Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Analýza dat z inteligentního včelího úlu

Bakalářská práce

Petr Hála
Vedoucí práce: Ing. Miroslav Skrbek, Ph.D.

České Budějovice 2020

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Petr Hála
(jméno, příjmení, tituly)

Obor – zaměření studia: 1801R001 / Aplikovaná informatika

Katedra: Ústav aplikované informatiky

Školitel: Ing. Miroslav Skrbek, Ph.D.
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF: .....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Analýza dat z inteligentního včelího úlu

Cíle práce:

Seznamte se s inteligentním včelím úlem provozovaným Biologickým centrem AV ČR v Českých Budějovicích a formáty dat nasbíranými za období provozu. Navrhněte a implementujte programové vybavení pro analýzu a vizualizaci dat. Zaměřte se zejména na zpracování audio záznamů bzukotu včel uvnitř úlu. Uvažujte také možnost kombinace s ostatními měřenými veličinami v úlu. Při analýze aplikujte metody zpracování signálu, jako je filtrace a FFT. Programové vybavení navrhněte tak, aby data, která jsou výsledkem analýzy mohla být vkládána dávkově i průběžně do databáze. V datech vyhledejte zajímavé časové úseky, pro které umožníte jejich vizualizaci. Programové vybavení vyvíjejte tak, aby dílčí algoritmy byly použitelné ve formě knihoven v Pythonu. Rozsah analýzy a způsob analýzy dat upřesněte po dohodě s vedoucím práce.

Základní doporučená literatura: literaturu dodá vedoucí práce v průběhu řešení bakalářské práce.

Financování práce : práce nemá finanční nároky

Vedoucí práce : Ing. Miroslav Skrbek, Ph.D. .....podpis : .....

U externích vedoucích fakultní garant práce.....podpis : .....

Garant oboru bak.. studia (nepožaduje se u zaměření „příprava na mag. studium biologie)

..... podpis : .....

Vedoucí katedry ..... podpis

.....

Případný souhlas vedoucího ústavu AV .....podpis : .....

V Českých Budějovicích dne .....

Převzal/a dne..... podpis : .....

Bibliografické údaje:

Hála, P., 2020: Analýza dat z inteligentního včelího úlu. [Analysis of data from an intelligent beehive. Bc. Thesis, in Czech.] – 72 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic

Anotace:

Cílem bakalářské práce je návrh a implementace programového vybavení, které zpracovává zvuková data ze včelího úlu. Audio signál je analyzován ve frekvenční doméně s využitím rychlé Fourierovy transformace a ve specifických frekvenčních pásmech pomocí filtrace v časové doméně. RMS filtrovaných signálu je do databáze ukládáno dávkově i průběžně a vizualizováno pomocí webové aplikace. Analýzu dílčích zvukových záznamů lze zobrazovat pomocí modulů disponujících v jazyce Python s možností výstupu grafů do PDF. Nad daty byla provedena shluková analýza pomocí samoorganizujících se map (SOM), kterou byly detekovány různorodé zvukové projevy včelstva v různých ročních obdobích s následnou vizualizací pomocí teplotních map.

Klíčová slova:

Digitální filtr, Fourierova transformace, Python, signál, záznam, shluková analýza, včelí úl

Annotation:

The aim of the bachelor thesis is the design and implementation of software that processes audio data from a beehive. The audio signal is analyzed in the frequency domain using fast Fourier transform and in specific frequency bands using filtering in the time domain. The RMS of filtered signals is stored in the database in batches or continuously and visualized by a web application. Analysis of partial audio records can be displayed by modules available in the Python with the possibility of outputting graphs to PDF. Above the data a cluster analysis was performed with using of self-organizing maps (SOM), which detected various types of sound manifestations of the bee colony in different seasons, followed by visualization in heatmaps.

Keywords:

Digital filter, Fourier transform, Python, signal, record, cluster analysis, beehive

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne: 21.11.2020 Podpis autora: 

Poděkování:

Rád bych touto cestou poděkoval Ing. Miroslavovi Skrbkovi, Ph.D. za jeho odborné konzultace ohledně implementace a volby programového vybavení a Ing. Václavu Krištůfkovi, CSc. za jeho cenné rady ve včelí problematice.

Samozřejmě bych rád poděkoval i rodině za podporu ve studiu.

Obsah

1 Úvod

Tato práce se zabývá návrhem a implementací programového vybavení pro analýzu včelích akustických dat z inteligentního včelího úlu. To nám umožní sledovat zvukové chování včel napříč sezónami a sledovat jednotlivé aspekty jejich zvukového chování. Vědci, kteří se již dávno v minulosti zabývali včelím bzukotem, byli například Eskow, Hanson, Lasz a Woods. Díky nim je nám dostupná vědecká literatura, která poskytuje informace o chování včelstva při bzukotu na určitých frekvencích, a můžeme tak odhadnout při přezkumu i dalších měřených veličin, co se v úlu odehrává. Ve včelím úlu je možno slyšet zvuky v okruhu 20 - 10 000Hz. Naše oblast zájmu se však specifikuje na rozsah včelího bzučení 100 - 500Hz, jelikož intenzita včelího bzučení je v této oblasti nejsilnější. **[1]** Včelí zvukové projevy je tedy vhodné rozdělovat do několika frekvenčních pásem a umožnit pro ně vykreslení, abychom měli lepší náhled na včelí akustické chování. Je známo, že včelstev je neustálý úbytek, a tato práce poskytuje odborný náhled na zvukové chování včel v nepřetržitém časovém úseku a umožní udělat celkovou představu o jejich reakcích na různé situace. Zvukové chování může též poukázat na nemoci nebo například napadení úlu, pro což jsou charakteristické určité frekvence, které jsou vyšší než ty běžné.

Sledovat zvukové chování (tedy i včel) lze dvěma způsoby:

1. Sledování amplitudy signálu v daném frekvenčním rozsahu a přiřazovat jí k určitému vzorci chování.
2. Moderní metodou pokročilé akustické analýzy, kterou představují neuronové sítě. **[2]**

Dále je důležité zmínit, že na trhu není obdobný systém pro zpracovávání včelího chování v porovnání s tím, který pro účely inteligentního včelího úlu Biologického centra AV ČR v Českých Budějovicích byl vyvinut mými kolegy, kteří na této problematice pracovali, a mnou. Tento systém totiž nabízí vědecký náhled na všechna spektra včelího chování a napomáhá k řešení problematiky zefektivnění včelařství.

Práce byla finančně podpořena projektem Strategie AV 21 udělené Biologickému centru AV ČR, v. v. i. v Č. Budějovicích (2019-2020).

Obrázek 1 - Náhled na inteligentní včelí úl provozovaný AV ČR v Českých Budějovicích

1. 1 Členění práce

Teoretická část práce (kapitola ***3***) se zabývá zvukovými projevy včelstva, rozborem problematiky kolem digitalizace signálu a jeho následné analýzy. Tato část práce čerpá informace z odborné literatury a poskytuje potřebné znalosti pro návrh a implementaci programového vybavení.

Kapitoly *4 - 7* řeší celkový rozbor problematiky pro praktickou část práce. Kapitola ***4*** poskytuje rešerši obdobných technologií, a tedy možný vývoj mého programového vybavení doplněný o porovnání mnou navržených programů pro analýzu včelích akustických dat. Kapitola ***5*** seznamuje čtenáře se sběrem akustických dat za období provozu a kapitoly ***6-7*** popisuje postup a využité programové vybavení související s operačním systémem serveru a programovacím jazykem pro praktickou část této práce.

Praktická část práce (kapitola ***8***) se zabývá návrhem a implementací programového vybavení aplikovaného na včelí akustická data. Tato část práce popisuje mnou navrhované a implementované řešení, které se skládá ze spektrální analýzy, číslicové filtrace a shlukové analýzy. Je zde popsán samotný postup řešení, způsoby ukládání výsledků analýz a možnosti jejich vizualizace.

Kapitola *9* se zaobírá testováním programového vybavení určeného pro spektrální analýzu a číslicovou filtraci z praktické části práce (kapitoly ***8***).

Kapitola *10* čtenáři poskytuje diskuzi ohledně výsledků číslicové filtrace a shlukové analýzy aplikované na včelařský rok. Kapitola ***10. 1*** nabízí diskuzi ohledně výsledků shlukové analýzy a jejího celkového rozboru skrze jednotlivá období včelařského roku. Kapitola ***10. 2*** naopak nabízí několik zajímavých momentů (rojení a léčení) pomocí číslicové filtrace a následně vizualizované prostřednictvím webové aplikace.

2 Cíl

Nejprve je nutné se seznámit s inteligentním včelím úlem provozovaným Biologickým centrem AV ČR v Českých Budějovicích a formáty dat nasbíraných za období provozu. Hlavním cílem práce pak bude návrh a implementace programového vybavení pro analýzu a vizualizaci akustických dat. Konkrétně se jedná o zpracování audio záznamů bzukotu včel uvnitř úlu. Cílem je užití metod, jako je číslicová filtrace signálu a rychlá Fourierova transformace (FFT – Fast Fourier transform). Programové vybavení musí být navrženo tak, aby data, která jsou výsledkem analýzy, mohla být vkládána dávkově i průběžně do databáze. Pro výsledky analýzy bude umožněna vizualizace dílčích záznamů pomocí Python modulu matplotlib či tištitelných souborů ve formátu PDF. Nicméně výsledky filtrování bude možné sledovat i v reálném čase webovou aplikací Grafana. Programové vybavení musí být vyvinuto tak, aby dílčí algoritmy byly použitelné ve formě knihoven v Pythonu. Rozsah analýzy bude přesněji specifikován v kapitole ***Metodika*.**

Dalším cílem práce je analýza dat neuronovými sítěmi s cílem znázornit charakteristické včelí zvukové chování napříč sezónami.

Kroky, které vedou k úspěšnému cíli práce:

1. Seznámení se s včelími audio daty za období provozu a jejich zpracování serverem.
2. Rešerše obdobných technologií pro analýzu včelích audio dat.
3. Metodika - postup analýzy audio dat.
4. Návrh programového vybavení.
5. Implementace programového vybavení.
6. Testování programového vybavení.

3 Teorie

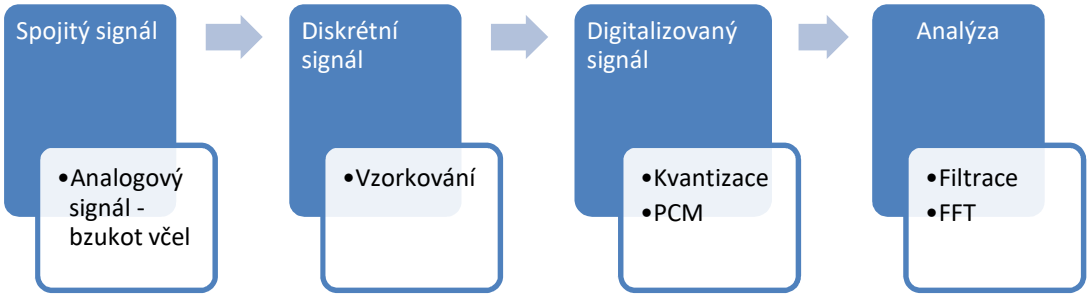
3. 1 Zvukové projevy včelstva

Zvukové signály, které se v úlu projevují, vydává včela nebo kasta včel cíleně při varování či komunikaci. Signál může být ovlivněn i stresorem, jako je léčení, onemocnění nebo špatná teplota. Signál však mohou ovlivnit i vnější jevy. Výsledná kombinace signálů z jednoho specifického záznamu je generována přímo včelou nebo oscilací substrátu (plástu), po kterém se včely pohybují. **[2]** Níže je přiložena tabulka, která stručně charakterizuje zvukové chování včel.

Interpretace, vzorec chování	Typ signálu	Rod	Základní frekvence (Hz)	Přibližný tvar signálu	Signál vysílá
Informace pro dělnice	35	-----			
Zamezení líhnutí dalších matek	„Troubení"	*A. mellifera*	350-500	Sekvence pulsů	Matka A
		A. cerana	2000-3000	Sekvence pulsů	Matka A
Informace pro Matku A a dělnice o přítomnosti a životaschopnosti další matky	„Kvákání"	*A. mellifera*	200-350	Sekvence pulsů	Matka B
Obrana/varování	„Syčení"	*A. mellifera*	Od 3000	Širokopásmový puls	Včelstvo
		A. cerana	300-3600		
Rekrutování dělnic na donášku nektaru – spojeno se závěrečnou fází jednoho komunikačního „tance"	„Pískání"	*A. mellifera*	300-500	Sekvence pulsů	Slídilka
Vybízí roj k odletu v průběhu rojení	„Pískání"	*A. mellifera*	100-2000	Sekvence pulsů	Průzkumnice, která hledá hnízdo nového roje
Upoutávání pozornosti dělnic, nosiček pylu resp. nektaru. Informace o výdatnosti a lokaci zdroje potravy	„Pískání"	*A. mellifera*	200-350	Puls	Slídilka

Tabulka 1 - Zvukové chování včel [38]

3. 2 Digitální zpracování signálu

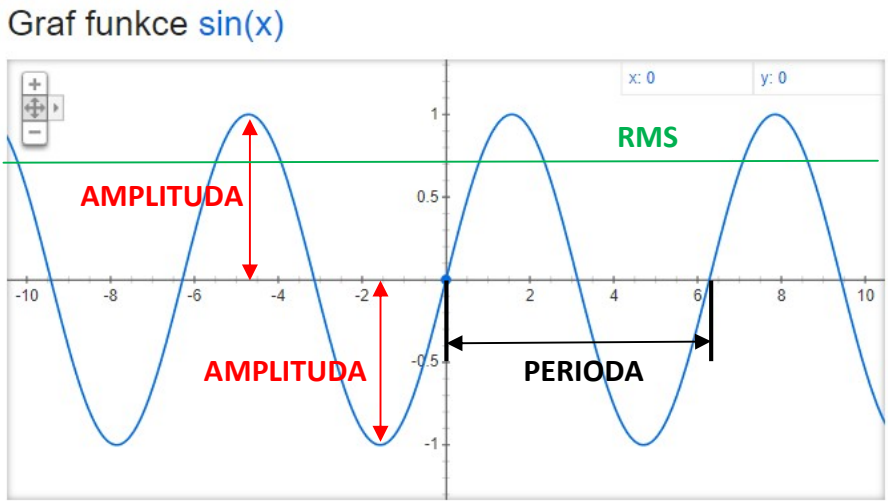


Obrázek 2 - Proces zpracování zvukového signálu

3. 2. 1 Spojitý signál

Je takový signál, jenž je spojitou funkcí času (popřípadě frekvence). Signál může být vyjádřen graficky, matematicky nebo tabulkou hodnot. Reálné signály se špatně matematicky vyjadřují, a proto se vytváří vhodný model nebo jejich aproximace.

Signály mohou být konečné či nekonečné. **[3]** Typickým příkladem spojitého signálu je periodický sinusový signál, znázorněný na obrázku 3.



Obrázek 3 - Průběh spojitého signálu

Na obrázku 3 je znázorněn signál **y = sin(x)**, pro který platí:

- **Perioda** – Doba, po které se signál opakuje. **[3]**

$$T = \frac{1}{f} \tag{1}$$

  - Perioda je v tomto případě 2∏.

- **Frekvence** – Počet opakování periody za časový úsek. **[3]**

$$f = \frac{1}{T} \tag{2}$$

$$\Omega = \frac{2\prod}{T} \tag{3}$$

  - Frekvence je v tomto případě 1Hz, protože periodický děj proběhne jednou za 2∏.

- **Amplituda** – Bod v největší vzdálenosti od středu signálu (vlny). **[4]**
  - Amplituda nabývá hodnoty 1

- **RMS** – Efektivní hodnota

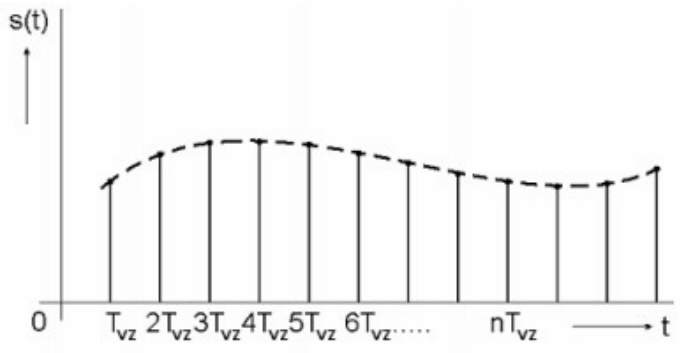
$$RMS(sin) = \frac{a}{\sqrt{2}} \tag{4}$$

  - RMS nabývá hodnoty přibližně 0,707

Takováto funkce popsaná na obrázku 3 i pro případ, že by byla časově omezená, by disponovala nekonečně mnoho výstupními hodnotami (pro každé x) a je tedy vhodné signál digitalizovat, aby byl výpočetně méně náročný, tím pádem se lépe uchovával a v případě signálů s šumem nastává možnost tento šum eliminovat.

3. 2. 2 Vzorkování signálu

Nejběžnějším příkladem vzniku diskrétního signálu je vzorkování spojitého signálu. Představme si, že na vstupu je spojitý signál, který je spínaný, a na výstupu se objeví vzorky, které jsme získali při každém sepnutí. Signál, který jsme tímto způsobem získali, nazýváme diskrétní signál. Vzorkování může být rovnoměrné, pokud je vzdálenost mezi dvěma sepnutími konstantní, nebo nerovnoměrné. Vzdálenost mezi dvěma vzorky označujeme $\boldsymbol{T_{VZ}}$ a říká se jí vzorkovací perioda, z čehož vychází vzorkovací frekvence $\boldsymbol{f_{VZ}}$, což je počet vzorků získaných za určitý čas, například za jednu vteřinu při zvukových záznamech. **[5]**



Obrázek 4 - Průběh vzorkovaného signálu [39]

Vzorkováním vždy dojde ke ztrátě informace, ale je důležité tuto ztrátu minimalizovat. V případě toho, že signál je frekvenčně omezen frekvencí $\boldsymbol{f_{MAX}}$, pro správné určení vzorkovací frekvence slouží **Shannonův teorém**, což je jednoduché matematické pravidlo, které určuje, jak má být vysoká frekvence vzorkování pro možnou rekonstrukci signálu z posloupnosti diskrétních vzorků. **[3]**

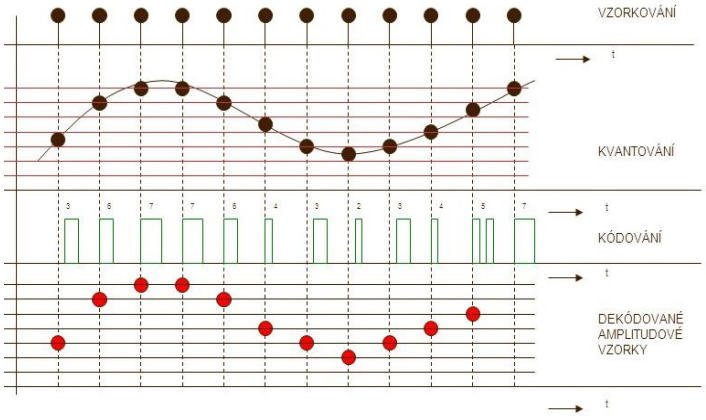
Shannonův teorém: $\boldsymbol{f_{VZ} \geq 2 \cdot f_{MAX}}$ (5)

Pokud nedojde k dodržení Shannonova teorému a vzorkovací frekvence bude nižší, dojde k tzv. **Aliasing efektu**, což vyvolá překryv spekter signálu a signál nelze zpět rekonstruovat, tudíž je vhodné se Shannonovým teorémem řídit. **[3]**

Následně se tento signál digitalizuje pomocí kvantování a kódování (kvantováním a kódováním se zabývá další kapitola).

3. 2. 3 Kvantování a PCM

Metoda, kterou vznikne digitální signál, je kombinace kvantizace a vzorkování současně. Kvantovaný signál je charakteristický svým omezeným oborem hodnot, který je velice ztrátový, proto je vhodné volit vhodné množství kvantizačních hladin. Po nastavení několika kvantizačních hladin, které jsou většinou od sebe stejně vzdálené (tedy lineární) jsou přiřazovány hladiny (kvanta) k hodnotám vzorku. Následně je aplikováno kódování pomocí PCM (pulsní kódové modulace), které odpovídá v binární soustavě přidělené hladině. Následně máme výstupní digitální signál. V Pythonu se o vytvoření digitálního signálu stará modul **pyAudio**. Na obrázku 5 je znázorněno získávání vzorků z původního signálu a následná aplikace kvantování a kódování.



