne využiť.

ZOZNAM POUŽITEJ LITERATÚRY

[1] BLAHO, Andrej. Jazyk Python [online]. [cit. 2021-5-6]. Dostupné z: http://python.input.sk/01.html#jazyk-pythonDruhý zdroj

[2] STX NEXT. What Is Python Used for? [online]. [cit. 2021-5-6]. Dostupné z: https://www.stxnext.com/what-is-python-used-for/

[3] LUTZ, Mark, c2009. Learning Python. 4th ed. Sebastopol: O'Reilly. ISBN 978-0-596-15806-4.

[4] Python classes [online]. [cit. 2021-5-9]. Dostupné z: https://www.w3schools.com/python/python_classes.asp

[5] UZAK, Martin. Python [online]. [cit. 2021-5-9]. Dostupné z: http://zenis.dnp.fmph.uniba.sk/prirucky/python/python6.htm

[6] Úvod do objektovo orientovaného programovania v Pythone [online]. [cit. 2021-5-9]. Dostupné z: https://www.itnetwork.sk/python/oop/python-tutorial-uvod-do-objektovo-orientovaneho-programovani

[7] VIKTORIN, Petr a Miro HRONČOK, 2017. Nauč se Python! [online]. [cit. 2021-5-6]. Dostupné z: https://naucse.python.cz/lessons/intro/pandas/

[8] THUMAR, Chirag, 2018. Why Python is Good for Data Analytics? [online]. [cit. 2021-5-6]. Dostupné z: https://thriveglobal.com/stories/why-python-is-good-for-data-analytics/

[9] KAZIL, Jacqueline a Katharine JARMUL, 2016. Data Wrangling with Python: TIPS AND TOOLS TO MAKE YOUR LIFE EASIER. 2016-02-02 First Release. United States of America: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. ISBN 978-1-491-94881-1.

[10] About pandas [online]. [cit. 2021-5-7]. Dostupné z: https://pandas.pydata.org/about/index.html

[11] VANDERPLAS, Jacob T., [2017]. Python data science handbook: essential tools for working with data. Beijing: O'Reilly. ISBN 978-149-1912-058.

[12] LANDUP, David. How to Set Axis Range (xlim, ylim) in Matplotlib [online]. [cit. 2021-5-9]. Dostupné z: https://stackabuse.com/how-to-set-axis-range-xlim-ylim-in-matplotlib/

[13] GAZONI, Eric a Charlie CLARK, 2021. Openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files [online]. [cit. 2021-5-10]. Dostupné z: https://openpyxl.readthedocs.io/en/stable/index.html

[14] HENRIQUE VASCONCELLOS, Paulo, 2018. Top 5 Python IDEs For Data Science [online]. [cit. 2021-5-10]. Dostupné z: https://www.datacamp.com/community/tutorials/data-science-python-ide

[15] BURIAN, Libor a Hana STANKOVÁ, 2015. Python pre Geovedné aplikácie. Bratislava: Univerzita Komenského v Bratislave. ISBN 978-80-223-3947-6.

[16] UROOJ, Wajiha, 2019. What is Python Spyder IDE and How to use it? [online]. [cit. 2021-5-10]. Dostupné z: https://medium.com/edureka/spyder-ide-2a91caac4e46

[17] Welcome to Spyder's Documentation [online]. [cit. 2021-5-10]. Dostupné z: https://docs.spyder-ide.org/current/index.html

[18] M. PERKEL, Jeffrey, 2018. Why Jupyter is data scientists' computational notebook of choice [online]. [cit. 2021-5-10]. Dostupné z: https://www.nature.com/articles/d41586-018-07196-1

[19] Instantly Launch a Jupyter Notebook [online]. [cit. 2021-5-10]. Dostupné z: https://community.next.tech/t/instantly-launch-a-jupyter-notebook/216

[20] Qt Designer Manual [online], 2021. [cit. 2021-5-10]. Dostupné z: https://doc.qt.io/qt-5/qtdesigner-manual.html

[21] Qt designer python [online]. [cit. 2021-5-10]. Dostupné z: https://pythonbasics.org/qt-designer-python/

[22] DRÁBEK, Pavel. Návrh zkoušení akustických parametrů zařízení techniky prostředí v kalorimetrické komoře. Zlín: Univerzita Tomáše Bati ve Zlíně, 2014, 121 s. (175 415 znaků). Dostupné také z: http://hdl.handle.net/10563/30212. Univerzita Tomáše Bati ve Zlíně. Fakulta aplikované informatiky, Ústav automatizace a řídicí techniky. Vedoucí práce Zálešák, Martin.

[23] ČSN EN ISO 3741. Akustika – Určování hladin akustického výkonu a hladin akustické energie zdrojů hluku pomocí akustického tlaku: Přesné metody pro dozvukové zkušební místnosti. Praha: Český normalizační institut, 2011.

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATEK

atď.	A tak ďalej
a pod.	A podobne
angl.	Anglicky
tzv.	Takzvaný
akust.	Akustický
kap.	Kapitola
FAI	Fakulta aplikovanej informatiky
UTB	Univerzita Tomáše Bati
POSIX	Portable Operating System Interface Prenosné rozhranie pre operačné systémy
OS	Opreračný systém
GUI	Graphical User Interface Grafické používateľské rozhranie
API	Application programming interface Rozhranie pre programovanie aplikácií
SQL	Structured Query Language Štruktúrovaný dopytovací jazyk
FTP	File Transfer Protocol Protokol prenosu súborov
HTML	HyperText Markup Language Hypertextový značkovací jazyk
XML	Extensible Markup Language Rozšíriteľný značkovací jazyk
SOAP	Simple Object Access Protocol

	Protokol jednoduchého objektového prístupu
OOP	Objektovo Orientované Programovanie
IDE	Integrated Development Environment
	Integrované vývojové prostredie

ZOZNAM OBRÁZKOV

ZOZNAM UKÁŽOK KÓDU

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

PRÍLOHA P I: CD-ROM

Priložené CD obsahuje

- Zdrojové kódy aplikácie uložené vo formáte .zip
- Bakalársku prácu vo formáte .pdf