Obrázek 5 - Vznik digitálního signálu [40]

3. 2. 4 Fourierova transformace

Jedná se o základní nástroj pro zpracování signálu. Účelem této transformace je převod z časové složky do frekvenční a naopak. Slouží pro analýzu a rozbor frekvenčního rozsahu signálu a to především pro periodické signály. Při neperiodických je zapotřebí signál rozložit na malé části (okna) a předpokládat, že signál těchto oken je periodický, nebo blízký periodicitě. **[6]**

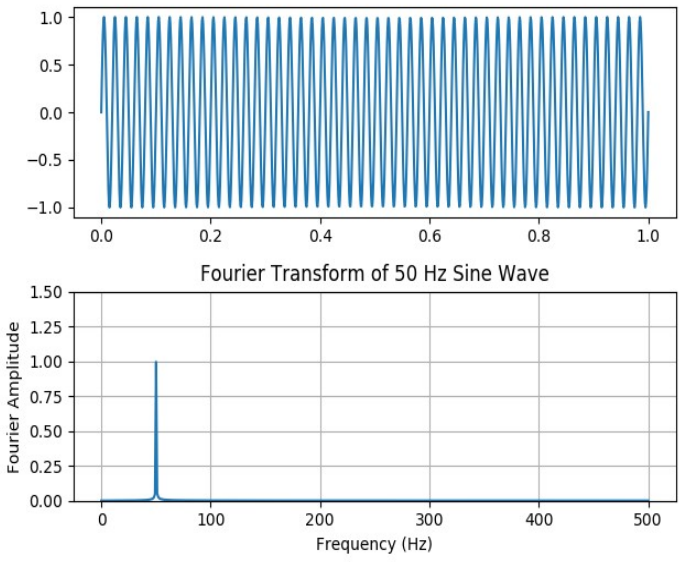
Jestliže ***F(k)*** označuje výsledek Fourierovy transformace a ***f(n)*** je vstupní signál, pak platí vzorec pro výpočet Fourierovy transformace **[6]**:

$$\boldsymbol{F(k) = \sum_{n=0}^{N-1} f(n) e^{\frac{-2\pi i k n}{N}}} \tag{6}$$

Při využití Fourierovy transformace je efektivnější využít algoritmus pro výpočet rychlé Fourierovy transformace, která disponuje lepší asymptotickou složitostí**: O (N log2 N)** než původní Fourierova transformace a stala se tak velice rozšířenou a obsaženou například v Matlabu, Octave či knihovnách Pythonu Numpy a Scipy. **[6]** Pro dosažení efektivního zobrazení transformace při nepeoridicitě zvukového signálu je zapotřebí signál rozdělit do několika oken a na každé z nich aplikovat FFT s překryvy hodnot. Pro výpočet FFT v Pythonu slouží funkce **numpy.fft.fft(a=vstupní signál)**, která vrací jednorozměrnou diskrétní Fourierovu transformaci. **[7]** Následný výpočet amplitudy po zisku Fourierovy transformace, kdy ***F(k)*** označuje výsledek FFT a ***n_samples*** počet vzorků, získáme vzorcem:

$$\boldsymbol{FFT\_amplitude(x) = \sum_{n=0}^{N-1} F(k)_n * \frac{2}{n\_samples}} \tag{7}$$

Na obrázku 6 je vyobrazen typický příklad Fourierovy transformace. Právě Fourierovou transformací se bude zabývat i praktická část práce.



Obrázek 6 - Fourierova transformace sinusového signálu 50Hz [41]

3. 2. 5 Digitální Filtrace

Digitální filtrací se rozumí úprava signálu a velikosti jeho amplitudy (v amplitudové frekvenční charakteristice) s ohledem na frekvenční složku, kdy signál disponuje více kmitočtovými složkami, a cílem je tyto nevyžádané hodnoty náležící určitým frekvencím a šum potlačit. **[8]** Tuto problematiku řeší horní, dolní a pásmová propust (popřípadě zádrž). Právě číslicové filtry jsou vhodné pro náhled na velikost amplitudy ve specifických frekvenčních pásmech.

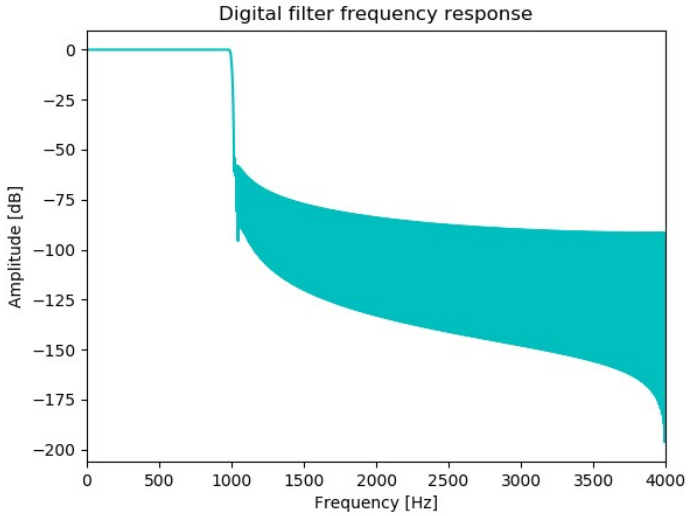
V Pythonu se o návrh digitálních filtrů a jejich implementaci stará například knihovna **Scipy** a její funkce **firwin** a **lfilter**, kdy funkce firwin navrhuje filtr **FIR** a tedy filtr s konečnou impulsní odezvou okénkovou metodou. Tyto filtry se snadno navrhují, implementují a disponují velkou stabilitou, kdy nehrozí rozkmitání. Jedná se o digitální filtry, které filtrují již digitalizovaný signál v posloupnosti vzorků a obsahují konečný počet hodnot po vybuzení filtru impulsem. **[9]** FIR filtr délky ***M*** se vstupem ***x(n)*** a výstupem ***y(n)*** lze popsat pomocí rozdílové rovnice:

$$\boldsymbol{y(n)} = \sum_{\boldsymbol{k=0}}^{\boldsymbol{M-1}} \boldsymbol{b(k)x(n-k)} \tag{8}$$

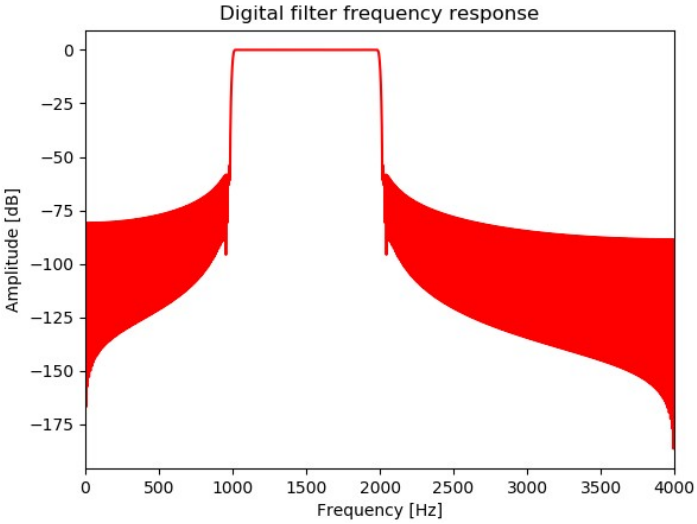
Kdy v předchozím vzorci ***b(k)*** jsou filtrační koeficienty a filtr používá aktuální vstup ***x(n)*** a zpožděné vstupní vzorky ***x(n-k)***. Filtr však nepoužívá zpožděné výstupní vzorky (tj. ***y(n-k)***). **[10]** Právě aplikací knihovny Scipy a aplikací digitálních filtrů na včelí data se bude zabývat praktická část práce.

Pro názornost problematiky filtr na obrázku 7 propouští digitální signál nižších frekvencí a to přesněji signál 0 – 1000Hz. Na obrázku je zřetelné, že amplituda signálu do 1000Hz bude nepozměněna a hodnotě 1000Hz náleží mezní frekvence, při které zisk poklesne a signál v pokračujícím frekvenčním rozsahu bude utlumen. Filtr je vhodné navrhnout tak, aby signál byl utlumen o **3dB** v mezní frekvenci. Poté s rostoucím útlumem se nachází v přechodové části, což je část, kdy filtr ztrácí na zisku až do rozkmitané části, kterou nazýváme nepropustným pásmem. **[10]** Obdobným způsobem se chovají filtry na obrázku 8 a 9, pouze v případě pásmové propusti je propuštěné navržené pásmo a v případě horní jsou propuštěny signály vyšších frekvencí.

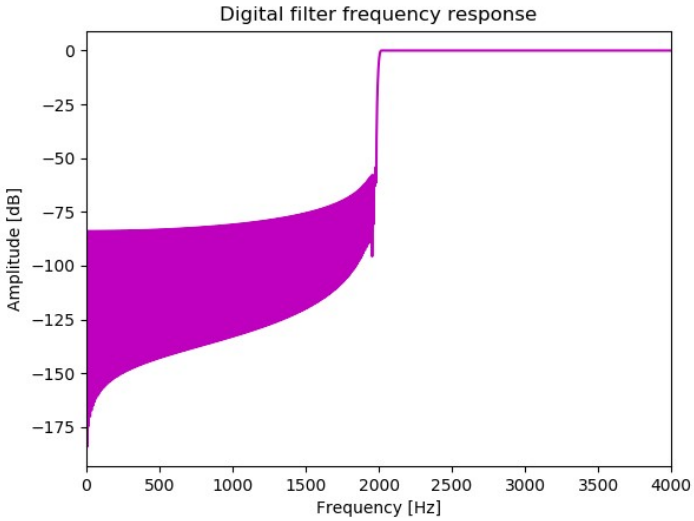
Rozsah frekvenční složky, pro kterou je možno filtry navrhovat, se odvíjí od vzorkovací frekvence, kdy tento rozsah odpovídá polovině vzorkovací frekvence, což vyplývá ze Shannonova teorému (při vzorkovací frekvenci 8000 vzorků za vteřinu je tedy možno navrhovat filtry do 4000Hz).



Obrázek 7 - Dolní propust



Obrázek 8 - Pásmová propust



Obrázek 9 - Horní propust

3. 3 Ukládání výsledků analýzy a vizualizace

Tato část práce se zabývá popisem prostředků, které je možné využít k vizualizaci nebo ukládání výsledků analýzy včelího bzukotu. Praktickým využitím Grafany, InfluxDB a vytvářením JSON záznamů se bude dále zabývat praktická část práce.

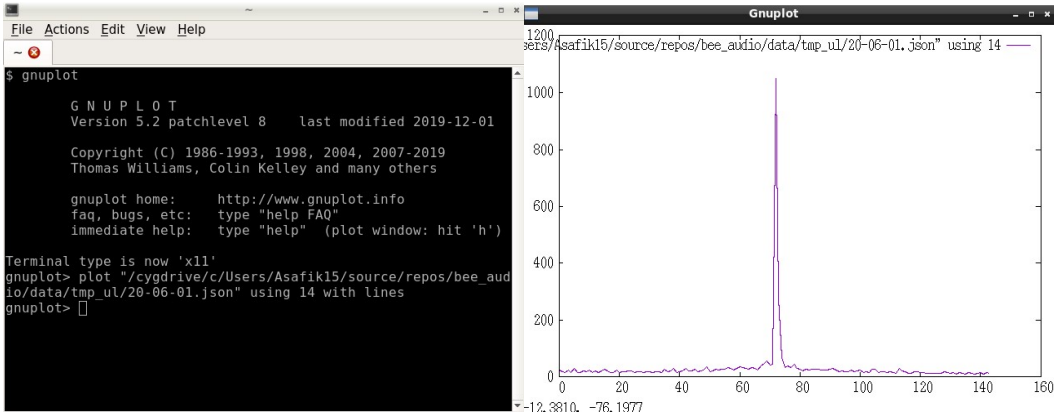
3. 3. 1 JSON

JSON (Java Script Object Notation) je odlehčený formát pro výměnu dat. Je snadné jej generovat a analyzovat. Jedná se o textový formát, který není závislý na programovacím jazyku, ale využívá známé programátorské návyky z rodiny jazyků C, Java, JavaScript, Perl, Python atd. Nejrozšířenější formát JSON souboru je formát, kdy objekt začíná levou složenou závorkou a končí pravou složenou závorkou. Uvnitř závorek je několik párů obsahující klíč a hodnotu, mezi kterými je dvojtečka, a páry jsou oddělené čárkou. **[12]** Tento formát lze přirovnávat k Python slovníku (dictionary). JSON může být jakýmsi mezikrokem mezi analýzou dat a ukládáním jejích výsledku do databáze. Navíc právě JSON je vhodnou zálohou databáze a může sloužit i pro kontrolu výsledků pomocí GNUPLOT.

Příklad JSON záznamu: *{"a": 10, "b":20, "c":30}*

3. 3. 2 GNUPLOT (Cygwin)

GNUPLOT je interaktivní nástroj pro vykreslování funkcí řízený příkazovým řádkem pro UNIX. Jedná se o grafický volně distribuovaný program, který umožňuje vizualizovat matematické funkce a data. **[13]** JSON data lze tedy snadno pro ověření výsledků nebo jednoduchou vizualizaci vykreslit pomocí knihovny GNUPLOT. Je zapotřebí mít nainstalován Cygwin a jemu náležící knihovny pro GNUPLOT a X11 (xorg-server a xinit). GNUPLOT vyžaduje GUI, a proto je nutná i tato implementace systému X window verze 11 (X11) v Cygwinu. **[14]** Na obrázku 10 je názorná vizualizace JSON dat pomocí GNUPLOT v Cygwinu.



Obrázek 10 - Příkaz pro vykreslení a vizualizace v GNUPLOT

Příkaz na obrázku 10 znázorňuje spuštění GNUPLOT a vykreslení 14. sloupce JSON dat nadefinovanou cestou.

3. 3. 3 InfluxDB

InfluxDB je databáze, do které je možné zapisovat pomocí Pythonu a je označována jako open-source time series database. Je zkrátka vhodná pro ukládání a správu časových sérií. Databáze je bez schématu a lze do ní vkládat libovolné pravidelné i nepravidelné časové řady. **[15]** Navíc Influx je již součástí včelího LINUX serveru, kam se ukládají zvukové záznamy. Formát InfluxDB linkového protokolu pro zápis bodů do databáze popisuje obrázek 11.

Obrázek 11 - Seznam jednotlivých položek linkového protokolu InfluxDB

Linkový protokol pro zápis bodů na obrázku 11 se skládá z:

- **measurement** - Název měření do kterého budou zapsána data.
- **tag_set** - Značka, která může být součástí datového bodu (nepovinné).
- **field_set** - Pole ve formátu <klíč> = <hodnota>
- **timestamp** - Časové razítko (UNIX timestamp) **[16]**

InfluxDB se může skládat z libovolného počtu databází a databáze může obsahovat několik měření, kam lze zapisovat body. Při zapsání bodu se časové razítko přiřadí automaticky v závislosti na tom, kdy byl bod zapsán, popřípadě ho lze samostatně definovat a bodu přiřadit námi vyhovující čas. **[17]**

V případě, že již máme nainstalovaný InfluxDB popřípadě vytvořené už určité databáze, můžeme využít příkazového řádku k příkazům a dotazům, popřípadě vše dělat prostřednictvím Pythonu. Názorná ukázka využití příkazového řádku InfluxDB na obrázku 12 a 13.

```
halape00@fprvpcskrbek:~$ influx
Visit https://enterprise.influxdata.com to register for updates, InfluxDB server
 management, and monitoring.
Connected to http://localhost:8086 version 1.1.1
InfluxDB shell version: 1.1.1
> show databases
name: databases
name
----
_internal
bee
bee_audio
bee_annotations
```

Obrázek 12 - Příklad využití příkazového řádku InfluxDB

```
> use bee_audio
Using database bee_audio
> select * from ull where time>=1601047932000000000
name: ull
time                 DP_100Hz  HP_2000Hz  PP_100-200Hz  PP_200-350Hz  PP_350-500Hz  PP_500-2000Hz  Puvodni_zaznam
----                 --------  ---------  ------------  ------------  ------------  -------------  --------------
1601047932000000000  81.58     8.41       40.19         43.16         25.53         20.75          110.68
1601048532000000000  49.85     3.78       9.08          12.06         8.71          8.5            54.13
1601049132000000000  50.6      4.59       24.61         20.24         13.74         12.03          63.75
```

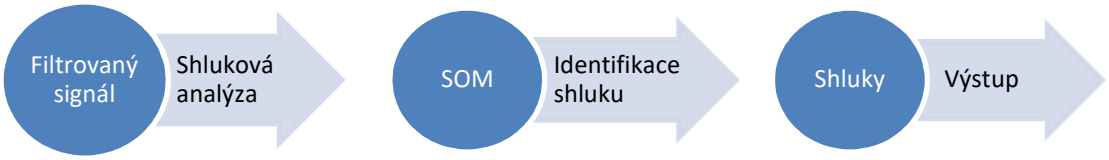
Obrázek 13 - Zobrazení dat v příkazovém řádku InfluxDB

3. 3. 4 Grafana

Grafana je multiplatformní open-source webová aplikace, umožnující vizualizaci a analýzu dat. **[18]** Po propojení s podporovaným zdrojem dat (např.: databází InfluxDB) je Grafana schopná tyto data v různých formách prezentovat, generovat výstrahy, zobrazovat anotace atd. V případě běžné základní vizualizace dat postačí vytvořit dashboard, do kterého se postupně přidávají panely určené pro vizualizaci. Detailnějším popisem k vizualizaci dat se zabývá praktická část.

Obrázek 14 - Vizualizace filtrovaných dat v Grafaně

3. 4 Shluková analýza zvukových signálů



Obrázek 15 - Proces popisující postup shlukové analýzy filtrovaných signálů

Využití shlukové analýzy je široké. Pro nás je ale důležité, že jí lze aplikovat i na zvuková data pro jejich klasifikaci a následné rozdělení dat do tříd. Pro případ shlukové analýzy, při jejímž učení se hledají nejmenší hodnoty vah mezi neuronovou sítí a vytěžovanými daty **[20]**, je typické učení bez učitele, pro nějž známe vstup, ale výstup nám je neznámý. **[21]**

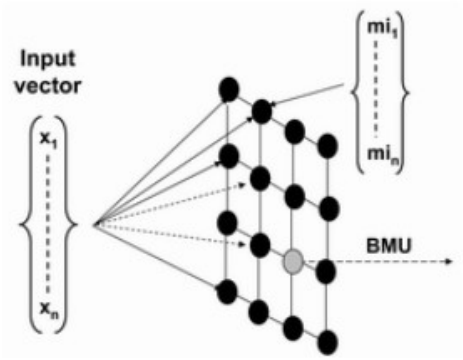
3. 4. 1 SOM

SOM (KOHONENOVA SAMOORGANIZAČNÍ MAPA) je umělá neuronová síť, jejíž vlastností je učení bez učitele, pomocí které je vygenerována dvou rozměrná mapa (může být i více rozměrná) reprezentující vstupní prostor. **[22]** Pomocí SOM je prováděna shluková analýza, při které jsou shlukovány podobná data, a následná vizualizace například U maticí, která znázorňuje průměrnou vzdálenost vah od sousedních neuronů. Neurony s podobnou váhou vytvářejí shluk a neurony s váhou odlišnou tyto shluky oddělují. Uspořádání neuronů při SOM může být lineárního nebo hexagonálního tvaru, kdy jsou mezi sebou propojeny všechny sousední neurony a jejich vzdálenost je ve všech směrech 1. Dalším uspořádáním při SOM je mřížka čtvercového, obdélníkového nebo toroidního tvaru, kdy jsou mezi sebou propojeny neurony pouze ve svislých a vodorovných osách a vzdálenost mezi neurony je 1 při těchto osách a při úhlopříčce odmocnina ze dvou, což odpovídá Pythagorově větě. (při toroidní mřížce jsou propojeny i protilehlé neurony). **[19]**, **[20]**, **[23]**

Eukleidovská vzdálenost (metrika) mezi neurony je dána vztahem:

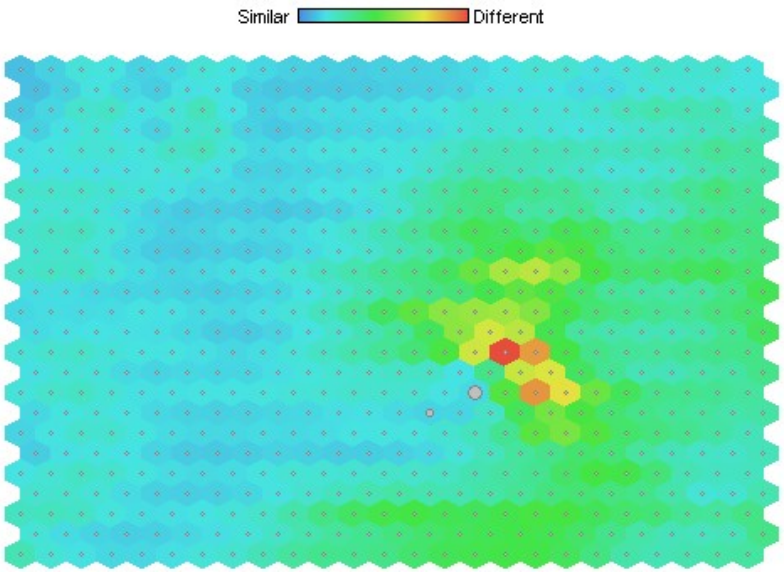
$$\boldsymbol{D_E\,(x_1, x_2) = \sqrt{\sum_{i=1}^{n} (x_{1_i} - x_{2_i})^2}} \tag{9}$$

Učení této neuronové sítě probíhá tak, že se inicializují jednotlivé váhy mezi vstupní a výstupní vrstvou a nastaví se parametr učení a poloměru sousedství. Následně na vstup sítě přiložíme trénovací množinu a nejpodobnější neuron danému vzoru se stane **BMU** (best matching unit). Stane se tedy vítězem a jeho váhy a váhy sousedů budou aktualizovány, což určuje síla změny vah, parametr alfa, který může nabývat hodnoty 0-1. Na začátku učení se parametr alfa blíží hodnotě 1 a na konci učení hodnotě 0. Učení probíhá iterativně a postupně se zpracovává každý vzor z trénovací množiny. **[23]**



Obrázek 16 - Průběh učení SOM neuronové sítě [42]

Na obrázku 17 je neuronová síť SOM vizualizovaná U maticí. Každému neuronu obsaženému v mřížce náleží v jeho vnitřku kružnice, jejíž obsah znázorňuje četnost vstupních vzorů, pro které se stal daný neuron vítězem. Barva neuronu udává podobnost svým sousedům. Modrá barva značí, že se jedná o blízké neurony, a vytváří tedy shluk. Barva zelená už tento shluk odděluje, a neuron je tedy svým sousedům více vzdálený. Červený neuron značí už velice vzdálený neuron od svých sousedů.



Obrázek 17 - Neuronová síť SOM vizualizovaná U maticí

3. 4. 2 Clustering

Jedná se o shlukovou analýzu a to algoritmus **K-means**, který rozdělí data do tříd (shluků) na základě jejich vlastností. Na počátků provádění algoritmu je nutno stanovit počet shluků (clusterů), kterých chci dosáhnout, přičemž jejich počet musí být menší než počet vstupních objektů. Každému shluku bude náležet jeden vygenerovaný bod,

který bude středem shluku. Objekt bude zařazen do shluku, jemuž daný centroid (střed) je nejblíže svou Eukleidovskou vzdáleností. Centroidy se budou přepočítávat tak, aby byly těžištěm pro všechny body náležící shluku až do absolutního ustálení v závislosti na maximálním počtu kol přepočítávání středu. **[24]**

3. 4. 3 RapidMiner

RapidMiner je grafický nástroj pro těžení dat. **[25]** Využívá se pro komerční i studijní účely. Je vybaven prostředím pro přípravu dat, strojové i hluboké učení, prediktivní analýzu a těžbu textu. Krom učení je vybaven i možností vizualizace výsledků. **[26]** RapidMiner Studio Educational, který slouží pro studijní účely, je po registraci možno bezplatně užívat po dobu jednoho roku. RapidMiner disponuje velkým množstvím operátorů, které aplikujeme na vstupní soubor a využijeme při návrhu výsledného řešení. Operátory v návrhu řadíme za sebou a algoritmicky se zpracovávají až k požadovanému řešení. Konkrétně současná verze nabízí v základní nabídce více než 400 operátorů sloužících pro předzpracování dat (př.: standardizace, filtrace…), modelování dat (př.: clustering, rozhodovací stromy…), ověřování (př.: porovnávání…), přesnosti modelů dat atd. **[27]** RapidMiner dále obsahuje Marketplace, kde je možno získat zdarma další požadované operátory, které nejsou ve standardní nabídce.

4 Rešerše obdobných technologií

Obdobných řešení jsem v této problematice našel několik (ne však tak propracovaných). Za zmínku ale stojí projekt **BuzzBox Mini**, který za cenu 200 amerických dolarů je schopen měřit hned několik veličin v úlu. Zařízení po připojení k WiFi a využití aplikace **OSBeehives - Digital Beekeeping Toolkit** (dostupné zdarma na App Store i Google play) je schopno uživateli provádět zvukovou analýzu, měřit vnitřní i venkovní vlhkost a teplotu v úlu, varuje při nástupu špatného počasí v rozmezí 0-5 dnů, krádeži a je opatřeno možností solárního nabíjení. Výrobce dále uvádí stupeň krytí IP68 (absolutně odolné vůči prachu a vniknutí vody při trvalém ponoření). **[28]**, **[29]**

Measurements		
Audio Recordings	20 Hz	3150 Hz
Hive Temperature	0 °C	65 °C
Temperature Accuracy	0.2 °C	
Hive Humidity	0%	100%
Humidity Accuracy	2%	
External Temperature	Local weather station dependent	
External Humidity	Local weather station dependent	
Atmospheric Pressure	Local weather station dependent	
Wind Speed	Local weather station dependent	
Wind Direction	Local weather station dependent	
Weather Forecast	0 Day	5 Day
Geolocation	Global	

Tabulka 2 – Technické specifikace BuzzBox Mini [43]

Dle tabulky 2 zařízení zachytí záznamy v rozsahu 20 - 3150Hz, což je sice pro zvukové chování včel dostačující, ale mé řešení je schopno při vzorkovací frekvenci 8000 vzorků za vteřinu zachytit pásmo přibližně s přihlédnutím ke zvukové kartě 20 - 4000Hz (Rozsah zachycujícího frekvenčního rozsahu $= \frac{\boldsymbol{vzorkovací\ frekvence}}{\mathbf{2}}$). Mé řešení navíc

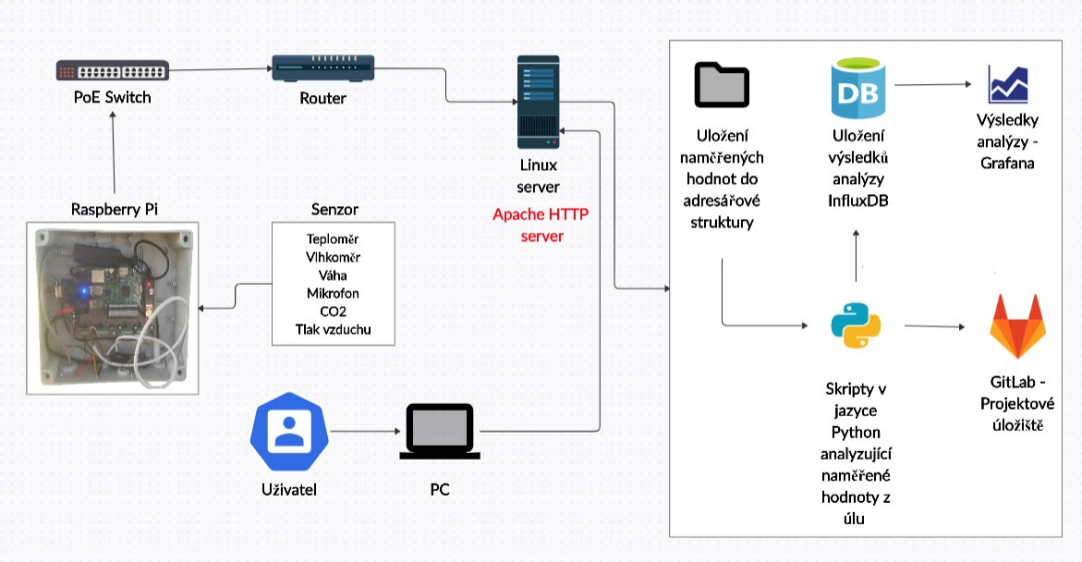
poskytuje krom vizualizace i hlubší náhled pro určité kritické momenty týkajících se dílčích záznamů, které uživatel odhalí v Grafaně, a možnost stálého ukládání výsledků filtrační analýzy a tím pádem i rozboru starších dat. Dalším důležitým aspektem je, že mé řešení je navíc opatřeno filtry kmitočtů, které jednoznačně odhalí velikost amplitudy při náležitém frekvenčním rozsahu oproti řešení BuzzBox Mini, které provádí pouze jednoduchou zvukovou analýzu. Navíc BuzzBox Mini zaznamenává audio záznamy ve větších časových rozmezích (a to každých 15 minut - 2 hodiny v závislosti na stavu baterie). Zajímavá myšlenka tohoto projektu je aplikace strojového učení a přidružení zdravotních indikátorů včelích úlů k zvukovým vzorům s cílem vytvořit robustní systém včasné detekce zdraví včel. **[30]** Tato myšlenka však dle nepotvrzených zdrojů nebyla doposud dokončena. Instalace BuzzBox Mini je ale jednoduchá a pro běžné uživatele celkem příznivá.

Obrázek 18 - Popis zařízení BuzzBox [44]

Obrázek 19 - Zařízení BuzzBox [45]

5 Záznam a zpracování zvukových signálů

Nahrávání akustických dat uvnitř úlu, následné vytváření zvukových digitálních signálů ve formátu **WAV** a jejich ukládání na server do adresářové struktury vychází z diplomové práce Mgr. Lukáše Širhala, který již zkonstruoval řešení pro zpracování audio záznamů serverem. Na tyto záznamy je poté možno aplikovat další analýzu, čímž se zabývá tato bakalářská práce. Na obrázku 20 je vyobrazen model popisující komunikaci úlu se serverem a následnou analýzu.



Obrázek 20 – Model popisující komunikaci úlu se serverem a následnou analýzu

V případě zvuku bude **elektretový mikrofon MCE103**, jehož frekvenční rozsah je **20Hz - 10kHz**, čekat na výzvu základní desky **Raspberry Pi**, na kterou je skrz rozhraní USB zvukové karty **Axagon ADA-17 mini HQ** připojen. Audio záznam se zpracuje jako binární soubor, odešle na server k dalšímu zpracování a uloží ve formátu WAV. **Raspberry Pi** zaznamená včelí bzukot každých deset minut (automatické spuštění Python skriptu Cronem, jenž byl vytvořen v diplomové práci Lukáše Širhala) a na serveru bude uložen digitalizovaný záznam pomocí Python modulu **pyAudio [11]** s následující specifikací:

- **Délka záznamu:** 10 vteřin
- **Interval mezi záznamy:** 10 minut
- **Vzorkovací frekvence:** 8000 vzorků za vteřinu
- **Název souboru:** “YY-MM-DD_hh_mm_ss.wav” (datum a čas vzniku souboru)

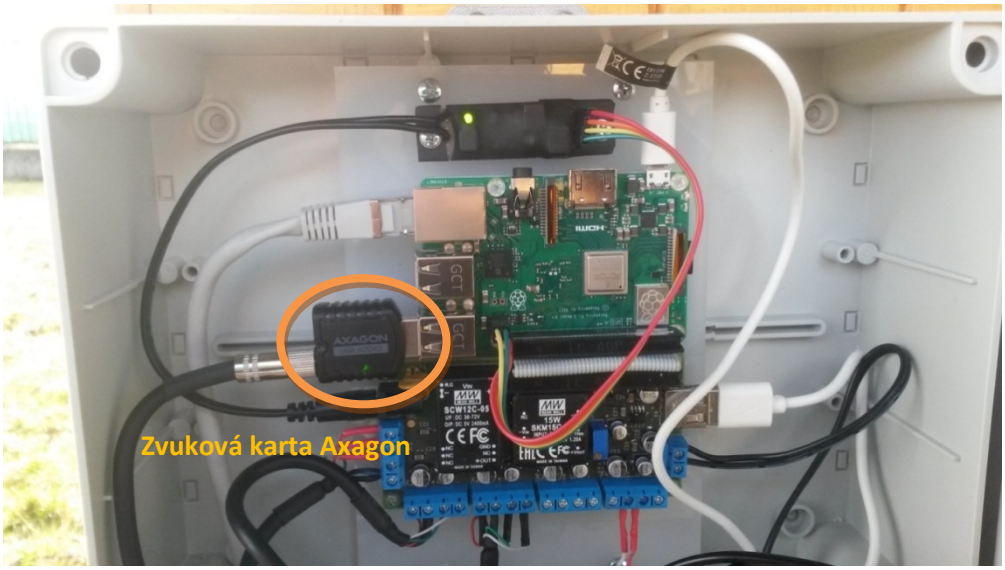
- **Kanály:** 1
- **Bitová hloubka:** 16 bitů (určuje počet hladin ve dvojkové soustavě, kvantování)
- **Přibližná datová velikost:** 157kB

$$\boldsymbol{Velikost[kB]} = \boldsymbol{vzorkovací\ frekvece} \cdot \boldsymbol{bitová\ hloubka} \cdot \boldsymbol{počet\ kanálů} \cdot \boldsymbol{délka\ záznamu} : \mathbf{8} : \mathbf{1024} \quad (10)$$

- **Místo uložení na serveru: úl1:** /mnt/storage/vcely/data/ul1

  úl2: /mnt/storage/vcely/data/ul2

  úl3: /mnt/storage/vcely/data/ul3

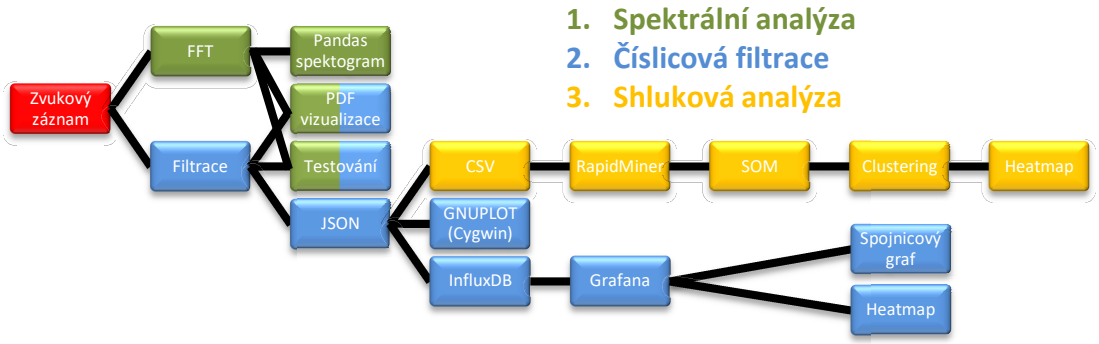


Obrázek 21 – Náhled na Raspberry Pi umístěném na inteligentním včelím úlu

6 Metodika

Nahrávání audio dat vně úlu a jejich následnou digitalizaci serverem, na což navazuje tato práce analýzou těchto dat, popisuje kapitola ***5***. Právě uvnitř úlu veškerý proces začíná tím, že záznamové zařízení, které zaznamenává včelí aktivitu každých deset minut, odesílá v součinnosti s Raspberry Pi 10ti vteřinový záznam na **LINUX server**, kde se digitalizuje. Pokud nedojde k výpadku, lze předpokládat, že za den bude zaznamenáno 144 záznamů (6 záznamu/hodina * 24). To je velké množství dat a je tedy nutné navrhnout takové programové vybavení, které bude schopno s těmito daty pracovat a ukládat výsledky filtrační analýzy průběžně i dávkově do databáze pro následnou vizualizaci a možnost odhalit kritické momenty, pro které bude umožněn rozbor pomocí modulů jazyku Python.

Právě pro tuto problematiku se mi jevilo jako nejvhodnější zvolit programovací jazyk **Python**, jelikož disponuje vhodnými knihovnami pro analýzu dat. Využité knihovny, kterými lze aplikovat veškeré kroky uvedené na obrázku 22, jsou popsané v kapitole *7*.



Obrázek 22 - Postup popisující zpracování zvukových záznamů

Obrázek 22 znázorňuje zpracování audio dat určitými postupy. Na zvukový záznam je nutné aplikovat rychlou Fourierovu transformaci a umožnit pro ni vizualizaci za účelem dalšího zpracování a představy, jak zvukové záznamy z úlu ve frekvenční složce vypadají. Právě Fourierova transformace a zvukové projevy včelstva (kapitola ***3. 1***) nám napomůžou k dalšímu řešení zvukové analýzy a to pomocí číslicových filtrů, jejich správného zvolení a přípravy.

Po dokončeném návrhu filtrů a jejich aplikaci na zvuková data v časové doméně bude umožněno ukládat RMS filtrovaných záznamů do podoby JSON s následnou možností zápisu JSON objektů dávkově pomocí Cronu i průběžně do databáze InfluxDB.

Pro data z databáze InfluxDB bude umožněna vizualizace pomocí aplikace Grafana, která bude schopna tyto data vykreslovat jako spojnicový graf nebo teplotní mapu.

Pro možnost rozboru dílčích záznamů bude umožněno vytvářet tištitelné soubory (PDF) pro sledování aplikace filtrů v časové doméně a FFT ve frekvenční doméně. Pro FFT bude navíc umožněn rozbor jednoho kritického dne pomocí spektrogramu v Pandas.

Zda programové vybavení zabývající se číslicovou filtrací a FFT pracuje správně, je nutno testovat a ověřit správnost výsledků. Proto je nutné vygenerovat vlastní sinusové signály disponující zvolenými frekvencemi a sledovat, zda programové vybavení pracuje správně a tyto signály správně analyzuje.

JSON záznam bude umožněno převést i do podoby CSV pro následnou shlukovou analýzu pomocí RapidMineru, který formát CSV vyžaduje pro vložení vstupních trénovacích dat.

Postup řešení:

1. Seznámení se zvukovými záznamy za období provozu a jejich ukládání na server.
2. Volba vhodného programovacího jazyka a softwarového vybavení.
3. Navrhnout program pro vizualizaci FFT dílčích včelích záznamů.
4. Návrh digitálních filtrů.
5. Aplikace digitálních filtrů na zvuková data v časové složce a výpočet jejich RMS.
6. Ukládání RMS filtrovaných signálů – průběžně i dávkově.
7. Vizualizace výsledků.
8. Testování.
9. Volba vhodného software pro shlukovou analýzu.
10. Aplikace shlukové analýzy na filtrované signály pomocí vhodného software.
11. Mapování výsledků aplikace shlukové analýzy na filtrovaná zvuková data.

7 Využité programové vybavení

7. 1 Operační systém

Operační systém serveru, na kterém se veškerá zvuková, ale i další data zpracovávají, je open-source Linuxová distribuce Ubuntu. Konkrétně **Ubuntu 20.04.1 LTS** (GNU/Linux 5.4.0-52-generic x86_64). Tento systém je spravován školitelem práce a každý z členů, který pracuje na včelí analýze, má zde přidělen svůj pracovní prostor.

U systému se pro možnost spuštění skriptů vzniklých za účelem této práce předpokládá, že bude vybaven programovacím jazykem Python, správcem balíčků pip a knihovnami pro Python popsaných v kapitole ***7. 2. 1***.

K serveru se lze připojit pouze z místní sítě JČU, popřípadě po připojení k **VPN** koncentrátoru JČU. Pak se k němu lze přihlásit například přes PUTTY na portu 22 (SSH) pomocí IP adresy **160.217.213.118**. Dalším využitým prostředkem k připojení může být například WinSCP či Cygwin. Mluvíme ve všech případech o připojení k Linux serveru z operačního systému Windows 10.

7. 2 Programovací jazyk

Pro akustickou analýzu jsem zvolil jako vhodný programovací jazyk Python (konkrétně **Python 3. 8. 5**). Hlavním důvodem bylo, že jazyk disponuje velkou řadou knihoven vhodných pro statistickou analýzu. Všechny skripty pro praktickou část této bakalářské práce vznikly v tomto programovacím jazyce.

7. 2. 1 Knihovny

Ve zdrojových kódech zvukové analýzy bylo pro maximální efektivitu využito Python knihoven (a jim náležících funkcí):

- **scipy** - Matematika, věda a technické záležitosti. **[31]** Vhodné pro návrh a aplikaci číslicových filtrů.
- **matplotlib** - Tvorba statistických, animovaných a interaktivních vizualizací. **[32]**

- **numpy** - Matematická správa matic, Fourierova transformace a další matematické funkce. **[33]**
- **pandas** - Práce s datovými rámci, další analýzu a manipulaci s daty. Snadná tvorba tabulek a možnost ukládání do formátu CSV.
- **math** - Matematické funkce.
- **os** - Poskytuje funkce pro interakci s operačním systémem. **[34]**
- **sys** - Dává informace o konstantách, funkcích a metodách interpreta. **[35]**
- **time** - Knihovna pro práci s časovými hodnotami. Její funkce time() vrací číslo s plovoucí desetinnou čárkou, které reprezentuje počet vteřin, jež uběhly od 1. 1. 1970 (UTC), což lze v Linuxu využít jako časové razítko (samozřejmě bez desetinné čárky a po nastavení časové přesnosti – time precision).
- **datetime** - Pro práci s daty a časy. Pro uživatele je tato knihovna přívětivější než time pro svou jednoduchost.
- **progress** - Cyklus, který v příkazovém řádku znázorňuje svůj průběh.
- **argparse** - Parsování parametrů. Po vybavení programu argparserem je možné skript spouštět z příkazové řádky společně s argumenty spustitelného skriptu.
- **json** - Knihovna pro práci se soubory ve formátu JSON.
- **influxdb** – Knihovna pro ukládání a správu časových sérií (databází).

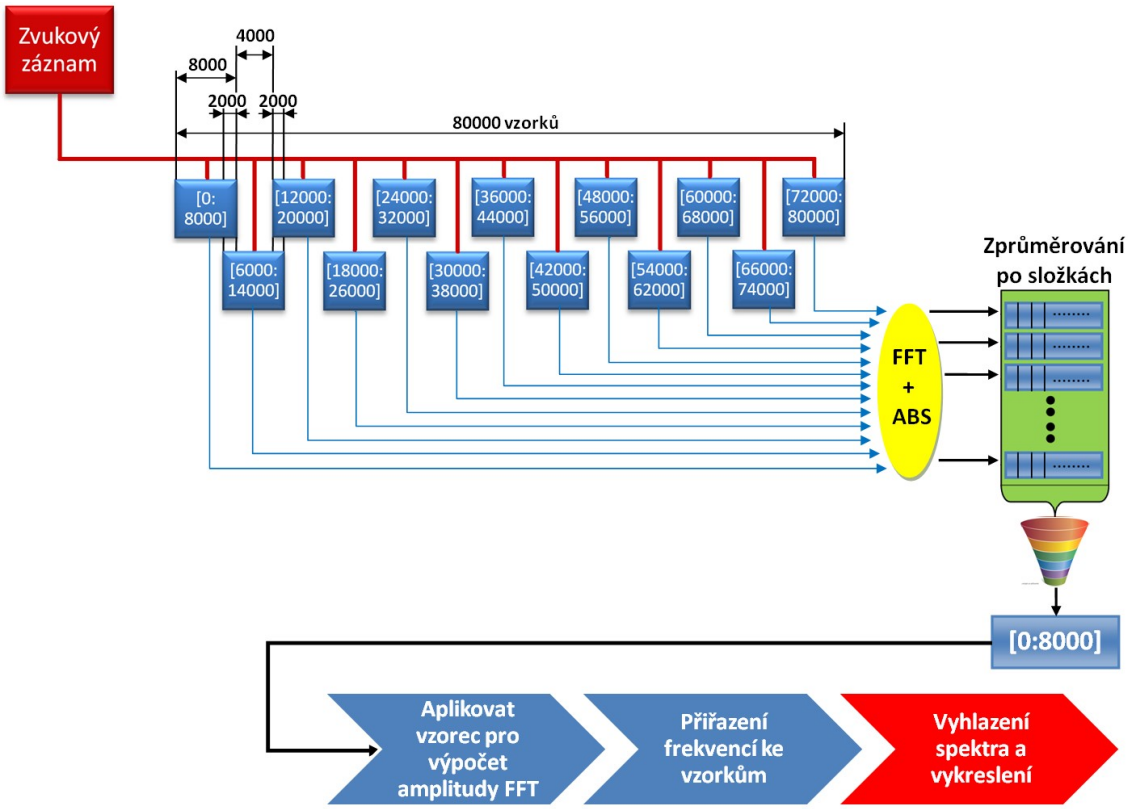
Zároveň veškeré zdrojové kódy (kvůli nevyžádané duplicitě a snadnější editaci) byly napsány tak, aby disponovaly možností importovat zdrojové kódy jako modul.

8 Praktická část

8. 1 Spektrální analýza

Včelí akustická data se skládají z většího množství signálů o různých frekvencích a účelem spektrální analýzy je možnost sledovat tyto signály ve frekvenční složce namísto složky časové.

Skript **fft_spectogram.py** byl pro tuto práci vytvořený za účelem sledování spektra včelího akustického záznamu. Po spuštění tohoto skriptu a volbě vhodných parametrů zobrazí buďto spektrální analýzu jednoho konkrétního záznamu, což popisuje obrázek 23, nebo vytvoří spektrogram v Pandas skládající se ze spektrálních analýz dílčích záznamů jednoho dne.



Obrázek 23 – Spektrální analýza akustického signálu

8. 1. 1 Fourierova transformace

Právě Fourierova transformace je prostředek, který nám poskytne možnost sledovat signál ve frekvenční doméně a poskytne frekvenční spektrum včelích akustických dat.

Skript **fft_spectogram.py**, který se stará o vizualizaci FFT (rychlé Fourierovy transformace) dle obrázku 23 nejdříve načte vstupní zvukový záznam pomocí knihovny **scipy** a její funkce - **scipy.io.wavfile.read(filename = vstupní soubor ve formátu wav),** která vrací vzorkovací frekvenci záznamu a data vstupního souboru v podobě **numpy array** (pole ve formátu numpy).

Následně je pole obsahující data vstupního **WAV** souboru rozsekáno na 13 polí. Kvůli rozsekávání pole se předpokládá, že vstupní soubor bude disponovat vzorkovací frekvencí 8000 vzorků za vteřinu a délkou záznamu 10 vteřin (tedy přesněji 80ti tisíci vzorky). V opačném případě bude program ukončen po vyhození výjimky. Data vstupního souboru budou rozsekány po 8000 vzorcích s překryvy 2000 vzorků, což napomůže k eliminaci šumu a nevyžádaných výchylek FFT (př.: Subarray1[0:8000], Subarray2[6000:14000] atd.)

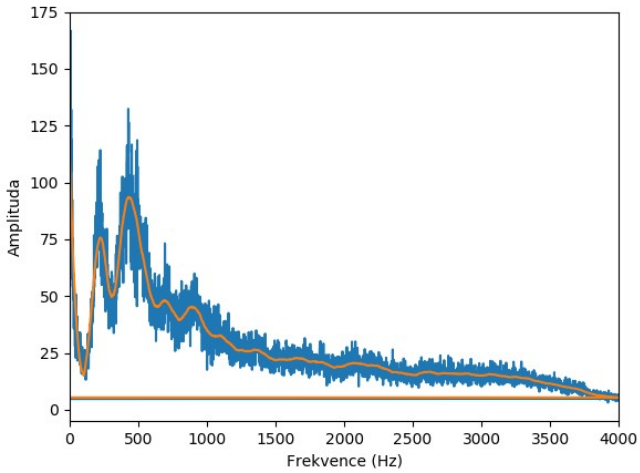
Nyní bude na každé z těchto polí aplikováno FFT, konkrétně funkce **numpy.fft.fft(a = vstupní pole)**. Jednotlivá pole je nutno následně po aplikaci FFT převést do podoby absolutní hodnoty, jelikož pole disponují komplexními čísly (**numpy.absolute(a = vstupní pole)**). Poté se iterativně berou prvky všech rozsekaných polí a z těchto prvků se vypočte průměr. Z těchto průměrů je vytvořeno jedno finální pole (zkrácené kvůli zprůměrování) o délce 8000 prvků reprezentující FFT. Po zisku výsledného FFT pole (které bylo kvůli rozsekání zkráceno) aplikujeme vzorec pro výpočet amplitudy jednotlivých prvků tohoto pole (viz. kapitola ***3. 2. 4 – rovnice (7)***).

Posledním krokem je přiřazení frekvencí ke vzorkům. K tomu napomůže funkce **numpy.fft.fftfreq(n = délka okna, d = rozteč vzorků)**. Délka okna odpovídá délce výsledného FFT pole (přesněji 8000, protože pole bylo kvůli rozsekání zkráceno) a rozteč vzorků 1/8000 odpovídá převrácené hodnotě ke vzorkovací frekvenci.

Poté stačí spektrální analýzu signálu pomocí FFT vykreslit knihovnou **matplotlib**. Konkrétně pomocí **matplot.pyplot.plot(x, y)** stanovit osu ***x***, která odpovídá přiřazeným frekvencím ke vzorkům a stanovit osu ***y***, která naopak odpovídá poli s vypočtenými amplitudami jednotlivých prvků FFT. V samém závěru postačí využít funkci **matplot.pyplot.show()** pro vykreslení spektrální analýzy.

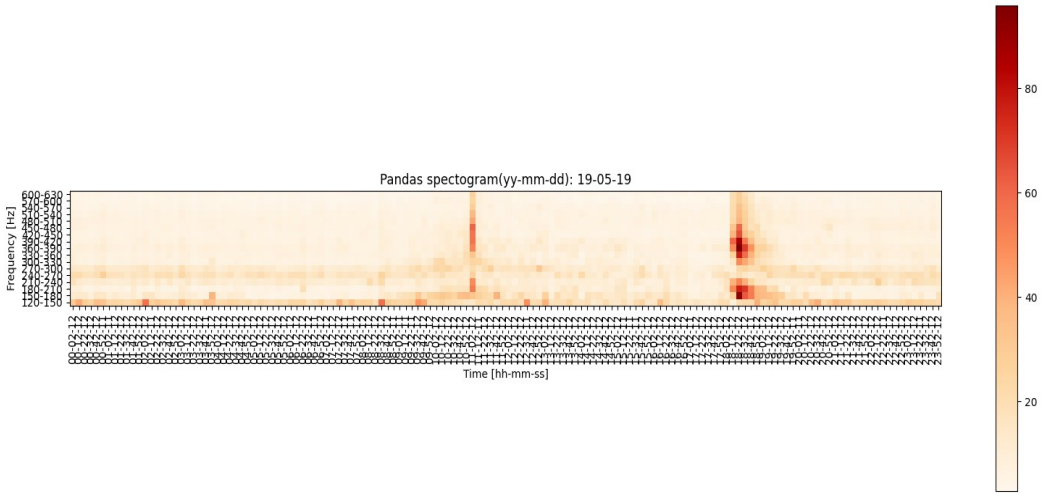
8. 1. 2 Vyhlazení spektra

Za účelem větší přehlednosti vizualizace již aplikované Fourierovy transformace byl využit **Savitzky - Golay filtr**. Tento filtr byl na FFT aplikován za účelem vyhlazení spektra a větší přehlednosti, aniž by byla narušena charakteristická vlastnost transformace. Na rychle měnící se signály (např.: testovací signály v kapitole ***testování***) není filtr vhodné aplikovat **[36]**. Pro tuto problematiku byla zvolena funkce **scipy.signal.savgol_filter(x=vstupní pole, w=délka okna, p=pořadí polynomu)**, která se stará ve skriptu **fft_spectogram.py** o vyhlazení spektra signálu. Filtr byl využit pouze za účelem přehlednější vizualizace FFT.



Obrázek 24 - Rychlá Fourierova transformace společně s aplikací Savitzky-Golay filtru

8. 1. 3 Spektrogram



Obrázek 25 - Pandas spektrogram

Na obrázku 25 je názorná implementace FFT do zobrazení knihovnou **Pandas** a konkrétně jeho dataframem. Spektrogram, který vytváří skript **fft_spectogram.py**, po volbě vhodných parametrů vytvoří dataframe, jenž se skládá ze všech FFT dílčích záznámů konkrétního dne zvoleného uživatelem a vykreslí velikost amplitudy náležící danému záznamu a rozsahu. Spektrogram se zabývá pásmem **FFT 120 - 630Hz**, kdy každé z polí dataframe náleží rozsahu 30Hz (př.: první řádek spektrogramu odpovídá rozsahu 120 - 150Hz). Pásmo 120 - 630Hz bylo zvoleno proto, že pro včely a jejich zvukové projevy je charakteristické (viz. Úvod). Velikost amplitudy náležící jednomu poli byl spočítán prostým aritmetickým průměrem daného rozsahu ve FFT, jelikož účelem spektrogramu bylo dát jednoduchý náhled na jeden specifický kritický den a poukázat na proměny chování včelstva.

Dle odborného pozorování obrázku 25 odpovída rojení včelstva čas kolem 11té hodiny dopolední a návrat velkého množství včel zpět do úlu času kolem 6té hodiny večerní. Spektrogram na tyto kritické momenty poskytne celkem přehledný a jednoduchý náhled. O zobrazení spektrogramu v Pandas se stejně jako o FFT stará Python modul **matplotlib**.

8. 2 Číslicová filtrace

Pro digitalizované záznamy z včelích úlů je nyní nutné navrhnout číslicové filtry dolní, horní a pásmových propustí. Aplikace těchto navržených filtrů na audio data je v práci využito za účelem náhledu na velikost amplitudy v daném frekvenčním pásmu. Rozsahy filtrů jsou navrženy dle zvukových projevů včelstva (kapitola ***3. 1***) a přezkumu záznamů z inteligentního včelího úlu ve frekvenční složce pomocí FFT. Filtry v mé práci jsou navrženy pomocí knihovních funkcí v Pythonu a následně vykresleny knihovnou **matplotlib**. Návrh číslicového filtru obstará funkce **scipy.signal.firwin(f_len, cut_off, pass_zero)**, kdy ***f_len*** je velikost okna filtru, ***cut_off*** místo, kde filtr ořízne počátek popřípadě konec filtru a ***pass_zero*** umožňuje volbu propusti či zádrže. O aplikaci navrženého filtru v časové složce se stará funkce **scipy.signal.lfilter(b, a, x)**, kdy ***b*** jsou vstupní koeficienty filtru, tedy koeficienty FIR filtru, které vrací funkce **firwin**, ***a*** je vektor koeficientu jmenovatele v 1-D sekvenci (v našem případě tedy [1.0]) a ***x*** je vstupní signál, na který se aplikuje číslicový filtr.

8. 2. 1 Návrh číslicových filtrů

Na obrázku 26 jsou znázorněny mnou navržené číslicové filtry pomocí skriptu **filter.py**. Filtry jsou navrženy dle tabulky 1 (***Zvukové projevy včelstva*** – kapitola ***3. 1***) s přihlédnutím k aplikované spektrální analýze na včelí audio data, která v drtivé většině případů nad 500Hz neprojevovala příliš vysoké hodnoty amplitud, což odpovídá tvrzení uvedenému v úvodu práce, a to že pro zvukové chování včel je typické chování do 500Hz. Spíše v letních obdobích se objevují v inteligentním včelím úlu frekvence až do 2000Hz. Frekvence 2000 - 4000Hz jsou dle spektrální analýzy celkem nevýznamné a jsou tedy zahrnuty do horní propusti. Vzorkovací frekvence 8000 vzorků za vteřinu je pro sledování akustických dat včelstva dostatečná, nicméně digitální filtry jsou navrženy pro všechny vzorkovací frekvence pro možné budoucí změny vzorkování.

Pro vzorkovací frekvenci 8000 vzorků za vteřinu jsou číslicové filtry navrženy v této podobě:

- **Dolní propust**: 0 - 100Hz
- **Pásmová propust**: 100 - 200Hz, 200 - 350Hz, 350 - 500Hz, 500 - 2000Hz
- **Horní propust**: 2000+ Hz (přesněji: D(f)= (2000Hz ; $\frac{\boldsymbol{vzorkovací\ frekvence}}{\mathbf{2}}$ Hz)
    - Horní propust vyfiltruje frekvenční pásmo s ohledem na vzorkovací frekvenci, kdy mezní frekvence počátku se nachází při hodnotě 2000Hz a bez jakéhokoliv útlumu je signál propouštěn do 4000Hz (Do jaké frekvence je signál propouštěn odpovídá Shannonovu teorému).

Úsek kódu filter.py:

```
def band_pass(sample_rate,f_low,f_high):

    f_len = int(sample_rate/8000*800+1)

    l = f_low / sample_rate * 2
    h = f_high / sample_rate * 2

    b = firwin(f_len, [l, h], pass_zero = "bandpass")
    return b
```

```
def high_pass(sample_rate,f_low):

    f_len = int(sample_rate/8000*800+1)

    l = f_low / sample_rate * 2

    b = firwin(f_len, l, pass_zero = "highpass")
    return b


def down_pass(sample_rate,f_high):

    f_len = int(sample_rate/8000*800+1)

    h = f_high / sample_rate * 2

    b = firwin(f_len, h, pass_zero = "lowpass")
    return b


def fil_plot(fs):

   def bp(f_low,f_high,fs):
      return freqz(band_pass(fs,f_low,f_high),worN=int(fs/2),fs=fs)

   def hp(f_low,fs):
      return freqz(high_pass(fs,f_low),worN=int(fs/2), fs = fs)

   def dp(f_high,fs):
      return freqz(down_pass(fs,f_high),worN=int(fs/2), fs = fs)
```

Zdrojový kód 1 – Návrh filtrů (filter.py)

Funkce **band_pass**, **high_pass** a **down_pass** v předchozím kódu mají jako vstupní parametry mezní frekvence a vzorkovací frekvenci. Jejich účelem je vypočíst koeficienty filtrů pro pásmovou, horní a dolní propust. Pro získání koeficientů potřebujeme zvolit správnou **délku filtrů** a vypočíst mezní **cutoff frekvence**.

- Vzorec pro délku filtru (okna) v Pythonu:

$$f_{len} = int(\frac{sample_{rate}}{8000} * 800 + 1) \tag{11}$$

  - $sample_{rate}$ = vzorkovací frekvence
  - Vzorec je v práci navržen pro možný vypočet délky filtru všech vzorkovacích frekvencí. Od délky okna se očekává, aby filtr v přechodu z propustného do nepropustného pásma byl utlumen na nejmenenším

možném rozsahu frekvencí a spádová hrana filtru tak měla v utlumujícím se rozsahu co nejvíce možný prudký spád, díky čemuž budou od sebe filtry perfektně odděleny a filtrace bude probíhat nejlepším možným způsobem. Musíme ale příhlédnou i k tomu, že délka okna nesmí být příliš velká, jelikož by byl algoritmus příliš výpočetně náročný. Vypočtená hodnota $\boldsymbol{f_{len}}$ musí být lichá, aby zahrnovala i Nyquistovu frekvenci, proto nakonci vzorce +1.

- Výpočet mezní frekvence v Pythonu:
  - Mezní frekvence je číslo mezi 0-1, které odpovída Nyquistově frekvenci. Příklady níže pro vzorkovací frekvenci 8000 vzorků za vteřinu.

$$\boldsymbol{Mezní\ frekvence(DP): h = f\_high\ /\ sample_{rate} * 2} \quad (12)$$

př.: h (f_high = 100Hz) = 0,025

$$\boldsymbol{Mezní\ frekvence(HP): l = f\_low\ /\ sample_{rate} * 2} \quad (13)$$

př.: l (f_low = 2000Hz) = 0,5

  - Pásmová propust: skládá se z mezní frekvence dolní propusti (DP) a horní propusti (HP)

- Výpočet parametru (koeficientů) ***b*** pro funkci **scipy.signal.lfilter** a **scipy.signal.freqz** zajistí funkce **scipy.signal.firwin** (Zdrojový kód 1)

Pro vizualizaci navržených číslicových filtrů v **matplotlib** je nutno využít funkci **scipy.signal.freqz(b)**, která vypočte frekvenční odezvu digitálního filtru. Jako povinný parametr je parametr ***b***, a tedy pole, které vrací funkce **firwin**. Funkce **freqz** vrací dva parametry ***w*** a ***h***. Parametr ***h*** je frekvenční odezva jako komplexní číslo a parametr ***w*** je frekvence, při kterých bylo počítáno ***h***, ve stejných jednotkách jako vzorkovací frekvence. **[37]** Zdrojový kód 2 se v součinnosti s funkcemi ze zdrojového kódu 1 postará o vizualizaci navržených filtrů.

Úsek kódu filter.py:

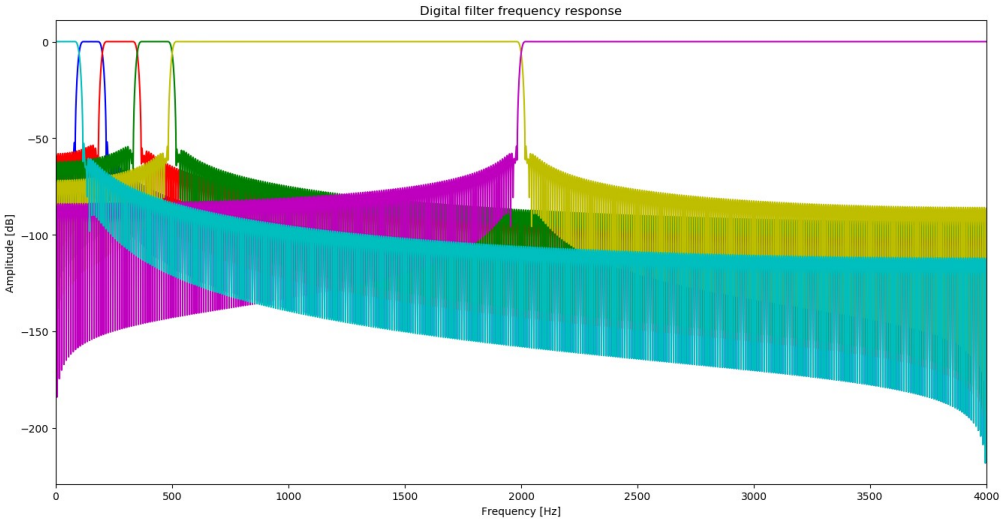
```
w,h=bp(100,200,fs)
plt.plot(w, 20 * np.log10(abs(h)), 'b')
w,h=bp(200,350,fs)
plt.plot(w, 20 * np.log10(abs(h)), 'r')
w,h=bp(350,500,fs)
plt.plot(w, 20 * np.log10(abs(h)), 'g')
w,h=bp(500,2000,fs)
plt.plot(w, 20 * np.log10(abs(h)), 'y')
w,h=hp(2000,fs)
plt.plot(w, 20 * np.log10(abs(h)), 'm')
w,h=dp(100,fs)
plt.plot(w, 20 * np.log10(abs(h)), 'c')
```

Zdrojový kód 2 – Vykreslení navžených filtrů (filter.py)

Při náhledu na zdrojový kód 2 je vidět implementace pro vykreslování navržených filtrů v Pythonu, kdy parametr ***w*** odpovídá ose x a tedy frekvenční složce a parametr ***h*** byl převeden do podoby útlumu dle následujícího vzorce:

$$\boldsymbol{a = 20 * log_{10} * abs(h)} \tag{14}$$

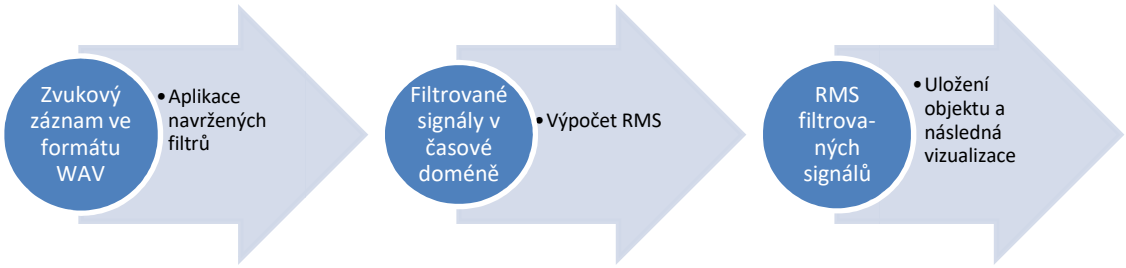
Mezní frekvence však odpovídá -6dB (místo obvyklých -3dB) a zisk v tomto bodě klesne o polovinu narozdíl od jiných metod návrhu filtru. To, že amplituda signálu bude v mezní frekvenci utlumena dvojnásobně, bylo testováno skriptem **test.py** (po úpravě kódu - více popsáno v kapitole ***Testování***). Parametr ***h*** byl převeden do pole logaritmických hodnot a na parametr byla aplikovaná metoda ***abs*** z důvodu toho, že ***h*** je frekvenční odezva jako komplexní číslo. Výsledkem jsou tedy navržené číslicové filtry zobrazené na obrázku 26.



Obrázek 26 - Vizualizace navržených filtrů

8. 2. 2 Aplikace číslicových filtrů na zvuková data

Postup aplikace filtru je pro názornost zobrazen obrázkem 27.



Obrázek 27 - Proces znázorňující postup zpracování filtru

Následující zdrojový kód 3 je úsek skriptu **filter.py**, který se stará o aplikaci filtrů na zvuková data.

Funkce třídy **Filtered_signal** vyfiltrují celistvý původní záznam dle využitého číslicového filtru. **__init__** inicializuje objekt dané třídy, kdy následně funkce **filtered_x_DP** (dolní propust), **filtered_x_HP** (horní propust) a **filtered_x_BAND** (pásmová propust) vrací filtrovaný signál v časové doméně. U všech zmiňovaných funkcí je proměnné f1, f2 či f3 přiřazena vrácená hodnota funkcemi zmíněných ve zdrojovém kódu 1, a to přesněji již zmiňované vstupní koeficienty filtru (parametr ***b*** pomocí funkce firwin), které vrací navržené filtry se specifikovanými mezními frekvencemi filtru. Vstupní koeficienty filtru, které vrací navržené filtry, přidělíme funkci **lfilter(b, a=[1.0], x=vstupní signál)** jako parametr **b.** Tato funkce vyfiltruje datovou sekvenci pomocí digitálního filtru, a získáme tedy konečný filtrovaný záznam v časové složce dle navrženého filtru.

Funkce **filtered_sample** třídy **Filtered_sample** vypočte jednu reprezentační hodnotu pro signál filtrovaný v čase, přesněji efektivní hodnotu RMS, kterou je poté možné ukládat jednoduše do databáze. Efektivní hodnota RMS bude vypočtena pro každý číslicový filtr aplikovaný na včelí zvukový záznam v čase dle následujícího vzorce:

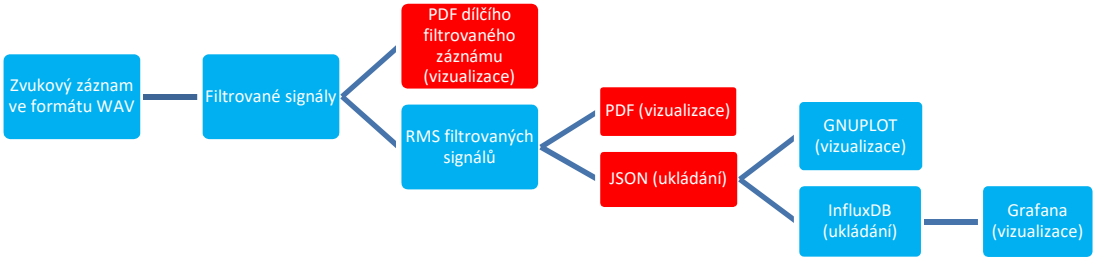
$$x_{RMS} = \sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}} \tag{15}$$

Úsek kódu filter.py:

```
class Filtered_signal:

  def __init__(self, sound, sample_rate, f_low, f_high):
    self.sound = sound
    self.sample_rate = sample_rate
    self.f_low = f_low
    self.f_high = f_high

  def filtered_x_DP(self):
    f1 = down_pass(self.sample_rate,self.f_high)
    filtered_DP = lfilter(f1, [1.0], self.sound)
    return filtered_DP

  def filtered_x_HP(self):
    f2 = high_pass(self.sample_rate,self.f_low)
    filtered_HP = lfilter(f2, [1.0], self.sound)
    return filtered_HP

  def filtered_x_BAND(self):
    f3 = band_pass(self.sample_rate,self.f_low,self.f_high)
    filtered_BAND = lfilter(f3, [1.0], self.sound)
    return filtered_BAND

class Filtered_sample:

  def __init__(self, signal, sample_rate):
    self.signal = signal
    self.sample_rate = sample_rate

  def filtered_sample(self):
    f_numpy = np.array(self.signal)
    wav_len = (len(f_numpy)/self.sample_rate)
    f_sample=sqrt((1/(self.sample_rate*wav_len))*sum...
    ((f_numpy)*(f_numpy)))
    return f_sample
```

Zdrojový kód 3 – Aplikace filtrů na zvuková data (filter.py)

8. 2. 3 Ukládání filtrovaných signálů do JSON a PDF



Obrázek 28 – Postup popisující vizualizaci a ukládání filtrovaného signálu

Výsledky filtrovaných zvukových dat (dle obrázku 28) je po aplikaci číslicových filtrů a vypočtení efektivních (reprezentačních) hodnot pro tyto filtry možno ukládat v několika formátech. Mnou vytvořený skript **wav_converter.py** (s využitím dalších importovaných modulů) je schopen ukládat výsledek po volbě parametrů formou PDF či JSON.

1) JSON

Skript **wav_converter.py** po volbě argumentů aplikuje filtry (pomocí skriptu **filter.py**) na všechna zvuková data (záznamy), nebo data časově omezená a vytvoří JSON soubor (popřípadě soubory) konkrétního dne (“YY-MM-DD_hh_mm_ss.wav”), dle kterého bude pojmenován i výsledný JSON soubor. Objekty v JSON souborech obsahují klíče a hodnoty, kdy hodnoty pro jednotlivé klíče každého objektu jsou vypočteny jako reprezentativní vzorek RMS (dle kapitoly ***8. 2. 2***). Každému klíči z těchto objektů náleží název filtru, na který byl aplikován výpočet RMS. Je nutné ale zmínit, že každému objektu v JSON souboru náleží i čas vzniku zvukového záznamu a RMS amplitudy nefiltrovaného signálu. Skript je připraven pro automatické zpracování Cronem, z čehož plyne, že JSON záznamy se budou generovat automaticky, popřípadě při manuálním spuštění skriptu. Níže je přiložen náhled na úsek JSON výstupu 20-06-01.json (ze dne 1. června 2020):

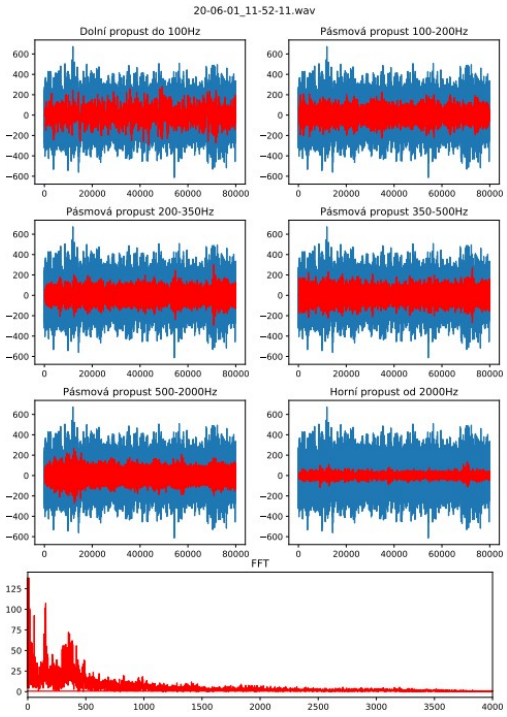
{"Cas": "20-06-01_11-52-11", "Puvodni_zaznam": 137.17, "DP_100Hz": 80.44, "PP_100-200Hz": 58.69, "PP_200-350Hz": 52.15, "PP_350-500Hz": 57.52, "PP_500-2000Hz": 44.15, "HP_2000Hz": 15.62}

{"Cas": "20-06-01_12-02-12", "Puvodni_zaznam": 1687.4, "DP_100Hz": 459.48, "PP_100-200Hz": 329.82, "PP_200-350Hz": 606.55, "PP_350-500Hz": 824.53, "PP_500-2000Hz": 1049.87, "HP_2000Hz": 521.67}

{"Cas": "20-06-01_12-12-12", "Puvodni_zaznam": 499.32, "DP_100Hz": 178.38, "PP_100-200Hz": 123.45, "PP_200-350Hz": 173.71, "PP_350-500Hz": 267.77, "PP_500-2000Hz": 277.42, "HP_2000Hz": 117.87}

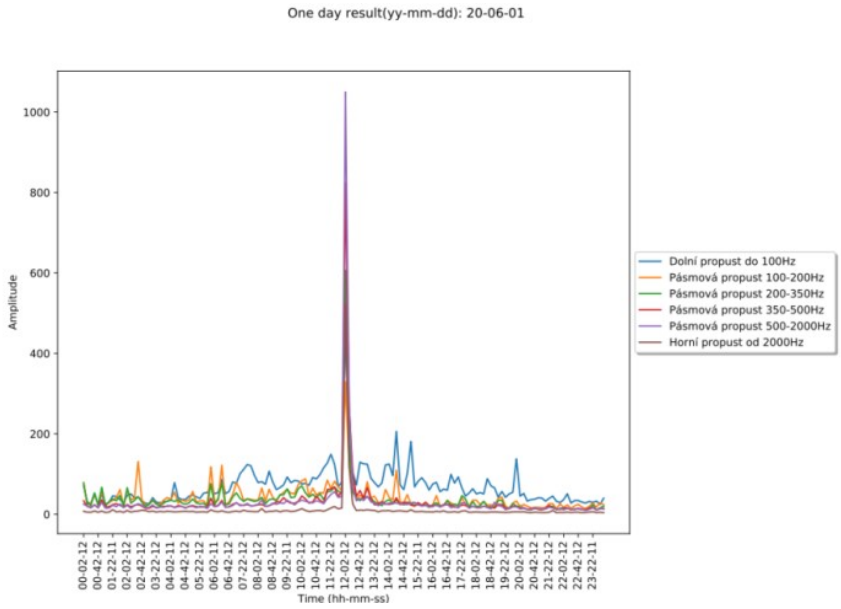
2) PDF

Skript **wav_converter.py** po volbě argumentů zpracuje všechna data, nebo data časově omezená a vytvoří tištitelný soubor PDF pro jednotlivé záznamy a souhrnný PDF soubor za každý daný den. PDF jednotlivých záznamů je rozděleno do několika oken, kdy každé okno znázorňuje aplikaci určitého filtru v čase na včelí zvukový záznam, konkrétně třídou **class Filtered_signal** (zdrojový kód 3). Jedno z oken je také určeno pro vizualizaci FFT za předpokladu, že zvukový záznam bude disponovat vzorkovací frekvencí 8000 vzorků za vteřinu. Okna s filtry na ose y odpovídají amplitudě a na ose x vzorkům (př.: Pokud má záznam délku 10 vteřin a vzorkovací frekvenci 8000 vzorků za vteřinu, jeho následná délka ve vzorcích je 80000 vzorků).



Obrázek 29 – Vizualizace záznamu pomocí PDF

Souhrnná PDF vizualizace celého jednoho dne (stejně jako JSON) má na výstupu vypočtené RMS filtrovaných signálů. Ovšem tyto hodnoty nejsou ukládány textově, ale PDF umožňuje jejich vizualizaci. Je to obdobná funkce jako vykreslení jedno denních dat Grafanou. Na obrázku 30 je tato vizualizace znázorněna.



Obrázek 30 - Vizualizace dne pomocí PDF

8. 2. 4 Ukládání RMS filtrovaných signálů do databáze

Výsledky analýzy včelího zvukového chování je vhodné ukládat i do databáze podporované Grafanou pro následnou vizualizaci v delších časových úsecích. Formát JSON je možné konvertovat a vložit do databáze pomocí skriptu **json_to_influx.py**. Data se do InfluxDB vkládají pravidelně bez činnosti uživatele pomocí Cronu, popřípadě při manuálním spuštění skriptu.

Úseky kódu json_to_influx.py:

```
1) client = InfluxDBClient(host='localhost', port=8086,...
   database="bee_audio")

2) client.switch_database('bee_audio')

#index -> řádek v JSON
#len(data)-> počet řádků JSON výstupu
#data -> nadefinovaný JSON soubor
#for cyklus slouží pro projití všech řádků JSON dat

3) for index in range(len(data)):
3)    write0=(data[index].get("Puvodni_zaznam"))
3)    write1=(data[index].get("DP_100Hz"))
```

```
3)    write2=(data[index].get("PP_100-200Hz"))
3)    write3=(data[index].get("PP_200-350Hz"))
3)    write4=(data[index].get("PP_350-500Hz"))
3)    write5=(data[index].get("PP_500-2000Hz"))
3)    write6=(data[index].get("HP_2000Hz"))

#funkce  timestamp  upraví  časové  razítko  z JSON  objektu  do...
#formátu"%y-%m-%d_%H:%M:%S"
4)    write_time=time.mktime(datetime.datetime.strptime(timestamp...
4)    (data[index].get("Cas")),"%y-%m-%d_%H:%M:%S").timetuple())
4)    write_time=int(write_time)

5)    json_body = [{"measurement": hive_measurement,
5)                  "fields":{"Puvodni_zaznam":write0,
5)                            "DP_100Hz":write1,
5)                            "PP_100-200Hz": write2,
5)                            "PP_200-350Hz": write3,
5)                            "PP_350-500Hz": write4,
5)                            "PP_500-2000Hz": write5,
5)                            "HP_2000Hz": write6,},
5)                  "time": write_time,}]

6)    client.write_points(json_body, time_precision = "s")
```

Zdrojový kód 4 – Zápis bodu do InfluxDB (json_to_influx.py)

Skript **json_to_influx.py**, který je schopen konvertovat JSON výstupy do Influx databáze, postupuje následujícím způsobem:

1) Vytvoření instance InfluxDBClient.
2) Přepnutí na cílovou databázi.
3) Metodou get získat hodnoty pro klíče z JSON objektu.
4) Vytvořit časové razítko ve formátu UNIX timestamp vyhovující času vytvoření včelího záznamu.
5) Vytvoření json_body dle formátu InfluxDB linkového protokolu.
6) Zapsat bod do databáze.

Úseky kódu **json_to_influx.py** (zdrojový kód 4) se dle postupu výše postarají o zápis bodu do databáze. Jelikož se jedná pouze o úseky, každý řádek kódu je pro lepší názornost doplněn o bod, ke kterému se vztahuje dle výše uvedeného postupu, a o komentáře pro případné nejasnosti.

Každý bod zapsaný v Influxu bude disponovat definovaným časovým razítkem, které je nastaveno na časovou přesnost jedné vteřiny stejně jako časové údaje uložené v JSON

datech. Časová přesnost jedné vteřiny byla stanovena pro časové razítko při zapisování bodu do databáze.

8. 2. 5 Automatické zpracování

Pro účely automatického zpracování záznamů do podoby JSON (**wav_converter.py**) s následným převedením do InfluxDB (**json_to_influx.py**) je vyžit softwarový démon Cron. Právě Cron je schopen skripty automaticky spouštět v námi vyhovující čas bez činnosti uživatele s určitými přednastavenými parametry (zobrazeno na obrázku 31).

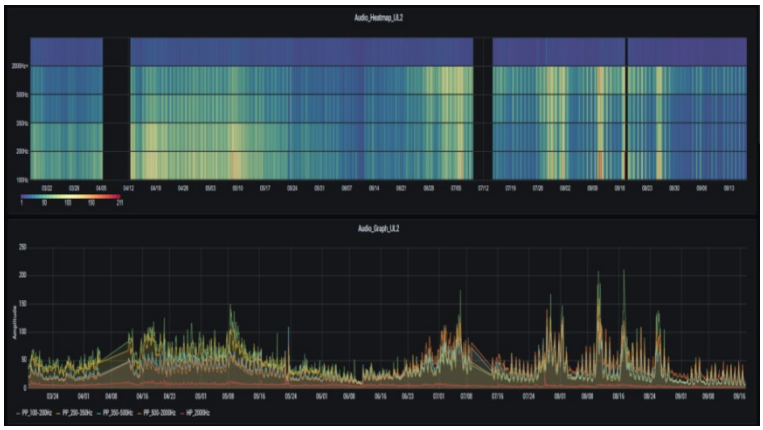
```
0 */6 * * * cd /home/halape00/bee_audio; /usr/bin/python3.6 /home/halape00/bee_audio/wav_converter.py --cron automatic --make_file json --hive ul1
0 */6 * * * cd /home/halape00/bee_audio; /usr/bin/python3.6 /home/halape00/bee_audio/wav_converter.py --cron automatic --make_file json --hive ul2
30 */6 * * * cd /home/halape00/bee_audio; /usr/bin/python3.6 /home/halape00/bee_audio/json_to_influx.py --cron automatic --hive ul1
30 */6 * * * cd /home/halape00/bee_audio; /usr/bin/python3.6 /home/halape00/bee_audio/json_to_influx.py --cron automatic --hive ul2
```

Obrázek 31 - Nastavení dávkového zpracování dat pomocí cronu

Na obrázku 31 je zřetelné, že skript každých 6 hodin převede data úlu 1 a úlu 2 z podoby WAV do formátu JSON (kapitola ***8. 2. 3***) a půl hodiny poté budou tyto data zaznamenána do InfluxDB. Časový interval půl hodiny byl stanoven proto, aby se data konvertovala z JSON do InfluxDB právě po zpracování skriptu **wav_converter.py**. Skript při automatickém zpracování zpracuje pouze den, kdy je skript spuštěn a den předchozí kvůli možným výpadkům. Vyhneme se tedy ztrátě dat, pokud by výpadek trval maximálně 2 dny. V opačném případě je nutno spustit skript manuálně.

8. 2. 6 Vizualizace RMS filtrovaných dat

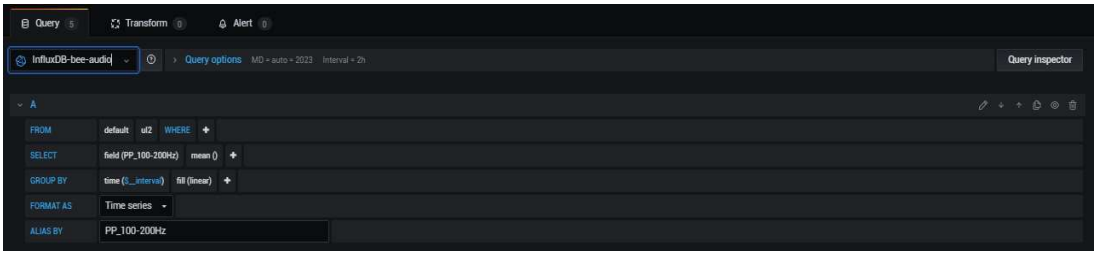
Po připojení k VPN Jihočeské univerzity je možné se do Grafany přihlásit v prohlížeči na adrese: https://bee.prf.jcu.cz:3000 pod účtem vytvořeným správcem Grafany.



Obrázek 32 – Vizualizace filtrovaných dat v Grafaně spojnicovým grafem a teplotní mapou

Na obrázku 32 je vytvořen dashboard pro účely vizualizace včelího chování uvnitř úlu, kdy první panel zobrazuje RMS filtrovaných akustických dat teplotní mapou, kde odpovídá jedno pole efektivní hodnotě filtrovaného signálu daného času v daném frekvenčním pásmu pro lepší orientaci ve druhém panelu (z důvodu podobnosti filtrovaných dat), který zobrazuje totožná data spojnicovým grafem. Ve Spojnicovém grafu osa y odpovídá RMS filtrovaného signálu a osa x času.

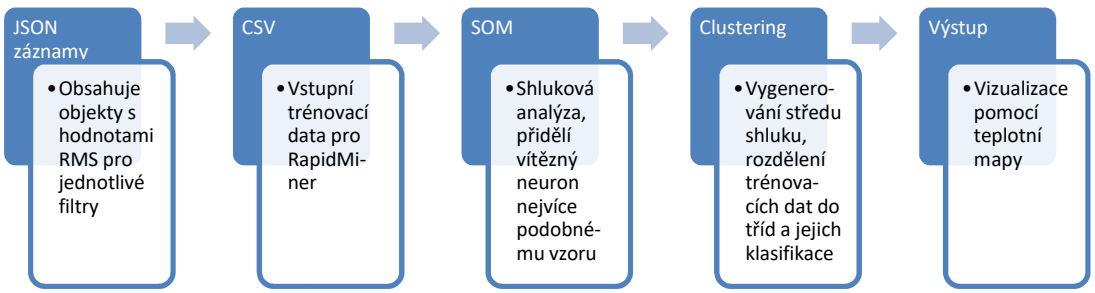
Jednotlivé spojnice jsou vkládány do grafu dotazem (querry) z InfluxDB. Postup pro dotaz je takový, že zvolíme dotazovanou databázi a následně se dotazujeme pomocí: FROM <measurement/měření> SELECT <field/pole> GROUP BY <time($_interval)>. Obrázek 33 dává náhled na vložení jedné spojnice do panelu. V teplotní mapě je postup dotazů obdobný, pouze je důležité změnit způsob vizualizace (konkrétně na Heatmap).



Obrázek 33 - Vkládání spojnice do panelu v Grafaně

Grafana pro účely této práce zobrazuje spojnicové grafy a teplotní mapy pro úl 1 a úl 2, kdy tyto panely zobrazují zvukové chování z těchto úlu od března 2019 do současnosti a panely se každých 6 hodin automaticky aktualizují po vytvoření nových bodů v Influxu Cronem, tudíž Grafana bude poskytovat náhled na včelí data v reálném čase bez nutnosti manuálních spouštění skriptů či zásahů do programů.

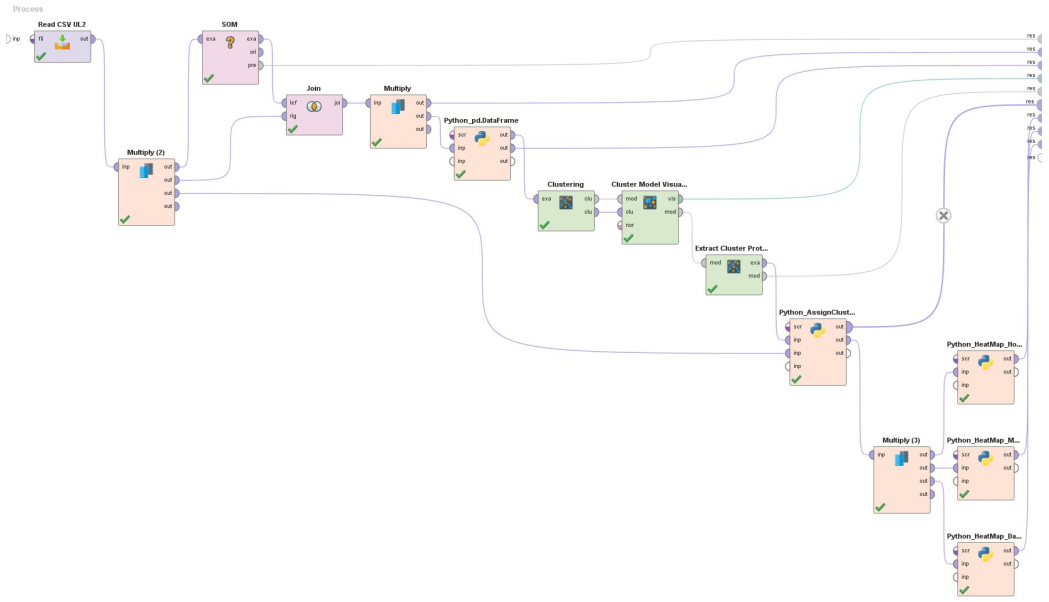
8. 3 Shluková analýza



Obrázek 34 - Postup shlukové analýzy

Grafana sice poskytne přehledný náhled na zvuková data v určitých časových úsecích, ale dále je zapotřebí data určitou formou specifikovat, rozlišovat a přiřadit je k určitému vzorci chování. K tomu napomůže shluková analýza. Pro tyto účely jsem využil vhodný systém pro vytěžování dat a to konkrétněji RapidMiner Studio Educational.

Data pro **RapidMiner** musí být připravena ve formátu CSV, proto jsem vytvořil jednoduchý skript **test.py**, který překonvertuje data z podoby JSON do podoby CSV vhodné pro RapidMiner. Jelikož není vhodné brát v potaz pouze velikost amplitudy (pro následný clustering v mém postupu) z důvodu měnící se hodnoty v různá časová období, tak před konvertováním do **CSV** byly hodnoty RMS filtrovaných signálů normované vstupním signálem a výsledná hodnota odpovídá intenzitě signálu na dané frekvenci. Skript disponuje možností časově omezit (vstupní trénovací) data pro shlukovou analýzu. V této části práce jsem přesněji využil data za jeden rok a to v časovém období **13. září 2019 - 13. září 2020**. Po aplikaci shlukové analýzy a klasifikaci akustických dat bude tedy prokazatelný vzorec akustického chování včel v určitých obdobích. Pro požadovaný výsledek jsem využil operátory: **Read CSV**, **Multiply**, **Join**, **Clustering**, **Cluster Model Visualizer**, **Extract Cluster Prototypes** a dále operátory z Marketplace: **SOM (Self-Organizing Map)** a **Python Scripting.**



Obrázek 35 - Proces zpracování filtrovaných signálů v RapidMineru

Na obrázku 35 je znázorněn postup k požadovaným výsledkům, kdy na vstupní data aplikujeme již zmíněné operátory.

1) **Read CSV**

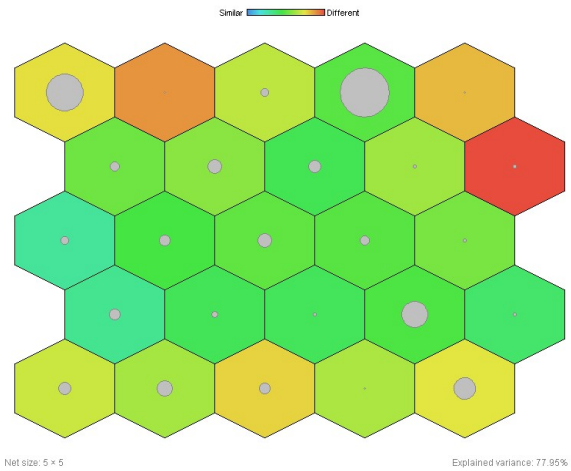
- **INPUT:** CSV soubor
- **OUTPUT:** example set (tabulka)

Operátor, který po definici souboru bude číst data z **CSV** a umožňuje tyto data vést ke zpracování dalším operátorům. V našem případě slouží pro import vstupních trénovacích dat.

2) **SOM**

- **INPUT:** example set
- **OUTPUT:** example set, original, preprocessing model

Samoorganizující neuronová síť verze 5. 2. 0 je dostupná na Marketplace. Síť při ose x odpovídá atributu ***SOM_0*** a při ose y atributu ***SOM_1***, kdy neuronu na místě [0;0] odpovídá neuron umístěn v levém horním rohu vizualizační matice. Jako vhodná pro následující data byla zvolena síť **5 x 5** (při volbě větší sítě ležely poblíž sebe blízké neurony) se 100000 trénovacími koly. Síť zobrazena na obrázku 36 pomocí U matice je toroidního tvaru a nemá tedy uzavřený konec. Na vstup se předkládají trénovací data (example set) a na výstupu je tabulka vítězného neuronu pro každý vzor trénovacích dat (example set – bohužel tato tabulka zobrazuje pouze název vzoru (čas) a vítězný neuron), vstupní trénovací množina (original) a vizualizační model SOM (preprocessing model), který umožňuje zobrazovat model různými typy matic (U matice, P matice atd.) barevně či černobíle, což znázorňuje podobnost sousedním neuronům. Počet vzorů náležících každému neuronu odpovídá obsahu šedého kruhu v každém neuronu.



Obrázek 36 - Vizualizovaná neuronová síť SOM s toroidní mřížkou pomocí U matice

3) Join

- **INPUT:** left, right
- **OUTPUT:** join

Operátor Join spojí dvě tabulky do jedné, respektive přiřadí vítězné neurony SOM každému vzoru z trénovací množiny. Z toho vyplývá, že pro vstup left odpovídá výstupní example set ze SOM a right odpovídá example set z výstupu Read CSV. Výstup join je též example set, ale už se spojenými tabulkami, což je dobře vidět na vzorové tabulce níže.

Time	SOM_0	SOM_1	DP_100Hz	PP_100-200...	PP_200-350...	PP_350-500...	PP_500-200...	HP_2000Hz
19-11-27_08-22-12	3	0	0.959	0.103	0.146	0.157	0.105	0.036
19-11-27_08-32-11	3	0	0.911	0.170	0.231	0.175	0.170	0.041
19-11-27_08-42-12	3	0	0.952	0.183	0.170	0.120	0.092	0.025
19-11-27_08-52-12	3	0	0.989	0.070	0.087	0.060	0.057	0.015
19-11-27_09-02-12	3	0	0.992	0.052	0.067	0.051	0.053	0.014
19-11-27_09-12-12	3	0	0.934	0.155	0.212	0.153	0.117	0.033
19-11-27_09-22-11	3	0	0.985	0.084	0.109	0.075	0.067	0.018
19-11-27_09-32-12	3	0	0.988	0.066	0.061	0.053	0.112	0.013
19-11-27_09-42-12	3	0	0.966	0.104	0.162	0.116	0.073	0.019
19-11-27_09-52-12	3	0	0.989	0.066	0.087	0.069	0.056	0.014
19-11-27_10-02-12	3	0	0.963	0.113	0.164	0.116	0.100	0.027
19-11-27_10-12-12	3	0	0.971	0.095	0.130	0.107	0.100	0.027
19-11-27_10-22-12	3	0	0.950	0.142	0.179	0.136	0.120	0.034
19-11-27_10-32-12	3	0	0.961	0.105	0.114	0.108	0.097	0.019

ExampleSet (50,426 examples, 1 special attribute, 9 regular attributes)

Tabulka 3 - Výstup join operátoru Join

4) Python pd.DataFrame

- **INPUT:** script, join (example set)
- **OUTPUT:** example set, example set

Python Scripting verze 9. 8. 0 je dostupná na Marketplace. Tento operátor umožňuje vytvořit nebo importovat libovolný Python skript. Právě Python skript je jedním ze vstupů tohoto operátoru. Druhým vstupem je tabulka, která je výstupem z operátoru Join. Skript vypočte centroidy pro každý neuron ze SOM a centroidy zaznamená do Pandas dataframe (tabulka). Dejme si příklad takový, že skript začne výběrem vzorů pro vítězný neuron SOM_0 = 0 a SOM_1 = 0. Pro vzory náležící tomuto neuronu bude vypočten průměr atributů (DP_100Hz, PP_100-200Hz, PP_200-350Hz, PP_350-500Hz a HP_2000Hz), které budou znázorňovat střed shluku pro daný neuron. Následně

projdeme cyklem i další neurony a získáme výslednou Pandas tabulku zobrazenou níže. Právě Pandas tabulka (example set) je výstupem operátoru, konkrétně tedy dvě tabulky, kdy jedna disponuje i atributy SOM_0 a SOM_1 a druhá nikoliv pro následující clustering.

Row No.	DP_100Hz	PP_100-200...	PP_200-350...	PP_350-500...	PP_500-200...	HP_2000Hz	SOM_0	SOM_1
1	0.402	0.569	0.476	0.320	0.293	0.067	0	0
2	0.564	0.432	0.487	0.319	0.274	0.079	0	1
3	0.633	0.419	0.462	0.272	0.242	0.075	0	2
4	0.664	0.470	0.385	0.222	0.203	0.068	0	3
5	0.556	0.543	0.416	0.232	0.212	0.070	0	4
6	0.631	0.402	0.394	0.295	0.238	0.068	1	0
7	0.777	0.303	0.371	0.255	0.219	0.068	1	1
8	0.695	0.354	0.429	0.285	0.246	0.073	1	2
9	0.721	0.379	0.347	0.271	0.252	0.087	1	3
10	0.755	0.410	0.337	0.193	0.176	0.060	1	4
11	0.881	0.228	0.287	0.183	0.156	0.051	2	0
12	0.886	0.185	0.239	0.207	0.217	0.076	2	1
13	0.816	0.262	0.317	0.254	0.230	0.078	2	2
14	0.767	0.304	0.285	0.274	0.324	0.082	2	3
15	0.855	0.293	0.275	0.183	0.172	0.061	2	4

ExampleSet (25 examples, 0 special attributes, 8 regular attributes)

Tabulka 4 - Výstup example set společně se SOM neurony operátoru Python_pd.DataFrame

5) Clustering

- **INPUT:** example set
- **OUTPUT:** cluster model, clustered set

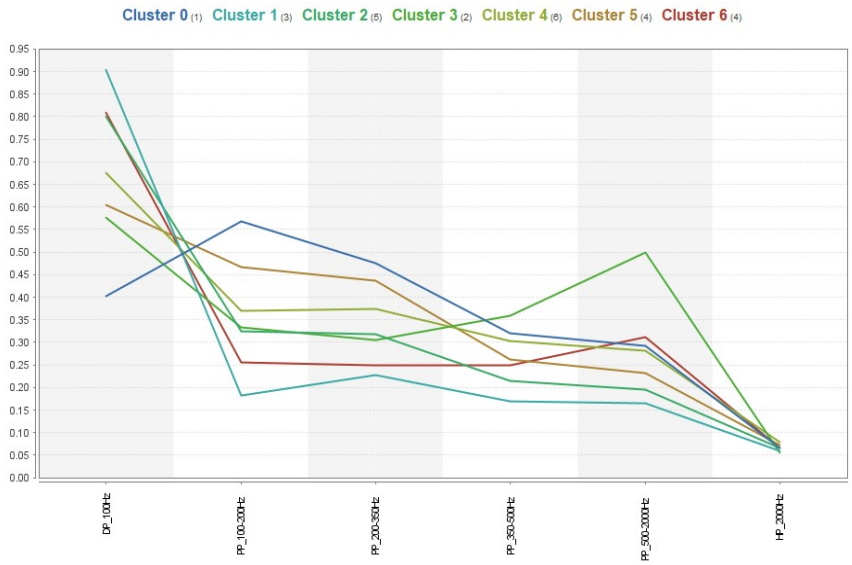
Vstupem operátoru je Pandas tabulka (example set), která je výstupem Python_pd.DataFrame. Počet clusterů v mém postupu byl nastaven na hodnotu 7 a rozdělí tedy centroidy SOM do **7 kategorií (shluků)**. Je zapotřebí si uvědomit, že nejlepším řešením je nejmenší možný počet clusterů, ale zároveň nesmí být tak nízký, aby euklidovská vzdálenost mezi středy shluků a daty nebyla příliš vysoká. Při přihlédnutí k této myšlence a nahlédnutí na vzdálenost mezi sousedními neurony u SOM byla tato hodnota adekvátní, jelikož clustering byl aplikován až na vypočtené centroidy SOM (nikoliv na trénovací data). Výstupem operátoru je cluster model, který umožňuje vizualizovat cluster model centroidů. Druhým výstupem je clustered set, který disponuje tabulkou s atributy vypočtených centroidů. Pro lepší vizualizaci

a za účelem další práce s daty byly oba výstupy převedeny na další operátor Cluster Model Visualizer (až další operátor umožňuje vykreslení centroidů – středů shluků).

6) Cluster Model Visualizer

- **INPUT:** model, clustered data, normalization model
- **OUTPUT:** visualizer output, model output

Vstupem operátoru je model a clustered data (vstup pro normalizaci není využit), kdy vstup model je výstupem operátoru Clustering (cluster model). Vstup clustered data je naopak výstupem clustered set z operátoru Clustering. Tento operátor umožňuje přehlednější zobrazení centroidů clusterů, respektive jejich středů oproti operátoru Clustering, o což se postará výstup visualizer output. Výstup model output je vyveden na vstup dalšího operátoru pro následný převod na example set (prostá tabulka pro další zpracování).



Obrázek 37 - Vizualizace středů shluků

Cluster	DP_100Hz	PP_100-200Hz	PP_200-350Hz	PP_350-500Hz	PP_500-2000Hz	HP_2000Hz
Cluster 0	0.402	0.569	0.476	0.320	0.293	0.067
Cluster 1	0.905	0.182	0.227	0.170	0.164	0.058
Cluster 2	0.800	0.324	0.317	0.215	0.194	0.066
Cluster 3	0.577	0.334	0.304	0.359	0.498	0.055
Cluster 4	0.676	0.370	0.375	0.304	0.282	0.079
Cluster 5	0.604	0.466	0.437	0.261	0.232	0.073
Cluster 6	0.810	0.255	0.249	0.249	0.312	0.065

Tabulka 5 – Středy shluků

7) **Extract Cluster Prototypes**

- **INPUT:** model
- **OUTPUT:** example set, model

Operátor byl využit, jelikož výstup model output z operátoru Cluster Model Visualizer byl zapotřebí převést na výstup example set (tj. tabulku), kterou bude možno přivést na vstup dalšího operátoru (konkrétně Python skriptu). Výstup toho operátoru model provádí obdobnou funkci jako výstup předešlého operátoru Cluster Model Visualizer (visualizer output).

8) **Python AssignCluster**

- **INPUT:** script, example set, example set
- **OUTPUT:** example set, example set

Operátor, jehož vstupem je Python skript, středy jednotlivých clusterů operátoru Extract Cluster Protypes (výstup example set) a vstupní data (výstup operátoru Read CSV), vypočte euklidovskou vzdálenost mezi středy clusterů a vstupními daty. Konkrétně pomocí funkce v Pythonu **scipy.spatial.distance.euclidean(list_a,list_b)**, kdy ***list_a*** jsou atributy vzoru (vstupních záznamů) a ***list_b*** atributy centroidu. Všechny vzory trénovací množiny se prochází pomocí cyklu a centroidy jsou postupně těmto vzorům přiřazovány na základě vzdálenosti, při čemž každému centroidu bude přiřazen nejbližší vzor trénovací množiny. Vstupní data budou tedy rozřazena do shluků nejbližších centroidů a vytvoří výslednou tabulku v Pandas, která bude výstupem (example set). Druhý výstup bude též tabulka (example set). Tento výstup nahradí funkci multiply a jeden výstup bude vyveden na další operátor a druhý na result (výsledek) procesu.

Row No.	Time	Distance	Winner
1	19-09-13_00-...	0.129695530...	cluster_5
2	19-09-13_00-...	0.104964538...	cluster_2
3	19-09-13_00-...	0.152093921...	cluster_2
4	19-09-13_00-...	0.183175602...	cluster_5
5	19-09-13_00-...	0.130458086...	cluster_2
6	19-09-13_00-...	0.158319447...	cluster_5
7	19-09-13_01-...	0.068710057...	cluster_2
8	19-09-13_01-...	0.087144005...	cluster_2
9	19-09-13_01-...	0.130918663...	cluster_5
10	19-09-13_01-...	0.153413717...	cluster_2
11	19-09-13_01-...	0.113812115...	cluster_2
12	19-09-13_01-...	0.175281471...	cluster_2
13	19-09-13_02-...	0.041471193...	cluster_2
14	19-09-13_02-...	0.134935393...	cluster_2
15	19-09-13_02-...	0.147592500...	cluster_5
16	19-09-13_02-...	0.166016937...	cluster_2

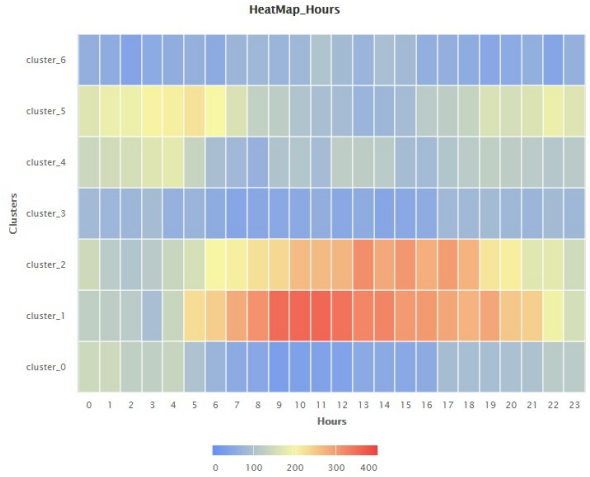
ExampleSet (50,426 examples, 0 special attributes, 3 regular attributes)

Tabulka 6 - Rozdělení záznamů do clusterů a jejich vzdálenost od středu

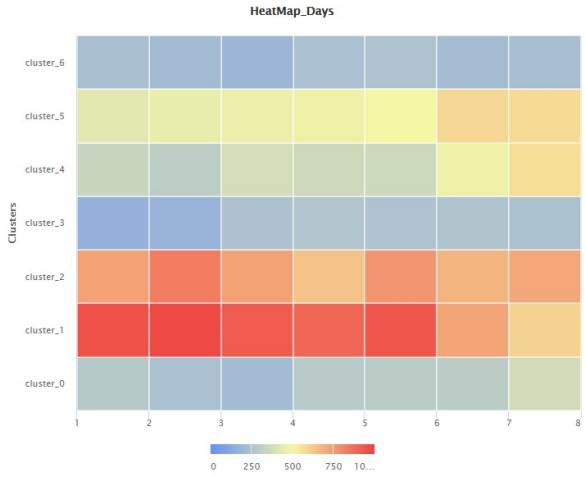
9) Python_HeatMap

- **INPUT:** script, example set
- **OUTPUT:** example set

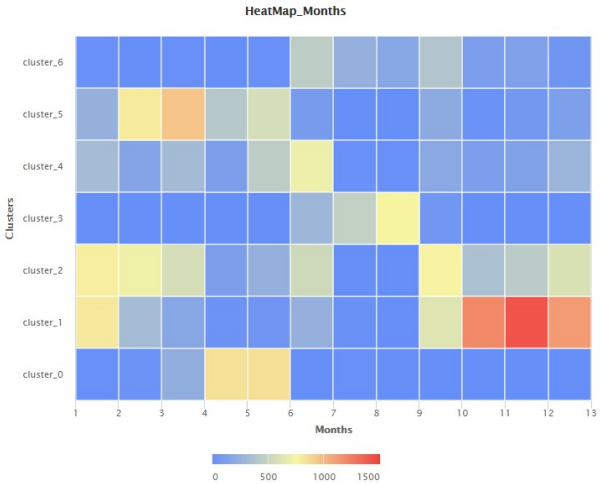
Operátory Python_HeatMap_Hours (hodinový průběh), Python_HeatMap_Days (denní průběh) a Python_HeatMap_Months (mesíční průběh) mají na vstupu operátoru Python skript a klasifikovaná data (přesněji řečeno vzory rozřazené do shluků - výstup operátoru Python_AssignCluster). Účelem těchto operátorů je vytvořit teplotní mapu. Vzory s **euklidovskou vzdáleností** od svého centroidu **<=0.11** se rovnou zahazují, protože se nachází ve velké vzdálenosti od středu a je důležité, aby data, náležící danému clusteru, byla středu shluku nejvíce podobná a teplotní mapa poskytla určitý vzorec chování. Teplotní mapy na obrázku 38, 39 a 40 zobrazují roční data v hodinových, denních a měsíčních úsecích, kdy pole je zbarveno natolik, nakolik počet objektů za celý rok se objevuje v daný moment v náležícím shluku.



Obrázek 38 -Vizualizace ročních dat v hodinových intervalech teplotní mapou



Obrázek 39 - Vizualizace ročních dat v denních intervalech teplotní mapou



Obrázek 40 -Vizualizace ročních dat v měsíčních intervalech teplotní mapou

Na obrázku 38 je zřetelné, že včely po celý rok v nočních intervalech vydávají v největší míře zvukové frekvence náležící *cluster_5* a v denních intervalech náležících *cluster_1*(vizualizace centroidů clusterů na obrázku 37 a tabulka 5 tyto centroidy popisuje). V celkovém průměru je tedy možné tvrdit, že zvukové chování se mění a jinak vypadá přes den a v noci.

Skrze týdenní intervaly je zřetelné, že včely mění své zvukové chování přes týden a víkendy, což je zajímavá myšlenka, jež spíše souvisí s okolním ruchem, který je v těchto obdobích měnící.

Měsíční chování je v tomto případě velice rozmanité a měnící a bylo tedy nutné rok skrze měsíce rozdělit do jednotlivých období odpovídajících **včelařskému roku** a sledovat změny zvukových projevů včelstva náležících těmto obdobím v hodinových intervalech, což bude detailněji rozebráno v kapitole ***Diskuze***.

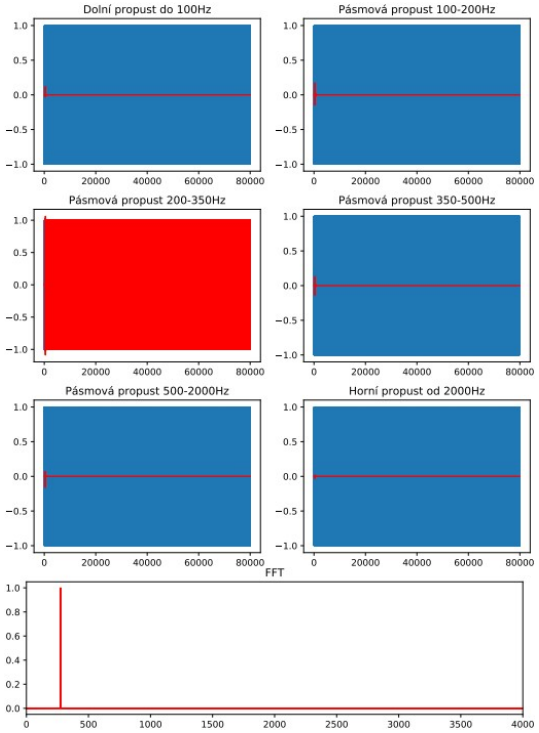
ÚPRAVA PROCESU PRO ÚČELY VČELAŘSKÉHO ROKU:

Právě pro účely odpovídající včelařskému roku byl upraven proces zpracování v RapidMineru – **příloha C**. Neuronové sítě se trénují dle úlu 2 a následně jsou přidělovány centroidy clusterů vstupním datům (Read CSV) úlu 1 a úlu 2 (pro možnost porovnání - doposud jsme řešili vizualizaci dat pouze pro jeden úl). Pro větší přehlednost procesu jsou operátory pro zpracovávání teplotních map obsaženy v Subprocessu (**příloha C**).

Python skript obsažen v Subprocessu - Python_HeatMap_Hours_Seasons obsahuje více výstupů pro možnost vizualizace každého jednotlivého období včelařského roku teplotní mapou v hodinových úsecích, kdy vstupní data budou rozdělena do těchto požadovaných období odpovídajících včelařskému roku a následně vizualizována. Doposud jsme totiž řešili vizualizaci v hodinových úsecích pro celý rok pomocí operátoru Python_HeatMap_Hours, který disponuje pouze jedním výstupem, a bude tak vizualizovat včelí zvukové chování teplotní mapou pro všechna vstupní data bez rozdělení do období včelařského roku.

9 Testování

Fáze testování proběhla za účelem kontroly správnosti výstupů v porovnání se vstupními daty. Z tohoto důvodu bylo zapotřebí využít takový zvukový záznam, u kterého je předem znám jeho výsledek. Pro tento účel jsem vytvořil Python skript **test.py**, který vygeneruje 6 záznamů (přesněji sinusových signálů), kdy každý z nich disponuje vzorkovací frekvencí 8000 vzorků za vteřinu, amplitudou o hodnotě 1 a délkou záznamu 10 vteřin. Liší se pouze frekvencemi a to přesněji [50Hz, 150Hz, 275Hz, 425Hz, 1250Hz, 3000Hz]. Tyto frekvence jsem zvolil proto, že jejich hodnoty se nachází přesně uprostřed mnou navržených filtrů. Z tohoto důvodu by měl být výsledek maximálně přesný. Na následujícím obrázku je znázorněna aplikace filtrů v časové doméně a FFT na jednu z frekvencí a to 275Hz.



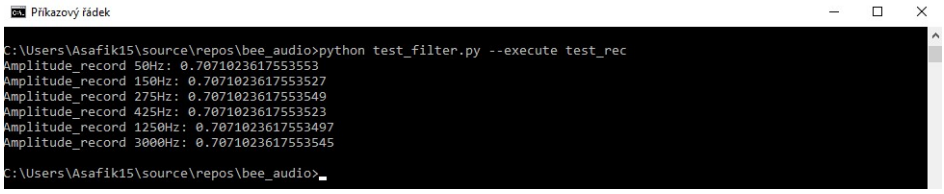
Obrázek 41 - Testování programového vybavení na sinusový signál 275Hz

Výsledek samozřejmě musí korespondovat i s JSON výstupem. Jelikož z 80ti tisíc vzorků filtrovaného signálu je zapotřebí získat jeden reprezentativní vzorek, který bude charakterizovat celý záznam, jehož vypočtená hodnota RMS ze sinusoidy s amplitudou 1 by měla odpovídat hodnotě: $\boldsymbol{RMS} = \frac{\mathbf{a}}{\sqrt{\mathbf{2}}}$ (přibližně 0,707).

Náhled na JSON výstup daného záznamu níže:

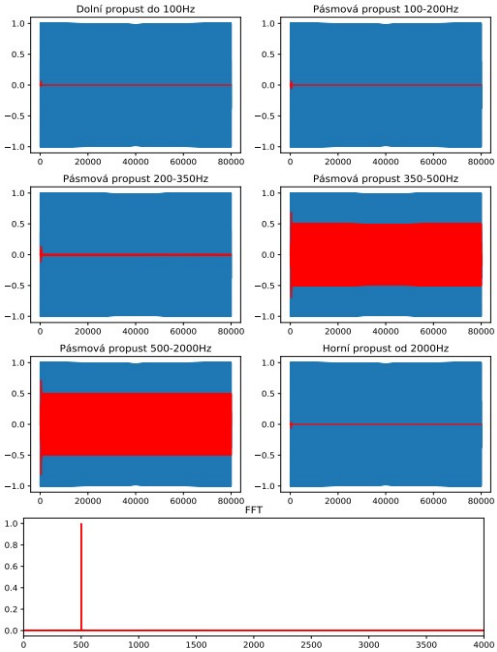
{"Cas": "02-01-01_00-00-00", "Puvodni_zaznam": 0.71, "DP_100Hz": 0.0, "PP_100-200Hz": 0.0, "PP_200-350Hz": 0.71, "PP_350-500Hz": 0.0, "PP_500-2000Hz": 0.0, "HP_2000Hz": 0.0}

Pro kontrolu správnosti výpočtu RMS byl aplikován vzorec výpočtu RMS i ve skriptu **test.py** na všechny nefiltrované záznamy (obrázek 42), se kterými jednotlivé výsledky JSON výstupů korespondují, tudíž lze tvrdit, že zpracování filtrů i výpočtů RMS funguje správně.



Obrázek 42 - Kontrola správnosti vzorce pro výpočet RMS

Bylo zapotřebí i testovat, zda mnou navržené filtry budou utlumovat signál o požadovanou hodnotu i v mezní frekvenci, kdy běžně této mezní frekvenci odpovídá pokles o 3dB. V případě funkce **firwin** pokles na mezní frekvenci odpovídá 6dB (tedy dvojnásobku). Za tímto účelem byl upraven skript **test.py**, který vygeneruje 500Hz sinusový signál pro testování. Toto testování přineslo správný a požadovaný výsledek.



Obrázek 43 - Testovaní programového vybavení na sinusový signál 500Hz

10 Diskuze ohledně výsledků

10. 1 Diskuze o aplikaci shlukové analýzy na včelařský rok

Po odborné diskuzi s Ing. Václavem Krištůfkem, CSc. jsme rozdělili včelařský rok dle chování včel v úlu, které je obsaženo v **příloze A,** do období popsaných níže:

1. Předjaří – 1. března – 31. března
2. Jaro – 1. dubna – 31. května
3. Časné léto – 1. června – 30. června
4. Plné léto – 1. července – 31. července
5. Podletí – 1. srpna – 31. srpna
6. Podzim – 1. září – 30. listopadu
7. Zima – 1. prosince – 28. února

Rok jsme rozčlenili za účelem sledování těchto období pomocí shlukové analýzy. Výsledky jsou přiloženy jako **příloha B**.

Z výsledků je zřetelné, že pro podzim a zimu jsou v úlu 1 a 2 nejvíce projevující se nízké frekvence do 100Hz s maximálními (spíše nočními) výchylkami do 350Hz, tudíž nízká včelí aktivita a klid v úlu i se současným přihlédnutím k amplitudě signálu. Jedná se o období léčení včelstva.

Při předjaří je pro úl 1 stále podobné zvukové chování podzimu a zimě. Naopak v úlu 2 tyto nižší frekvence ustávají a ve vyšších mírách se projevují frekvence v rozmezí 100 - 350Hz z důvodu zvyšující se včelí aktivity – kvete například olše lepkavá, začíná produkce pylu.

Na jaře frekvence 100 - 350Hz s přesahy do 500Hz se projevují ve vyšší míře už i v úlu 1, jelikož frekvence zvukových projevů včelstva v pásmu 350 - 500Hz náleží dle odborného zkoumání úlu 2 období rojení (k nahlédnutí v kapitole ***10. 2***).

Při časném létě je chování velice rozmanité, pouze v případě úlu 2 se začínají projevovat v nočních obdobích už i vyšší frekvence zvuku 500 - 2000Hz. Včely v tomto období provádí spoustu práce, je to období medobraní.

Pro plné léto i časné léto je charakteristické podobné zvukové chování, včelí aktivita v úlu se projevuje ve vyšších mírách už i na vysokých frekvencích do 2000Hz. V tomto období dochází ke krmení na obranu proti loupežím a nebezpečím, pro které jsou vysoké frekvence celkem charakteristické.

Závěrem lze říci, že včelí zvukové chování lze podle mého postupu rozdělit do 3 částí:

- **Podzim a zima** – Nízká včelí aktivita, charakteristické jsou velice nízké frekvence především do 100Hz.
- **Jaro** – Zvýšená včelí aktivita, pro kterou je charakteristické zvukové chování do 350Hz s přesahy do 500Hz.
- **Léto** – Maximální včelí aktivita, začínají se projevovat i vysoké frekvence do 2000Hz.

Z výsledků v **příloze B** je zřetelné, že ve všech clusterech je nejvýraznější frekvence 0 - 100Hz. Stálo za možnost tuto frekvenci vynechat, ale to by nebylo vhodné, protože v zimních obdobích tato frekvence převažuje (to samé platí i pro nízkou aktivitu včel v úlu). Musíme tedy brát v potaz, že pokud byl záznam přidělen clusteru s vyšší amplitudou pásma 0 - 100Hz než jiného pásma, nemusí tomu tak být úplně přesně i v realitě. Pouze je tomuto clusteru záznam nejbližší a mohou převažovat i jiné frekvenční pásma. Přeci jen je velice náročné rozdělit data za 1 rok do minimálního počtu clusterů a musíme tedy brát v potaz růst vyšších frekvencí a pokles těch nižších (0 - 100Hz) a dívat se na výsledky komplexně. Proto, jak již bylo popsáno v této kapitole, jsou zřetelné změny zvukového chování včel skrze včelařský rok.

Lze tedy závěrem říci, že není špatná úvaha na zvuková data aplikovat náročnější a pokročilejší neuronovou síť, protože výsledky výstupů těchto dat jsou zajímavá a přínosná. To by byl dobrý námět pro pokračování této práce.

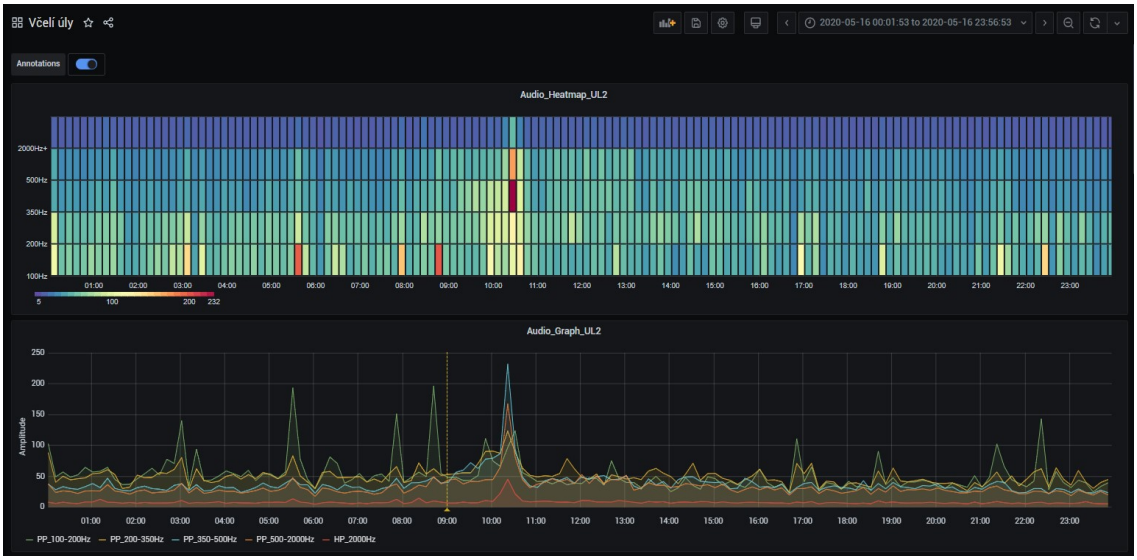
10. 2 Zajímavé časové úseky

Dle **přílohy A** bych touto cestou rád zobrazil zajímavé časové úseky vizualizované Grafanou.

Na obrázku 44, kde je zobrazen celý den v úlu 2, konkrétně 19. května 2019, se vyrojilo včelstvo. K vyrojení včelstva dle záznamů došlo v 11.00, ale pozorován osobně návrat včelstva od 18:00 do 18:30. Dle váhy v úlu šlo o návrat 5 kg včel. Na obrázku je zřetelné, že pro rojení je charakteristické pásmo 350 - 500Hz.

Obrázek 44 - Rojení 1

Na obrázku 45 je opět vidět rojení. Jedná se o den 16. května 2020 v úlu 2. Pro rojení je opět charakteristické pásmo 350 - 500Hz (v čas 10:22). Pokles váhy při rojení byl 5kg a byl tedy velice výrazný.



Obrázek 45 - Rojení 2

Na obrázku 46 je zobrazen úl 2 ze dne 10. 12. 2019, kdy proběhlo léčení Varidolem. Z obrázku je zřetelné, že včely zareagovali na nanesení Variolu a dále se celý den zkrátka vzpamatovávali z tohoto léčení, protože léčba se provádí v uzavřeném úlu.

Obrázek 46 - Léčení Varidolem

11 Závěr

Hlavním cílem této práce byl návrh a implementace programového vybavení pro analýzu včelích akustických dat. Analýza byla zaměřena na spektrální analýzu, shlukovou analýzu a číslicovou filtraci.

Pro spektrální analýzu a číslicovou filtraci bylo v této práci vytvořeno 6 skriptů napsaných v jazyce Python o přibližné délce 1 000 řádků kódu. Pro shlukovou analýzu byl využit grafický nástroj pro těžení dat RapidMiner doplněný o skripty v Pythonu z důvodu implementace chybějících funkcionalit.

Nejprve jsem navrhl a implementoval program pro účely spektrální analýzy, která poskytne včelí akustický záznam (popřípadě záznamy jednoho dne prostřednictvím spektrogramu) ve frekvenční doméně – skript **fft_spectogram.py**. Tento program je schopen výsledek spektrální analýzy vykreslovat pomocí knihovny matplotlib a byl základním zdrojem informací společně se zvukovými projevy včelstva v kapitole ***3. 1***, na jehož základě byly navrženy číslicové filtry.

Číslicové filtry poskytují hodnoty amplitud ve specifických frekvenčních pásmech. Návrh číslicových filtrů ve skriptu **filter.py** byl realizován pomocí knihovních funkcí v Pythonu. Právě tento skript navržené filtry na včelí akustická data aplikuje v časové doméně a následně vypočte jednu efektivní (reprezentační) hodnotu pro jednotlivé filtrované signály včelího akustického záznamu. Prostřednictvím skriptu **wav_converter.py** (a jemu náležícímu modulu **wav_converter_library.py**) je výsledek číslicové filtrace ukládán textově do JSON souboru (efektivní hodnoty filtrovaných signálů) nebo pomocí tištitelných souborů v PDF (filtrované signály v časové složce doplněné o spektrální analýzu). Objekty JSON souboru (popřípadě souborů) lze zapsat do databáze InfluxDB pomocí skriptu **json_to_influx.py** a následně po propojení této databáze s webovou aplikací Grafana lze efektivní hodnoty jednotlivých filtrovaných signálů uložených v databázi sledovat v reálném čase. Zároveň skripty **wav_converter.py** a **json_to_influx.py** byly napsány tak, aby umožnily průběžné i dávkové ukládání výsledků číslicové filtrace.

Výsledky frekvenční analýzy zvukových signálů z včelího úlu přineslo řadu zajímavých výsledků a několik z nich (konkrétně rojení a léčení) je popsáno v kapitole ***10. 2***.

Pro účely testování spektrální analýzy a číslicové filtrace jsem navrhl a implementoval program **test.py**, který vygeneruje 6 sinusových signálů s přednastavenou délkou, frekvencí a vzorkovací frekvencí. Výsledkem testování byly výstupy filtrů, které odpovídaly předem vypočteným hodnotám pro testovací signály, a testování tedy potvrdilo správnost výpočtů.

Poslední metoda analýzy provedena v rámci této práce je shluková analýza pomocí **RapidMineru**. Pomocí Python skriptu **test.py** s vhodně užitými parametry při spuštění se vyberou záznamy JSON formátu, bude vybrána časově omezená množina výstupů efektivních hodnot filtrů, které budou normované příslušnou hodnotou RMS nefiltrovaného signálu, a následně bude tato množina uložena jako jeden soubor ve formátu CSV. Ten byl využit jako vstupní trénovací množina v RapidMineru, na jejíž jednotlivé vzory byla aplikována neuronová síť typu SOM, ve které bylo identifikováno 7 shluků, pro něž byl vypočten střed shluku, který poskytl informaci o specifických projevech amplitud jednotlivých frekvenčních pásem, jež odpovídají odlišným projevům včelstva. Po následně provedené clusterizaci vstupních dat, a tedy jejich rozřazení do těchto shluků (k nejbližšímu středu) pomocí implementovaných skriptů v jazyce Python, byla zjištěna četnost těchto jednotlivých profilů skrze jeden rok. Tato četnost v hodinových, denních a měsíčních intervalech byla následně vizualizována pomocí teplotních map, které poskytly různé profily včelího zvukového chování napříč sezónami, což bylo detailně rozebráno v kapitole ***10. 1*** se zaměřením na jednotlivá období odpovídajících včelařskému roku.

Další možný vývoj práce by právě mohl směřovat k aplikaci pokročilejších neuronových sítí, které budou schopné včelí akustická data přiřazovat ke zdravotním indikátorům včelích úlů a včas detekovat nemoci včel, což je myšlenka, která již byla zmíněna v kapitole ***4***. Tento další vývoj práce by tedy zabránil nadměrnému úbytku včelstva.

Tato práce poskytuje možnost sledovat zvukové chování včel a reagovat na neobvyklé situace v úlu, pro které je umožněn hlubší rozbor a sledování těchto kritických momentů

v časové nebo frekvenční doméně. Tyto neočekávané výchylky zvukových signálů je umožněno přiřazovat k ostatním měřeným veličinám v úlu, což napomůže k prevenci a řešení problematiky úbytku včel. Tudíž tato práce je efektivním nástrojem pro monitoring akustického chování včelstva a výchozím bodem pro analýzu včelích akustických dat.

Seznam použité literatury, obrázků, tabulek a rovnic

Seznam použité literatury

[1] Využití akustické technologie pro vyhodnocování stavu včelstev. Jjvcela.sweb.cz [online]. [cit. 2020-10-23]. Dostupné z: http://jjvcela.sweb.cz/Poslouchame_vcely.html

[2] IVANSKÝ, Tomáš. Úvod do akustického monitoringu včlestva. Moderní včelař. 2017(1), strana 14. [cit. 2020-10-22]. ISSN 1214-5793.

[3] Analýza a zpracování signálů: 2. Analogové a diskrétní signály. Kiv.zcu.cz [online]. [cit. 2020-10-23]. Dostupné z: http://www.kiv.zcu.cz/~mautner/Azs/Azs2_Spojite_systemy_Vzorkovani.pdf

[4] 2.5 Amplitude, Period and Frequency [online]. Dec 17, 2014 [cit. 2020- 10-23]. Dostupné z: https://www.ck12.org/book/ck-12-trigonometry- second-edition/section/2.5/

[5] 2.4. DISKRÉTNÍ SIGNÁLY: 2.4.1. Vzorkování. In: Is.muni.cz [online]. [cit. 2020-10-22]. Dostupné z: https://is.muni.cz/el/1431/podzim2009/Bi5440/um/USS2_2.pdf

[6] HLAVÁČ, Václav. Fourierova transformace v 1D a 2D: Fourierova tx v 1D, výpočetní složitost, FFT. *People.ciirc.cvut.cz* [online]. České vysoké učení technické v Praze [cit. 2020-10-23]. Dostupné z: http://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/12FourierTxCz.pdf

[7] *Numpy.fft.fft* [online]. České vysoké učení technické v Praze [cit. 2020- 10-23]. Dostupné z: https://numpy.org/doc/stable/reference/generated/numpy.fft.fft.html

[8] SCHWARZ, Daniel. Lineární a adaptivní zpracovní dat: 5. Lineární filtrace: FIR, IIR. Iba.muni.cz [online]. [cit. 2020-10-23]. Dostupné z: https://www.iba.muni.cz/esf/res/file/bimat-prednasky/linearni-a- adaptivni zpracovani-dat/LaAZD-05.pdf

[9] VOJÁČEK, Antonín. Použití filtrů FIR v digitálním zpracování signálů. Automatizace.hw.cz [online]. 2004 [cit. 2020-10-23]. Dostupné z: https://automatizace.hw.cz/clanek/2005110801

[10] ALEXANDER, Winser E. a Cranos M. WILLIAMS. Digital Signal Processing: Principles, Algorithms and System Design. North Carolina State University NC, USA, 2017. ISBN 978-0-12-804547-3.

[11] ŠIRHAL, Lukáš. Sběr a analýza dat z inteligentního včelího úlu: Návrh architektury. In: Theses.cz [online]. České Budějovice, 2018 [cit. 2020- 10-22].

Diplomová práce. Vedoucí práce Miroslav Skrbek. Dostupné z: https://theses.cz/id/51gl1g/Diplomov_prce_-_Luk_irhal.pdf

[12] Introducing JSON: ECMA-404 The JSON Data Interchange Standard. *Json.org* [online]. [cit. 2020-10-23]. Dostupné z: https://www.json.org/json-en.html

[13] Cygwin: Package: gnuplot-wx. *Cygwin.com* [online]. [cit. 2020-10-23]. Dostupné z: https://cygwin.com/packages/summary/gnuplot-wx.html

[14] Archlinux: Xorg (Česky). Wiki.archlinux.org [online]. [cit. 2020-10-23]. Dostupné z: https://wiki.archlinux.org/index.php/Xorg_(%C4%8Cesky)

[15] VAŠICA, Jakub. Databáze pro ukládání a správu časových řad: 3.6 InfluxDB [online]. Technická univerzita Ostrava, 2016, strana 22 a 23 [cit. 2020-10-23]. Dostupné z: https://dspace.vsb.cz/bitstream/handle/10084/115897/VAS0073_FEI_N2647_2612T025_2016.pdf?sequence=1&isAllowed=y. Diplomová práce. Technická univerzita Ostrava. Vedoucí práce Jan Martinovič.

[16] Exploring the InfluxDbPublisher in Talaiot: InfluxDb. *Proandroiddev.com* [online]. Apr 15, 2019 [cit. 2020-10-23]. Dostupné z: https://proandroiddev.com/exploring-the-influxdbpublisher-in-talaiot-ae6c60a0b0ec

[17] KUBICA, Tomáš. IoT: InfluxDB a data ze senzorů. *Cloudsvet.cz* [online]. 15. 9. 2016 [cit. 2020-10-23]. Dostupné z: https://www.cloudsvet.cz/?p=674

[18] Grafana. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2020-10-23]. Dostupné z: https://en.wikipedia.org/wiki/Grafana

[19] KAŇA, Michal. KOHONENOVA SÍŤ [online]. Brno, 2011 [cit. 2020-10-23]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=42233 . Bakalářská práce. Vysoké účení technické v Brně. Vedoucí práce Václav Jirsík.

[20] SKRBEK, Miroslav. Výpočetní inteligence (766 - 2018 - LS) [online]. In: https://elearning.jcu.cz. 9. 5. 2018. [cit. 2020-10-22]. Dostupné z: https://elearning.jcu.cz/course/index.php?categoryid=1006

[21] Neuronové sítě a princip jejich fungování. In: Napocitaci.cz [online]. 8. 9. 2017 [cit. 2020-10-23]. Dostupné z: https://www.napocitaci.cz/33/neuronove-site-a-princip-jejich-fungovani-uniqueidgOkE4NvrWuNY54vrLeM670eFNQh552VdDDulZX7UDBY/

[22] ZÁČEK, Viktor. KOHONENOVA SAMOORGANIZAČNÍ MAPA: Samoorganizační mapy [online]. Brno, 2012 [cit. 2020-10-23]. Diplomová

práce. VUT Brno. Vedoucí práce Václav Jirsík. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=52027

[23] KOMORÁŠ, Jakub. Modul samoorganizačních map pro program Modeler neuronových sítí [online]. Ostrava, 2020 [cit. 2020-10-23]. Dostupné z: https://dspace.vsb.cz/bitstream/handle/10084/140539/KOM0053_FEI_N2647_2612T025_2020.pdf?sequence=1&isAllowed=y . Diplomová práce. VŠB – Technická univerzita Ostrava. Vedoucí práce David Ježek

[24] Shluková analýza: Algoritmus k-means [online]. In: is.muni.cz. [cit. 2020-10-23]. Dostupné z: https://is.muni.cz/th/172767/fi_b/5739129/web/web/kmeans.html

[25] Vytěžování dat, cvičení 2: Uvod do RapidMineru. *Cw.fel.cvut.cz* [online]. Fakulta elektrotechnická, ČVUT [cit. 2020-10-23]. Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/a7b36vyd/cviceni/c02-rapidminer.pdf

[26] RapidMiner. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2020-10-23]. Dostupné z: https://en.wikipedia.org/wiki/RapidMiner

[27] GÓRECKI, Jan. Rapid Miner [online]. Ostrava, 2011, strana 4 a 9 [cit. 2020-10-23]. Referát. Katedra informatiky FEI VŠ-TUO. Dostupné z: http://www.person.vsb.cz/archivcd/FEI/MAD/Priloha%201%20Referat%20Rapid%20Miner.pdf

[28] Osbeehives: tech-specs [online]. [cit. 2020-10-22]. Dostupné z: https://www.osbeehives.com/pages/tech-specs

[29] Stupeň krytí. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation [cit. 2020-10-23]. Dostupné z: https://cs.wikipedia.org/wiki/Stupe%C5%88_kryt%C3%AD

[30] COPLEY SMITH, Tristan. The Hiveeyes Community [online]. Jun '17 [cit. 2020-10-22]. Dostupné z: https://community.hiveeyes.org/t/hello-from-open-source-beehives/372

[31] SciPy.org [online]. [cit. 2020-10-23]. Dostupné z: https://www.scipy.org/

[32] Matplotlib: Matplotlib: Visualization with Python [online]. [cit. 2020-10-23]. Dostupné z: https://matplotlib.org/

[33] NumPy: NUMERICAL COMPUTING TOOLS [online]. [cit. 2020-10-23]. Dostupné z: https://numpy.org/

[34] OS Module in Python with Examples [online]. 2019 [cit. 2020-10-23]. Dostupné z: https://www.geeksforgeeks.org/os-module-python-examples/

[35] *Python Advanced Course Topics: sys-Modul* [online]. 2019 [cit. 2020-10-23]. Dostupné z: https://www.python-course.eu/sys_module.php

[36] Savitzky – Golayův filtr. *Cs.qaz.wiki* [online]. [cit. 2020-10-23]. Dostupné z: https://cs.qaz.wiki/wiki/Savitzky%E2%80%93Golay_filter

[37] Scipy.signal.freqz [online]. [cit. 2020-10-23]. Dostupné z: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.freqz.html#scipy-signal-freqz

Seznam použitých obrázků a tabulek

[38] **Tabulka 1 - Zvukové chování včel [38]:** IVANSKÝ, Tomáš. Úvod do akustického monitoringu včlestva. Moderní včelař. 2017(1), strana 14. [cit. 2020-10-22]. ISSN 1214-5793.

[39] **Obrázek 4 - Průběh vzorkovaného signálu [39]:** 2.4. DISKRÉTNÍ SIGNÁLY: 2.4.1. Vzorkování. In: Is.muni.cz [online]. [cit. 2020-10-22]. Dostupné z: https://is.muni.cz/el/1431/podzim2009/Bi5440/um/USS2_2.pdf

[40] **Obrázek 5 - Vznik digitálního signálu [40]:** Přehled modulačních metod: Princip pulzní kódové modulace PCM. In: SlidePlayer.cz: slide/3125414/ [online]. [cit. 2020-10-22]. Dostupné z: https://www.google.com/search?q=kvantovani+a+pcm&sxsrf=ALeKk02UY76Ys4ieemQAJUR2qdXWzfJAcw:1602148649868&source=lnms&tbm=isch&sa=X&ved=2ahUKEwix9pvp1KTsAhUQEBQKHbR7C8cQ_AUoAnoECAwQBA&biw=1455&bih=717#imgrc=xCoxaCiZrLBwaM

[41] **Obrázek 6 - Fourierova transformace sinusového signálu 50Hz [41]:** Fourier Transform Talk and Python Code. In: Seismos: An irregular blog about the science, history, and culture of geophysics [online]. 6 Feb 2018 [cit. 2020-10-22]. Dostupné z: http://seismosblog.blogspot.com/2018/02/fourier-transform.html

[42] **Obrázek 16 - Průběh učení SOM neuronové sítě [42]:** VOJÁČEK, Antonín. Samoučící se neuronová síť - SOM, Kohonenovy mapy: Princip a struktura [online]. In: www.kiv.zcu.cz . 14 Květen, 2006, strana 3 [cit. 2020-10-22]. Dostupné z: https://www.kiv.zcu.cz/studies/predmety/uir/NS/Samouc_NN2.pdf

[43] **Tabulka 2 – Technické specifikace BuzzBox Mini [43]**: Osbeehives: tech-specs [online]. [cit. 2020-10-22]. Dostupné z: https://www.osbeehives.com/pages/tech-specs

[44] **Obrázek 18 - Popis zařízení BuzzBox [44]:** COPLEY SMITH, Tristan. The Hiveeyes Community [online]. Jun '17 [cit. 2020-10-22]. Dostupné z: https://community.hiveeyes.org/t/hello-from-open-source-beehives/372

[45] **Obrázek 19 - Zařízení BuzzBox [45]:** MF Case Study #017: From Prototypes to Production with MacroFab [online]. In: https://macrofab.com/. [cit. 2020-10-22]. Dostupné z: https://macrofab.com/case-studies/prototypes-production-macrofab/

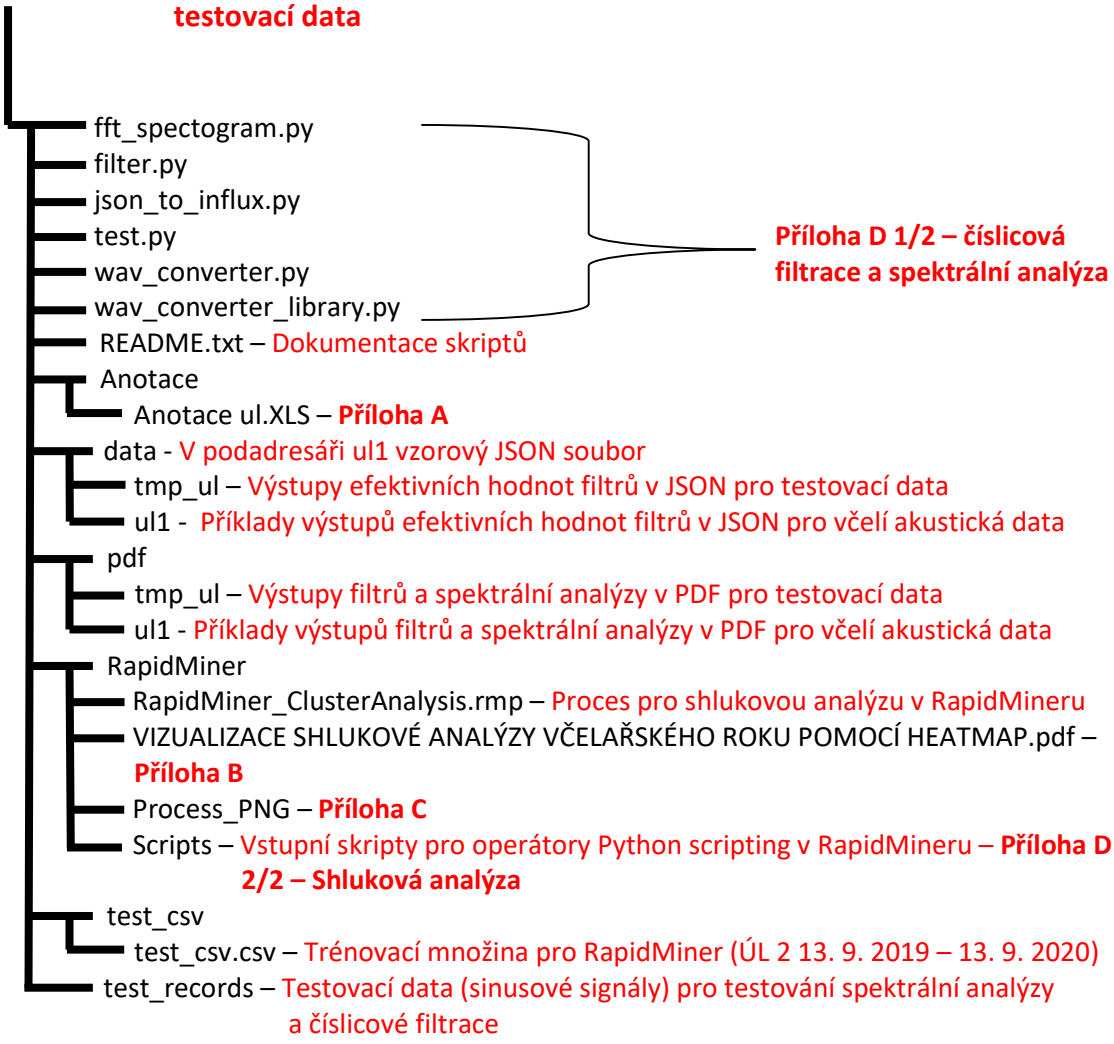
Seznam obrázků, tabulek a zdrojových kódu

Seznam rovnic

Přílohy

Struktura příloh bakalářské práce

audio_data – Příklad akustických dat z úlu
bee_audio – Adresář obsahující skripty, řešení shlukové analýzy, výsledky analýz a testovací data



Příloha A - Anotace

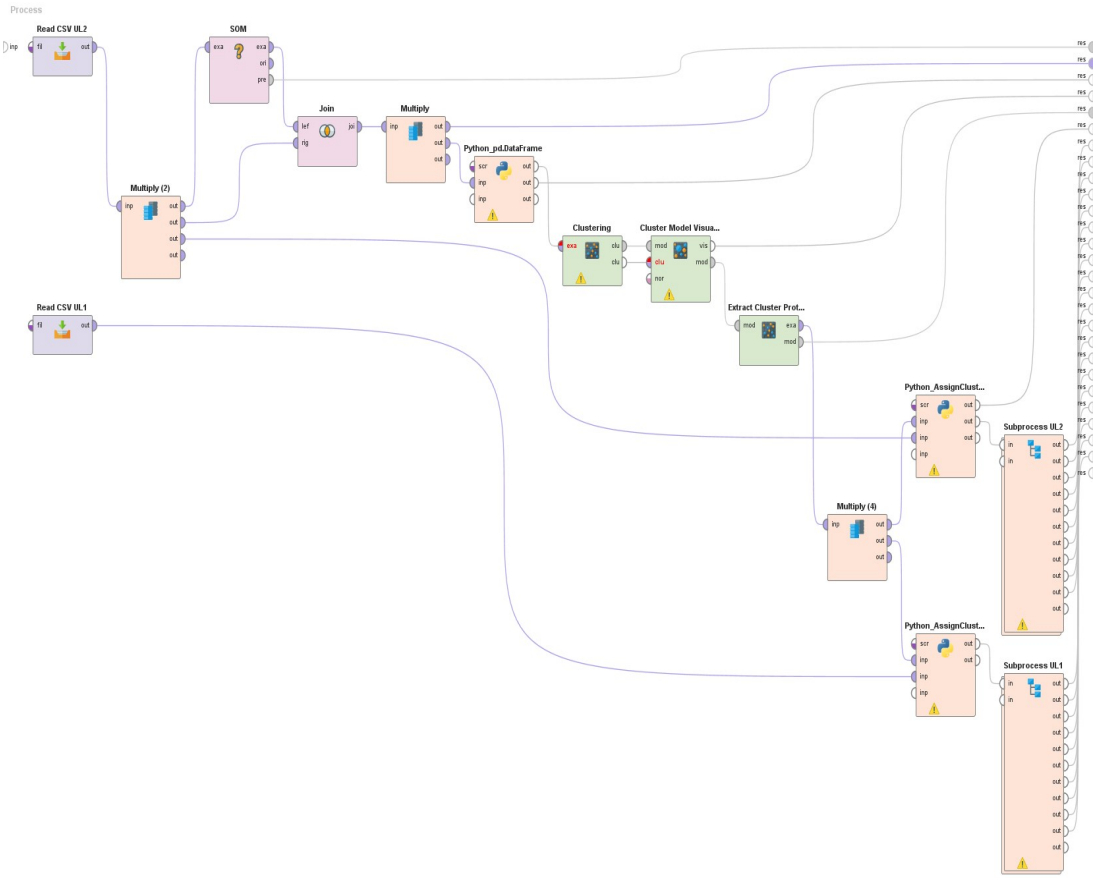
Součástí předložené práce v adresáři: **bee_audio\Anotace\Anotace ul.XLS**

Příloha B - Vizualizace shlukové analýzy včelařského roku

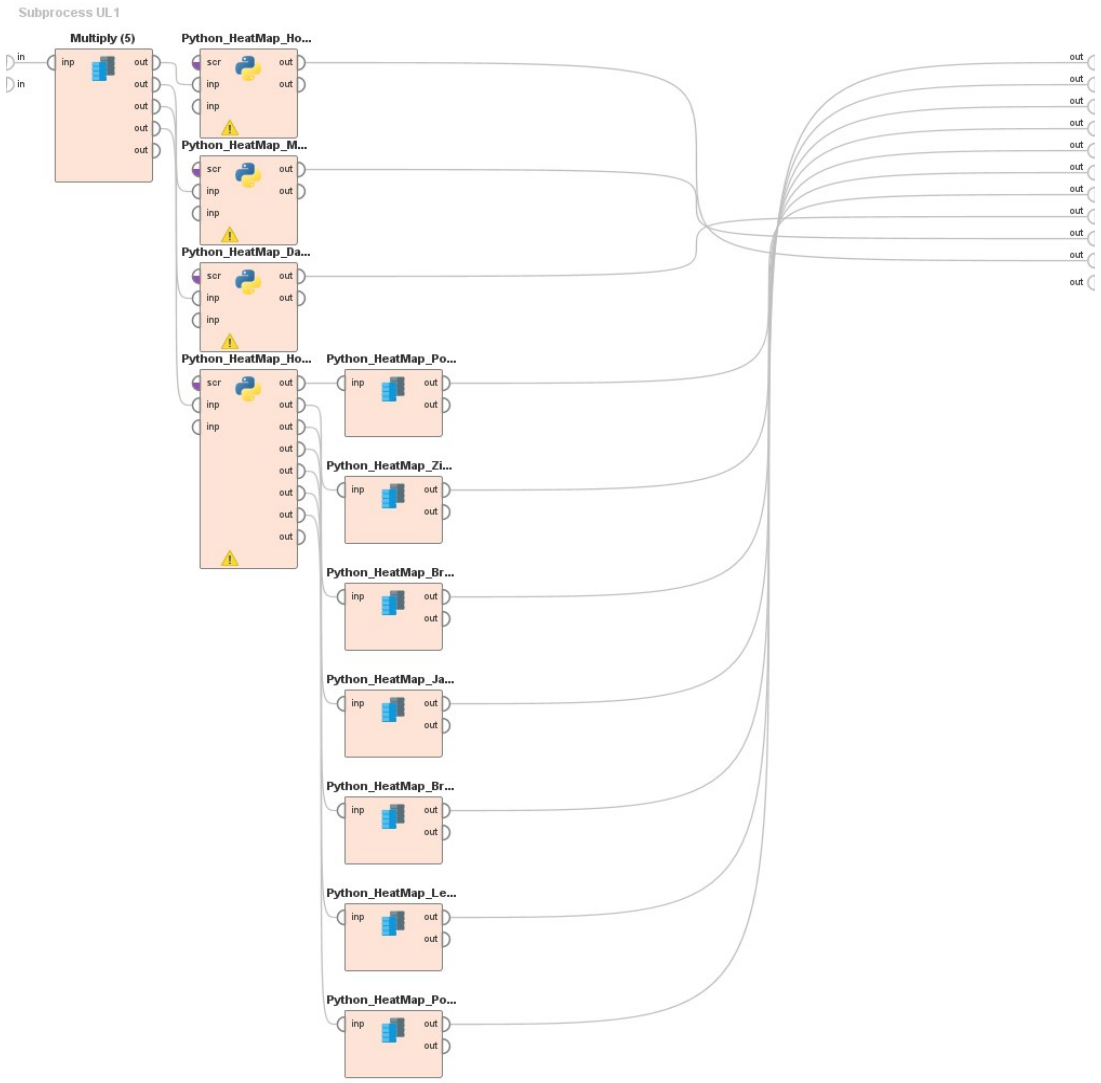
Součástí předložené práce v adresáři: **bee_audio\RapidMiner\VIZUALIZACE SHLUKOVÉ ANALÝZY VČELAŘSKÉHO ROKU POMOCÍ HEATMAP.pdf**

Příloha C - Proces zpracování shlukové analýzy v RapidMineru

Process



Subprocess UL1 a Subprocess UL2



Příloha D - Zdrojové kódy

- Zdrojové kódy jsou součástí předložené práce v adresáři **bee_audio** a **bee_audio\RapidMiner\Scripts**.

- Zdrojové kódy pro autentizované uživatele po připojení k VPN JČU na GitLabu na adrese: https://bee.prf.jcu.cz:3100/mskrbek/bp-analyza-dat-z-inteligentniho-vceliho-ulu